Teorie kódování - Kapitola 2
Věta o kódování bez šumu pro zdroje bez paměti

Jan Paseka

Ústav matematiky a statistiky
Masarykova univerzita

28. února 2023

Obsah

Základní pojmy

FI V této kapitole dokážeme první a snadnější ze dvou Shannonových hlavních vět pro nejjednodušší třídu zdrojů.

Základní pojmy

FI V této kapitole dokážeme první a snadnější ze dvou Shannonových hlavních vět pro nejjednodušší třídu zdrojů.

Stručný oxfordský slovník definuje ***zdroj*** jako pramen, vrchol zřídla, ze kterého proudí výstupy.

Základní pojmy

FI V této kapitole dokážeme první a snadnější ze dvou Shannonových hlavních vět pro nejjednodušší třídu zdrojů.

Stručný oxfordský slovník definuje ***zdroj*** jako pramen, vrchol zřídla, ze kterého proudí výstupy.

Ve své plné obecnosti, je to přesně to, co uvažujeme v teorii informace, ačkoliv typicky považujeme ***zdroj za proud symbolů jisté konečné abecedy***.

Základní pojmy

FI V této kapitole dokážeme první a snadnější ze dvou Shannonových hlavních vět pro nejjednodušší třídu zdrojů.

Stručný oxfordský slovník definuje ***zdroj*** jako pramen, vrchol zřídla, ze kterého proudí výstupy.

Ve své plné obecnosti, je to přesně to, co uvažujeme v teorii informace, ačkoliv typicky považujeme ***zdroj za proud symbolů jisté konečné abecedy***.

Zdroj má obvykle nějaký náhodný mechanismus, který je založen na statistice situace, která je modelovaná.

Základní pojmy

FI V této kapitole dokážeme první a snadnější ze dvou Shannonových hlavních vět pro nejjednodušší třídu zdrojů.

Stručný oxfordský slovník definuje ***zdroj*** jako pramen, vrchol zřídla, ze kterého proudí výstupy.

Ve své plné obecnosti, je to přesně to, co uvažujeme v teorii informace, ačkoliv typicky považujeme ***zdroj za proud symbolů jisté konečné abecedy***.

Zdroj má obvykle nějaký náhodný mechanismus, který je založen na statistice situace, která je modelovaná.

Tento náhodný mechanismus může být poměrně dost komplikovaný, ale my se budeme pro okamžik soustředit na následující opravdu speciální a jednoduchý příklad.

Základní pojmy

Značí-li X_i i-tý symbol vytvořený zdrojem, dohodneme se pak, že, pro každý symbol a_j, ***pravděpodobnost***

$$P(X_i = a_j) = p_j$$

je nezávislá na i a tedy je nezávislá na všech minulých nebo v budoucnosti vyslaných symbolech.

Základní pojmy

Značí-li X_i i-tý symbol vytvořený zdrojem, dohodneme se pak, že, pro každý symbol a_j, ***pravděpodobnost***

$$P(X_i = a_j) = p_j$$

je nezávislá na i a tedy je nezávislá na všech minulých nebo v budoucnosti vyslaných symbolech.

Jinak řečeno, $X_1, X_2, \ldots$ je právě posloupnost identicky distribuovaných, nezávislých náhodných veličin. Takovýto zdroj nazveme ***zdrojem s nulovou pamětí*** nebo ***zdrojem bez paměti*** a jeho entropie H je definována jako

$$H = -\sum p_j \log p_j$$

kde sčítáme přes množinu j takových, že $p_j > 0$.

Procvičování

Cvičení 1.1

1. *Je-li $\mathcal{S}$ zdroj bez paměti s abecedou Σ,* **rozšíření řádu** *n S je zdroj bez paměti $S^{(n)}$ s abecedou $\Sigma^{(n)}$ skládající se ze všech řetězců délky n symbolů ze Σ tak, že pravděpodobnost každého řetězce σ je určena pravděpodobností, že je to řetězec prvních n symbolů vyslaných S. Dokažte, že $S^{(n)}$ má entropii*

$$H(S^{(n)}) = nH(S).$$

Obsah

Základní pojmy

FI Hlavní problém řešený v této kapitole je následující.

Předpokládejme, že máme zdroj bez paměti $\mathcal{S}$, který vysílá symboly z abecedy $W = \{w_1, \ldots, w_n\}$ s pravděpodobnostmi $\{p_1, \ldots, p_n\}$.

Základní pojmy

FI Hlavní problém řešený v této kapitole je následující.

Předpokládejme, že máme zdroj bez paměti $\mathcal{S}$, který vysílá symboly z abecedy $W = \{w_1, \ldots, w_n\}$ s pravděpodobnostmi $\{p_1, \ldots, p_n\}$.

Z pedagogických důvodů budeme prvky W nazývat ***zdrojová slova*** a ptát se na následující otázku.

Základní pojmy

FI Hlavní problém řešený v této kapitole je následující.

Předpokládejme, že máme zdroj bez paměti $\mathcal{S}$, který vysílá symboly z abecedy $W = \{w_1, \ldots, w_n\}$ s pravděpodobnostmi $\{p_1, \ldots, p_n\}$.

Z pedagogických důvodů budeme prvky W nazývat ***zdrojová slova*** a ptát se na následující otázku.

Je-li Σ abeceda D symbolů, **jak můžeme zakódovat zdrojová slova** w_i pomocí symbolů z Σ, abychom dostali **co možná nejekonomičtější zakódování?**

Příklad

Příklad 2.1

Předpokládejme, že zdroj $\mathcal{S}$ vysílá čtyři zdrojová slova a, b, c, d s pravděpodobnostmi

$$p_a = 0.9, \quad p_b = 0.05, \quad p_c = p_d = 0.025.$$

Srovnáme-li pak zakódování

$$a \rightsquigarrow 0, \quad b \rightsquigarrow 111, \quad c \rightsquigarrow 110, \quad d \rightsquigarrow 101,$$

a

$$a \rightsquigarrow 00, \quad b \rightsquigarrow 01, \quad c \rightsquigarrow 10, \quad d \rightsquigarrow 11,$$

je evidentně průměrná délka zakódovaného zdroje 1.2 v prvním kódu a 2 v druhém kódu.

Základní pojmy - Kódování

Formálněji, ***kódování*** nebo ***kód*** je zobrazení f z $\{w_1, \ldots, w_n\}$ do Σ^*, kde Σ^* označuje soubor konečných řetězců symbolů z Σ.

Základní pojmy - Kódování

Formálněji, ***kódování*** nebo ***kód*** je zobrazení f z $\{w_1, \ldots, w_n\}$ do Σ^*, kde Σ^* označuje soubor konečných řetězců symbolů z Σ.

Zpráva je každý konečný řetězec zdrojových slov a, je-li

$$m = w_{i_1} \ldots w_{i_k}$$

a je-li f kódování, pak ***rozšíření*** f na W^* je definováno obvyklým způsobem pomocí zřetězení

$$f(m) = f(w_{i_1}) \ldots f(w_{i_k}).$$

Základní pojmy - Jednoznačně dekódovatelné kódy

Kódování f je ***jednoznačně dekódovatelné***, jestliže každý konečný řetězec z Σ^* je obraz nejvýše jedné zprávy.

Základní pojmy - Jednoznačně dekódovatelné kódy

Kódování f je ***jednoznačně dekódovatelné***, jestliže každý konečný řetězec z Σ^* je obraz nejvýše jedné zprávy.

Řetězce $f(w_i)$ se nazývají ***kódová slova*** a přirozená čísla $|f(w_i)|$ jsou ***slovní délky*** kódování f.

Základní pojmy - Jednoznačně dekódovatelné kódy

Kódování f je ***jednoznačně dekódovatelné***, jestliže každý konečný řetězec z Σ^* je obraz nejvýše jedné zprávy.

Řetězce $f(w_i)$ se nazývají ***kódová slova*** a přirozená čísla $|f(w_i)|$ jsou ***slovní délky*** kódování f.

Průměrná délka $\langle f \rangle$ kódování f je definovaná jako

$$\langle f \rangle = \sum_{i=1}^{m} p_i |f(w_i)|.$$

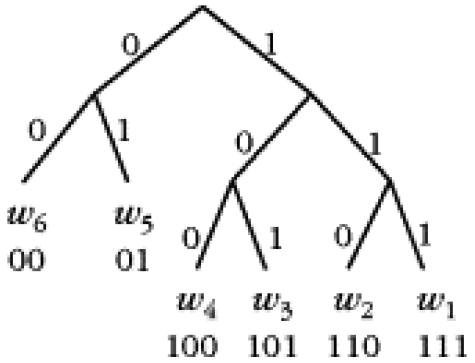
Základní pojmy - Prefixové kódy

Kódování f se nazývá ***bezprostředně dekódovatelné*** nebo ***prefixové***, jestliže neexistují různé w_i a w_j tak, že $f(w_i)$ je prefix $f(w_j)$.

Základní pojmy - Prefixové kódy

Kódování f se nazývá ***bezprostředně dekódovatelné*** nebo ***prefixové***, jestliže neexistují různé w_i a w_j tak, že $f(w_i)$ je prefix $f(w_j)$.

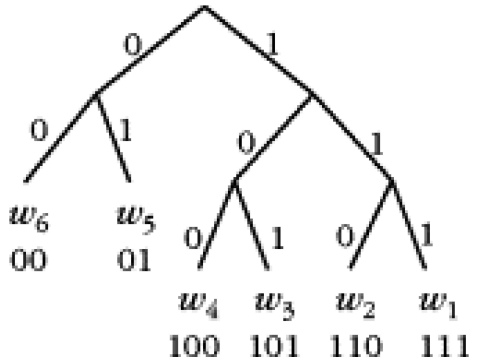
Zde používáme, jak lze očekávat, ***prefix*** v obvyklém smyslu, že pokud $x, y \in \Sigma^*$, pak x je prefix y, jestliže existuje $z \in \Sigma^*$ tak, že $xz = y$.



Základní pojmy - Prefixové kódy

Kódování f se nazývá ***bezprostředně dekódovatelné*** nebo ***prefixové***, jestliže neexistují různé w_i a w_j tak, že $f(w_i)$ je prefix $f(w_j)$.

Zde používáme, jak lze očekávat, ***prefix*** v obvyklém smyslu, že pokud $x, y \in \Sigma^*$, pak x je prefix y, jestliže existuje $z \in \Sigma^*$ tak, že $xz = y$.



Prefixová kódování jsou jednoznačně dekódovatelná. Skutečně, mají silnější vlastnost, že prefixový kód může být dekódován ‘on line’ bez pohledu do budoucnosti.

Příklad - Prefixové kódy

Příklad 2.2

Předpokládejme, že $\Sigma = \{0, 1\}$ *a máme čtyři zdrojová slova* $w_1, .., w_4$.

Příklad - Prefixové kódy

Příklad 2.2

Předpokládejme, že $\Sigma = \{0, 1\}$ *a máme čtyři zdrojová slova* $w_1, .., w_4$.

Prefixové kódování je

$$f(w_1) = 0, \quad f(w_2) = 10, \quad f(w_3) = 110, \quad f(w_4) = 1110.$$

Příklad - Prefixové kódy

Příklad 2.2

Předpokládejme, že $\Sigma = \{0, 1\}$ *a máme čtyři zdrojová slova* $w_1, .., w_4$.

Prefixové kódování je

$$f(w_1) = 0, \quad f(w_2) = 10, \quad f(w_3) = 110, \quad f(w_4) = 1110.$$

Například zprávu 0110100101001 0 *lze dekódovat jako* $w_1 w_3 w_2 w_1 w_2 w_2 w_1 w_2$.

Příklad - Prefixové kódy

Příklad 2.2

Předpokládejme, že $\Sigma = \{0, 1\}$ *a máme čtyři zdrojová slova* $w_1, .., w_4$.

Prefixové kódování je

$$f(w_1) = 0, \quad f(w_2) = 10, \quad f(w_3) = 110, \quad f(w_4) = 1110.$$

Například zprávu 0110100101001 0 *lze dekódovat jako* $w_1 w_3 w_2 w_1 w_2 w_2 w_1 w_2$.

(Toto je příklad toho, co je známo jako čárkové *kódování, protože evidentně používáme nulu, abychom signalizovali konec slova.)*

Příklad - Prefixové kódy

Ne každé jednoznačně dekódovatelné kódování je prefixové.

Příklad 2.3

Předpokládejme, že $W = \{w_1, w_2\}, \Sigma = \{0, 1\}$ *a kódování* g *je definováno jako*

$$g(w_1) = 0, \quad g(w_2) = 01.$$

Příklad - Prefixové kódy

Ne každé jednoznačně dekódovatelné kódování je prefixové.

Příklad 2.3

Předpokládejme, že $W = \{w_1, w_2\}, \Sigma = \{0, 1\}$ *a kódování* g *je definováno jako*

$$g(w_1) = 0, \quad g(w_2) = 01.$$

Toto kódování ***není evidentně prefixové****, ale lze snadno ověřit, že* ***je jednoznačně dekódovatelné****, pokud budeme postupovat zpět z konce zprávy.*

Příklad - Prefixové kódy

Ne každé jednoznačně dekódovatelné kódování je prefixové.

Příklad 2.3

Předpokládejme, že $W = \{w_1, w_2\}, \Sigma = \{0, 1\}$ *a kódování* g *je definováno jako*

$$g(w_1) = 0, \quad g(w_2) = 01.$$

Toto kódování ***není evidentně prefixové****, ale lze snadno ověřit, že* ***je jednoznačně dekódovatelné****, pokud budeme postupovat zpět z konce zprávy.*
Je zřejmé, že jednoznačně dekódovatelná kódování jsou o mnoho obtížnější pojem, než prefixová kódování.

Příklad - Prefixové kódy

Ne každé jednoznačně dekódovatelné kódování je prefixové.

Příklad 2.3

Předpokládejme, že $W = \{w_1, w_2\}, \Sigma = \{0, 1\}$ *a kódování* g *je definováno jako*

$$g(w_1) = 0, \quad g(w_2) = 01.$$

Toto kódování ***není evidentně prefixové****, ale lze snadno ověřit, že* ***je jednoznačně dekódovatelné****, pokud budeme postupovat zpět z konce zprávy.*
Je zřejmé, že jednoznačně dekódovatelná kódování jsou o mnoho obtížnější pojem, než prefixová kódování.
Překvapivě ukážeme, že můžeme omezit pozornost na prefixová kódování v našem hledání jednoznačně dekódovatelných kódování, která mají minimální průměrnou délku.

Jiný pohled na prefixové kódování

Poznámka 2.4

Uvažujme D-ární strom, tj. každý uzel buď má D následníků nebo je listem. Nechť dále větve stromu reprezentují symboly příslušných kódových slov. Např., D větví vycházejících z kořenového uzlu reprezentuje možné výskyty prvního symbolu kódového slova. Pak každé kódové slovo je reprezentováno listem našeho stromu. Cesta od kořene k listu se pak řídí symboly kódového slova. Evidentně, protože kódová slova jsou právě listy, není žádné kódové slovo předchůdcem jiného kódového slova, tj. naše kódování je prefixové.

Jiný pohled na prefixové kódování

Poznámka 2.4

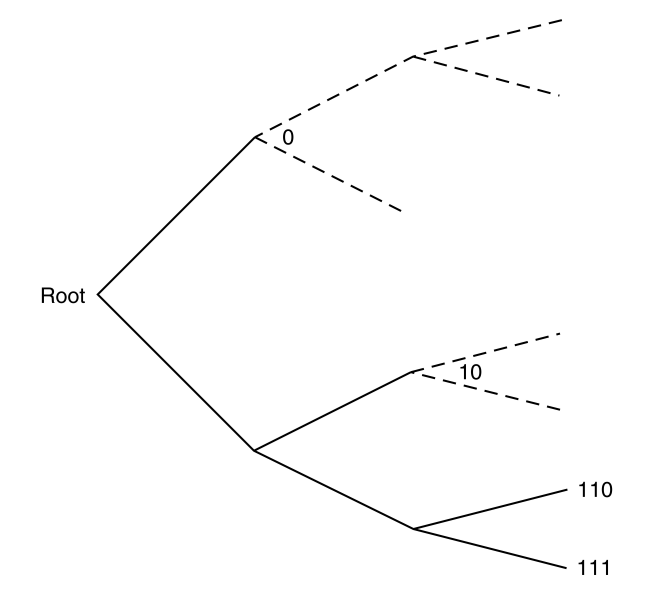
Uvažujme D-ární strom, tj. každý uzel buď má D následníků nebo je listem. Nechť dále větve stromu reprezentují symboly příslušných kódových slov. Např., D větví vycházejících z kořenového uzlu reprezentuje možné výskyty prvního symbolu kódového slova. Pak každé kódové slovo je reprezentováno listem našeho stromu. Cesta od kořene k listu se pak řídí symboly kódového slova. Evidentně, protože kódová slova jsou právě listy, není žádné kódové slovo předchůdcem jiného kódového slova, tj. naše kódování je prefixové.

Obráceně, mějme prefixové kódování na abecedě o D písmenech a uvažme maximální D-ární strom, který má hloubku rovnu nejdelšímu slovu našeho kódování.

Jiný pohled na prefixové kódování

Poznámka 2.4

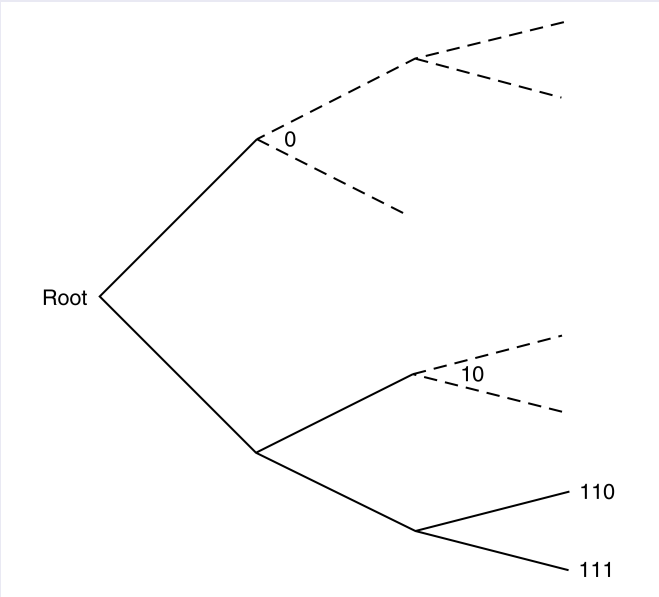
Podmínka, že se jedná o prefixové kódování implikuje, že žádné kódové slovo není předchůdcem jiného kódového slova v našem stromu. Zejména tedy každé kódové slovo eliminuje všechny své následníky jakožto možná kódová slova.



Jiný pohled na prefixové kódování

Poznámka 2.4

Podmínka, že se jedná o prefixové kódování implikuje, že žádné kódové slovo není předchůdcem jiného kódového slova v našem stromu. Zejména tedy každé kódové slovo eliminuje všechny své následníky jakožto možná kódová slova.



Lze tedy z maximálního stromu odebrat všechny následníky kódových slov a obdržíme pak D-ární strom, kde listy odpovídají kódovým slovům.

Jiný pohled na prefixové kódování

Poznámka 2.5

Ačkoliv jsme definovali kódování jako zobrazení, často ho identifikujeme se souborem C kódových slov.

Jiný pohled na prefixové kódování

Poznámka 2.5

Ačkoliv jsme definovali kódování jako zobrazení, často ho identifikujeme se souborem C kódových slov.

Cvičení 2.6

1. *Ukažte, že pro každé přirozené číslo m existuje prefixové kódování nad* $\{0, 1\}$*, které má slova všech délek v množině* $\{1, \ldots, m\}$.

Obsah

Kraftova a McMillanova nerovnosti

FI V tomto odstavci dokážeme dvě základní nerovnosti, které ospravedlňují naši dřívější poznámku, že můžeme v podstatě zapomenout na pojem jednoznačné dekódovatelnosti a omezit pozornost na prefixová kódování.

Kraftova a McMillanova nerovnosti

FI V tomto odstavci dokážeme dvě základní nerovnosti, které ospravedlňují naši dřívější poznámku, že můžeme v podstatě zapomenout na pojem jednoznačné dekódovatelnosti a omezit pozornost na prefixová kódování.

Nejdříve vyslovme nerovnosti.

KRAFTOVA NEROVNOST

Kraftova a McMillanova nerovnosti

FI V tomto odstavci dokážeme dvě základní nerovnosti, které ospravedlňují naši dřívější poznámku, že můžeme v podstatě zapomenout na pojem jednoznačné dekódovatelnosti a omezit pozornost na prefixová kódování.

Nejdříve vyslovme nerovnosti.

KRAFTOVA NEROVNOST

Je-li Σ abeceda mohutnosti D a W obsahuje N slov, pak nutná a dostatečná podmínka, že existuje prefixové kódování $f : W \to \Sigma^*$ se slovními délkami $l_1, \ldots, l_N$ je, že platí

$$\sum_{i=1}^{N} D^{-l_i} \leq 1. \tag{3.1}$$

Kraftova a McMillanova nerovnosti

McMILLANOVA NEROVNOST

Je-li Σ abeceda mohutnosti D a W obsahuje N slov, pak nutná a dostatečná podmínka, že existuje jednoznačně dekódovatelné kódování $f : W \to \Sigma^*$ se slovními délkami $l_1, \ldots, l_N$ je, že platí (3.1).

Kraftova a McMillanova nerovnosti

McMILLANOVA NEROVNOST

Je-li Σ abeceda mohutnosti D a W obsahuje N slov, pak nutná a dostatečná podmínka, že existuje jednoznačně dekódovatelné kódování $f : W \to \Sigma^*$ se slovními délkami $l_1, \ldots, l_N$ je, že platí (3.1).

Kombinací těchto dvou nerovností dostaneme:

Věta 3.1

Jednoznačně dekódovatelné kódování s předepsanou délkou slov existuje právě tehdy, když existuje prefixový kód se stejnou délkou slov.

Důkaz Věty 3.1

DŮKAZ KRAFTOVY NEROVNOSTI

Předpokládejme, že množina $\{l_1, \ldots, l_N\}$ splňuje

$$\sum_{i=1}^{N} D^{-l_i} \leq 1.$$

Důkaz Věty 3.1

DŮKAZ KRAFTOVY NEROVNOSTI

Předpokládejme, že množina $\{l_1, \ldots, l_N\}$ splňuje

$$\sum_{i=1}^{N} D^{-l_i} \leq 1.$$

Přepišme nerovnost do tvaru

$$\sum_{j=1}^{l} n_j D^{-j} \leq 1,$$

kde n_j je počet l_i rovných j, $l = \max l_i$ a vynásobme ji D^l.

Důkaz Věty 3.1

DŮKAZ KRAFTOVY NEROVNOSTI

Předpokládejme, že množina $\{l_1, \ldots, l_N\}$ splňuje

$$\sum_{i=1}^{N} D^{-l_i} \leq 1.$$

Přepišme nerovnost do tvaru

$$\sum_{j=1}^{l} n_j D^{-j} \leq 1,$$

kde n_j je počet l_i rovných j, $l = \max l_i$ a vynásobme ji D^l.
Přepišme tuto nerovnost opět do tvaru

$$n_l \leq D^l - n_1 D^{l-1} - \ldots - n_{l-1} D. \qquad (3.2)$$

Důkaz Věty 3.1 - Kraftova nerovnost

Protože n_j jsou všechna nezáporná, postupně dostaneme z (3.2) nerovnosti

$$
\begin{aligned}
n_{l-1} &\leq D^{l-1} - n_1 D^{l-2} - \ldots - n_{l-2} D, \\
n_{l-2} &\leq D^{l-2} - n_1 D^{l-3} - \ldots - n_{l-3} D, \\
&\vdots \\
n_3 &\leq D^3 - n_1 D^2 - \ldots - n_2 D, \\
n_2 &\leq D^2 - n_1 D, \\
n_1 &\leq D.
\end{aligned}
\tag{3.3}
$$

Důkaz Věty 3.1 - Kraftova nerovnost

Protože n_j jsou všechna nezáporná, postupně dostaneme z (3.2) nerovnosti

$$
\begin{aligned}
n_{l-1} &\leq D^{l-1} - n_1 D^{l-2} - \ldots - n_{l-2} D, \\
n_{l-2} &\leq D^{l-2} - n_1 D^{l-3} - \ldots - n_{l-3} D,
\end{aligned}
$$

$$
\begin{aligned}
n_3 &\leq D^3 - n_1 D^2 - \ldots - n_2 D, \\
n_2 &\leq D^2 - n_1 D, \\
n_1 &\leq D.
\end{aligned}
\tag{3.3}
$$

Tyto nerovnosti jsou klíč ke konstrukci kódování s danou délkou slov.

Důkaz Věty 3.1 - Kraftova nerovnost

Nejdříve vyberme n_1 slov délky 1, přičemž použijeme různá písmena z Σ. Zbývá nám $D - n_1$ nepoužitých symbolů a můžeme vytvořit $(D - n_1)D$ slov délky 2 přidáním písmena ke každému z těchto symbolů.

Důkaz Věty 3.1 - Kraftova nerovnost

Nejdříve vyberme n_1 slov délky 1, přičemž použijeme různá písmena z Σ. Zbývá nám $D - n_1$ nepoužitých symbolů a můžeme vytvořit $(D - n_1)D$ slov délky 2 přidáním písmena ke každému z těchto symbolů.

Vyberme našich n_2 délky 2 libovolně z těchto slov a zbývá nám pak $D^2 - n_1 D - n_2$ prefixů délky 2.

Důkaz Věty 3.1 - Kraftova nerovnost

Nejdříve vyberme n_1 slov délky 1, přičemž použijeme různá písmena z Σ. Zbývá nám $D - n_1$ nepoužitých symbolů a můžeme vytvořit $(D - n_1)D$ slov délky 2 přidáním písmena ke každému z těchto symbolů.

Vyberme našich n_2 délky 2 libovolně z těchto slov a zbývá nám pak $D^2 - n_1 D - n_2$ prefixů délky 2.

Tyto lze užít pro vytvoření $(D^2 - n_1 D - n_2)D$ slov délky 3, ze kterých můžeme vybrat n_3 libovolně atd. Pokračujeme-li tímto způsobem, pokaždé je zachována vlastnost, že žádné slovo není prefixem jiného.

Důkaz Věty 3.1 - Kraftova nerovnost

Nejdříve vyberme n_1 slov délky 1, přičemž použijeme různá písmena z Σ. Zbývá nám $D - n_1$ nepoužitých symbolů a můžeme vytvořit $(D - n_1)D$ slov délky 2 přidáním písmena ke každému z těchto symbolů.

Vyberme našich n_2 délky 2 libovolně z těchto slov a zbývá nám pak $D^2 - n_1 D - n_2$ prefixů délky 2.

Tyto lze užít pro vytvoření $(D^2 - n_1 D - n_2)D$ slov délky 3, ze kterých můžeme vybrat n_3 libovolně atd. Pokračujeme-li tímto způsobem, pokaždé je zachována vlastnost, že žádné slovo není prefixem jiného.

V každém případě zjistíme, že nerovnosti (3.3) nám dovolí provést tento výběr. Tedy skončíme s prefixovým kódováním s předepsanými délkami kódování.

Důkaz Věty 3.1 - McMillanova nerovnost

Dokázali jsme, že numerická podmínka (3.1) je dostatečná pro existenci prefixového kódování. Ačkoliv Kraft rovněž dokázal i nutnost podmínky (3.1), jedná se o bezprostřední důsledek McMillanovy nerovnosti, kterou v dalším dokážeme. Podaný důkaz je o mnoho jednodušší, než McMillanův původní důkaz a patří Karushovi (1961).

Důkaz Věty 3.1 - McMillanova nerovnost

Dokázali jsme, že numerická podmínka (3.1) je dostatečná pro existenci prefixového kódování. Ačkoliv Kraft rovněž dokázal i nutnost podmínky (3.1), jedná se o bezprostřední důsledek McMillanovy nerovnosti, kterou v dalším dokážeme. Podaný důkaz je o mnoho jednodušší, než McMillanův původní důkaz a patří Karushovi (1961).

DŮKAZ McMILLANOVY NEROVNOSTI

Předpokládejme, že máme jednoznačně dekódovatelné kódování C s délkami slov $l_1, \ldots, l_N$.

Důkaz Věty 3.1 - McMillanova nerovnost

Dokázali jsme, že numerická podmínka (3.1) je dostatečná pro existenci prefixového kódování. Ačkoliv Kraft rovněž dokázal i nutnost podmínky (3.1), jedná se o bezprostřední důsledek McMillanovy nerovnosti, kterou v dalším dokážeme. Podaný důkaz je o mnoho jednodušší, než McMillanův původní důkaz a patří Karushovi (1961).

DŮKAZ McMILLANOVY NEROVNOSTI

Předpokládejme, že máme jednoznačně dekódovatelné kódování C s délkami slov $l_1, \ldots, l_N$.
Pokud $l = \max l_i$, pak, pro každé kladné celé číslo r, máme

$$\left(D^{-l_1} + \cdots + D^{-l_N}\right)^r = \sum_{i=1}^{rl} b_i D^{-i}, \qquad (3.4)$$

kde b_i je nezáporné celé číslo.

Důkaz Věty 3.1 - McMillanova nerovnost

Ale celá čísla b_i jsou právě počet možností, kolika způsoby lze řetězec délky i z symbolů abecedy Σ utvořit konkatenací r kódových slov z délek vybraných z množiny $\{l_1, \ldots, l_N\}$.

Důkaz Věty 3.1 - McMillanova nerovnost

Ale celá čísla b_i jsou právě počet možností, kolika způsoby lze řetězec délky i z symbolů abecedy Σ utvořit konkatenací r kódových slov z délek vybraných z množiny $\{l_1, \ldots, l_N\}$.

Jelikož je kódování C jednoznačně dekódovatelné, každý řetězec délky i tvořený z kódových slov musí odpovídat nejvýše jedné posloupnosti kódových slov. Musíme tedy mít

$$b_i \leq D^i \qquad (1 \leq i \leq rl).$$

Důkaz Věty 3.1 - McMillanova nerovnost

Ale celá čísla b_i jsou právě počet možností, kolika způsoby lze řetězec délky i z symbolů abecedy Σ utvořit konkatenací r kódových slov z délek vybraných z množiny $\{l_1, \ldots, l_N\}$.

Jelikož je kódování C jednoznačně dekódovatelné, každý řetězec délky i tvořený z kódových slov musí odpovídat nejvýše jedné posloupnosti kódových slov. Musíme tedy mít

$$b_i \leq D^i \qquad (1 \leq i \leq rl).$$

Dosadíme-li do (3.4), obdržíme

$$\left(D^{-l_1} + \cdots + D^{-l_N}\right)^r \leq lr.$$

Důkaz Věty 3.1 - McMillanova nerovnost

Proto

$$\sum_{j=1}^{l} n_j D^{-j} \leq l^{1/r} r^{1/r},$$

a protože r bylo libovolné kladné celé číslo, dostáváme limitním přechodem $r \to \infty$ na pravé straně McMillanovu nerovnost.

Důkaz Věty 3.1 - McMillanova nerovnost

Proto

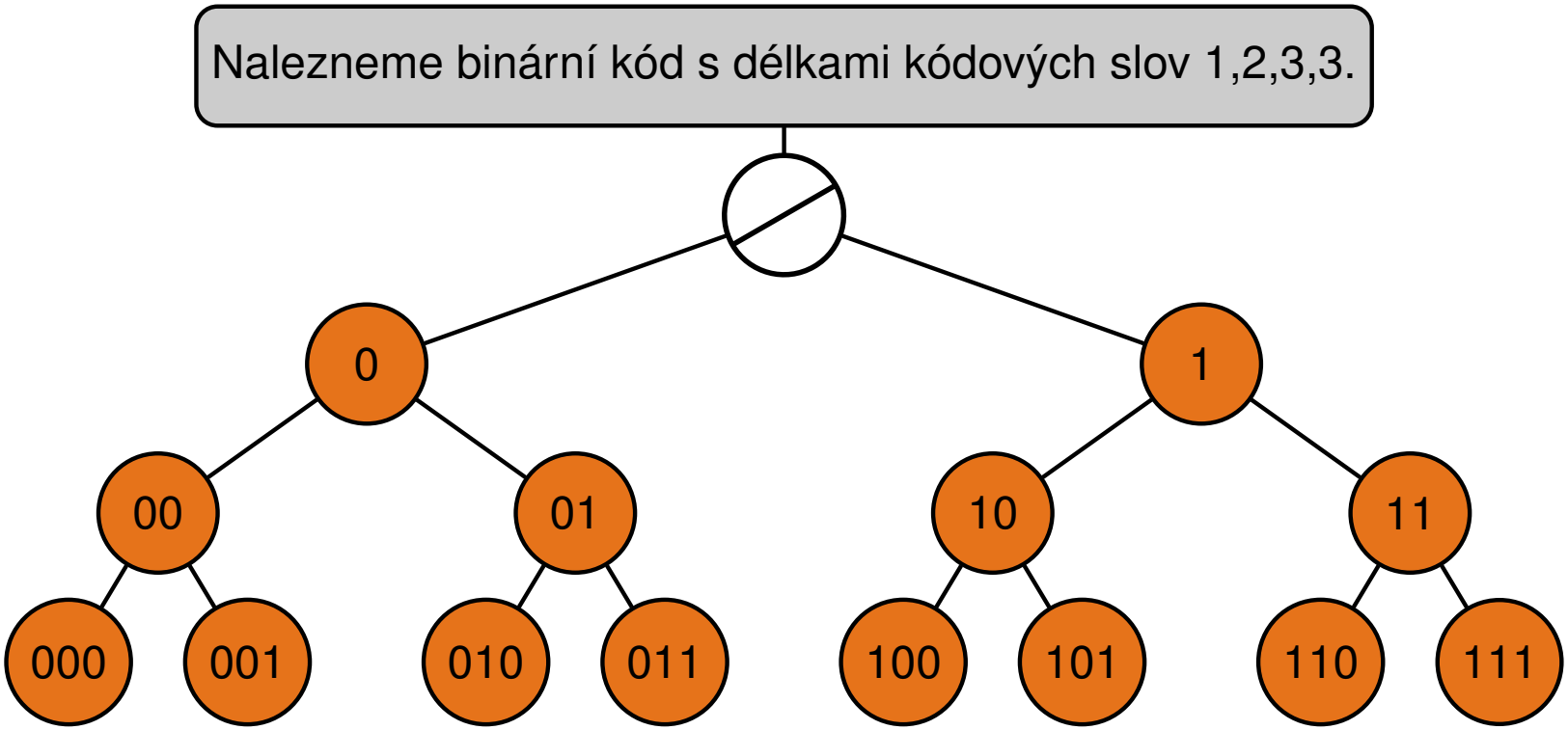
$$\sum_{j=1}^{l} n_j D^{-j} \leq l^{1/r} r^{1/r},$$

a protože r bylo libovolné kladné celé číslo, dostáváme limitním přechodem $r \to \infty$ na pravé straně McMillanovu nerovnost.

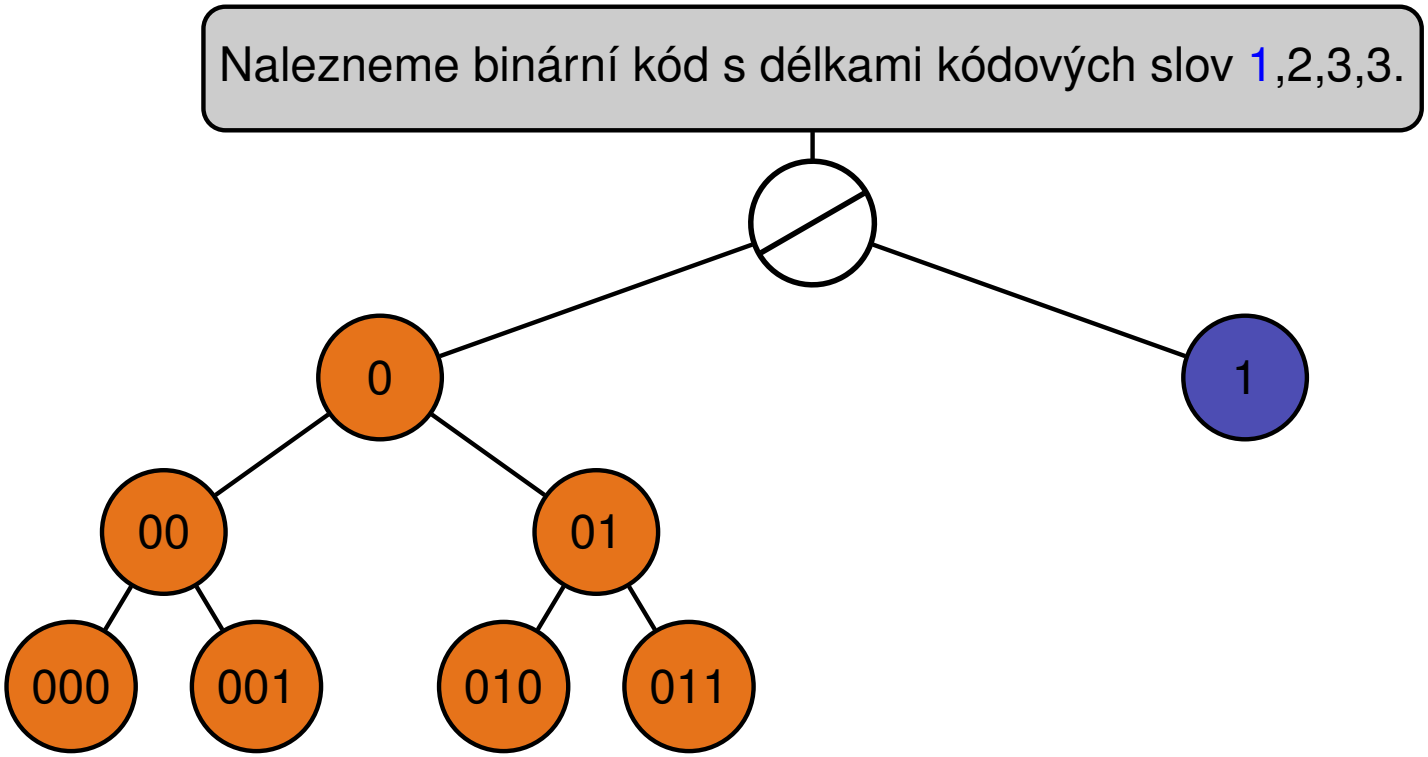
Cvičení 3.2

1. *Jaký je maximální počet slov binárního prefixového kódování, ve kterém je maximální délka slova 7?*

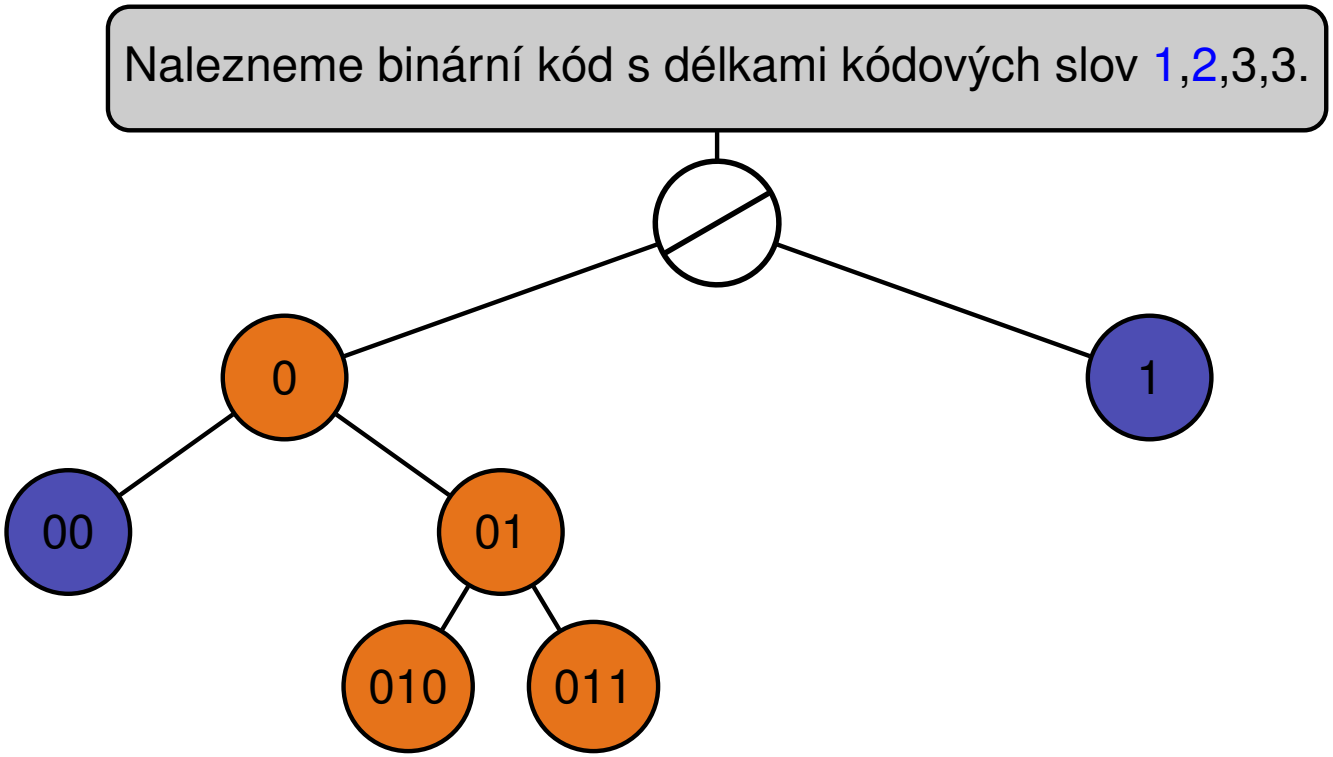
Ilustrace Kraftovy nerovnosti



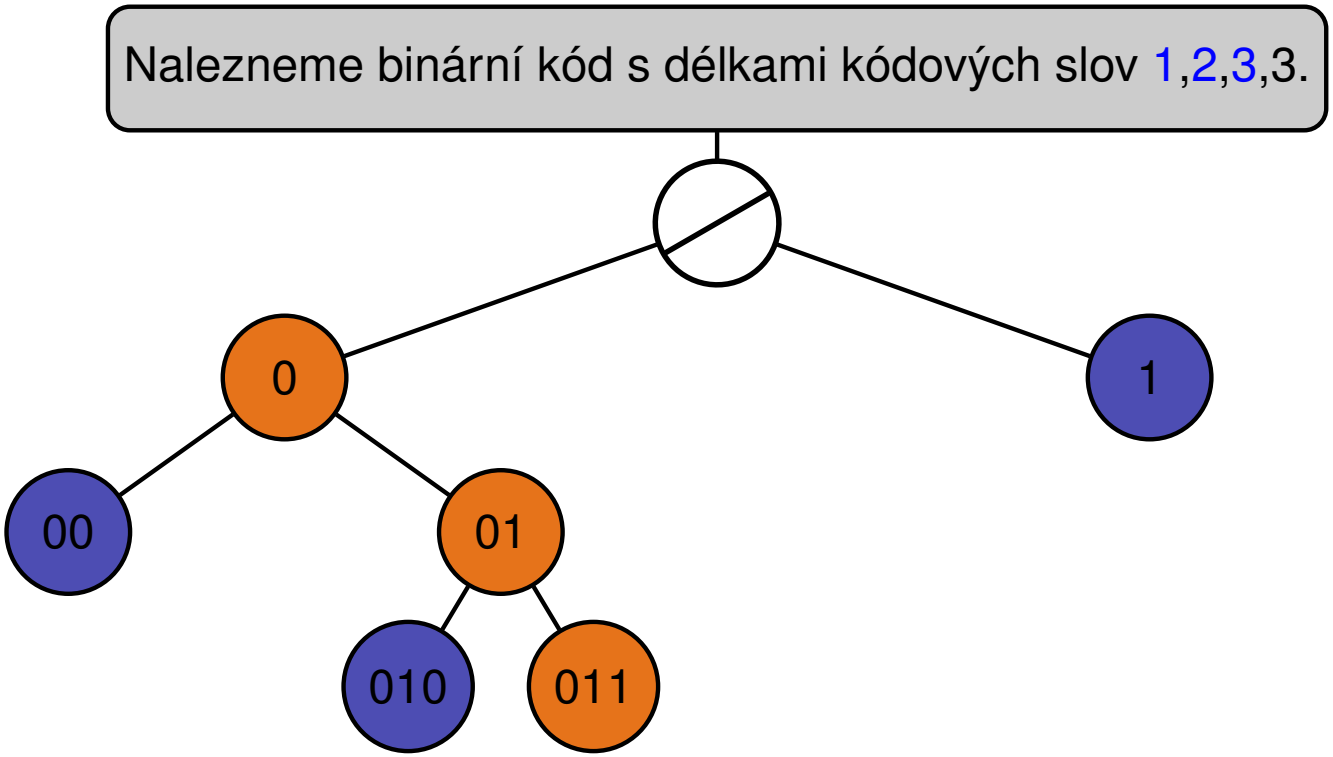
Ilustrace Kraftovy nerovnosti



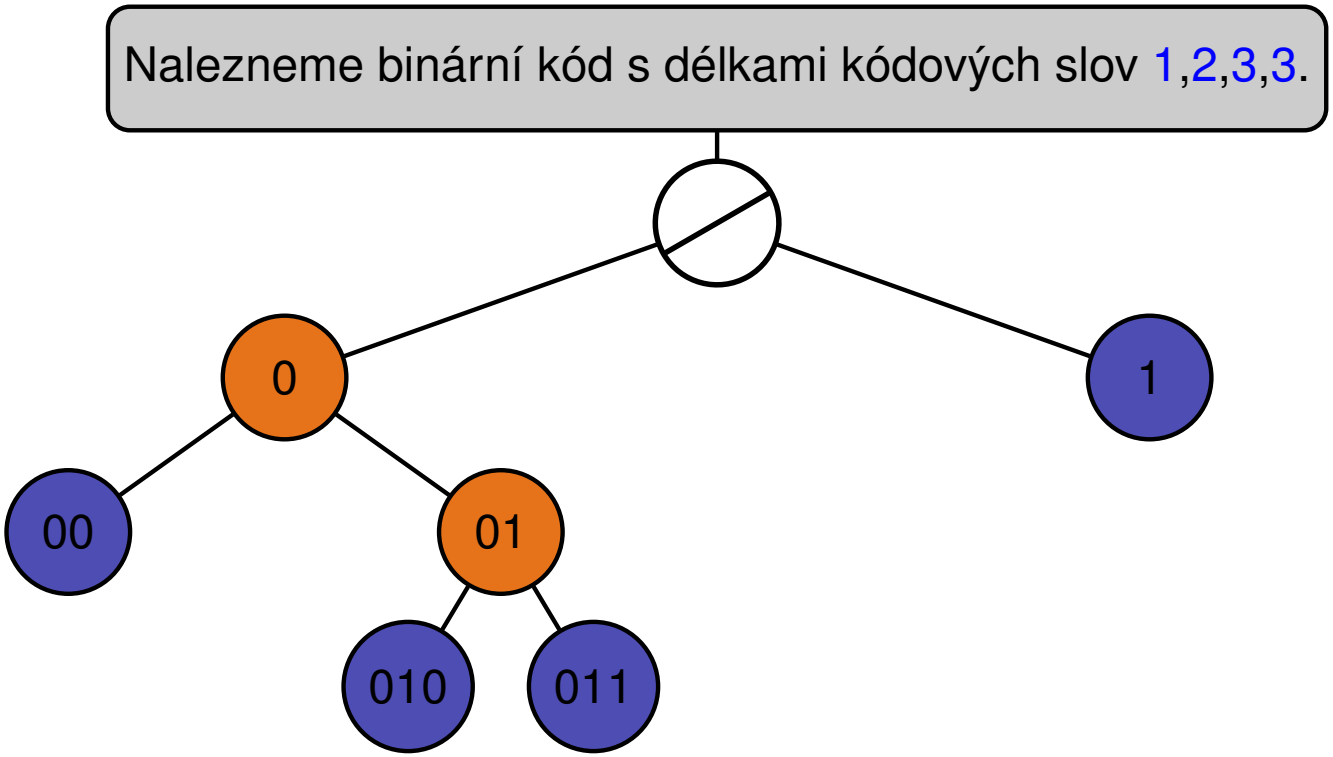
Ilustrace Kraftovy nerovnosti



Ilustrace Kraftovy nerovnosti



Ilustrace Kraftovy nerovnosti



Obsah

Věta o kódování bez šumu pro zdroje bez paměti

FI Uvažujme nyní následující situaci. Mějme zdroj $\mathcal{S}$ bez paměti, který vysílá slova $w_1, \ldots, w_N$ s (kladnými) pravděpodobnostmi $p_1, \ldots, p_N$, každé vyslané slovo je vybráno nezávisle na všech jiných slovech.

Věta o kódování bez šumu pro zdroje bez paměti

FI Uvažujme nyní následující situaci. Mějme zdroj $\mathcal{S}$ bez paměti, který vysílá slova $w_1, \ldots, w_N$ s (kladnými) pravděpodobnostmi $p_1, \ldots, p_N$, každé vyslané slovo je vybráno nezávisle na všech jiných slovech.

Náš problém je: je-li dán takovýto zdroj společně s abecedou Σ, najděte jednoznačně dekódovatelné kódování, jež má minimální průměrnou délku slov. Takovéto kódování nazýváme ***kompaktní***.

Věta o kódování bez šumu pro zdroje bez paměti

FI Uvažujme nyní následující situaci. Mějme zdroj $\mathcal{S}$ bez paměti, který vysílá slova $w_1, \ldots, w_N$ s (kladnými) pravděpodobnostmi $p_1, \ldots, p_N$, každé vyslané slovo je vybráno nezávisle na všech jiných slovech.

Náš problém je: je-li dán takovýto zdroj společně s abecedou Σ, najděte jednoznačně dekódovatelné kódování, jež má minimální průměrnou délku slov. Takovéto kódování nazýváme ***kompaktní***.

Heuristický přístup k tomuto problému by mohl být následující. Zdroj $\mathcal{S}$ má entropii

$$H = -\sum p_i \log p_i.$$

Věta o kódování bez šumu pro zdroje bez paměti

Maximální entropie v abecedě o D písmenech je $\log D$. Tedy počet symbolů abecedy potřebný v průměru na zakódování slova ze zdroje by měl být asi $H/\log D$.

Věta o kódování bez šumu pro zdroje bez paměti

Maximální entropie v abecedě o D písmenech je $\log D$. Tedy počet symbolů abecedy potřebný v průměru na zakódování slova ze zdroje by měl být asi $H/\log D$.

Tuto hrubou ideu nyní upřesníme.

Věta 4.1

Má-li zdroj bez paměti entropii H, pak každé jednoznačně dekódovatelné kódování C pro tento zdroj v abecedě o D symbolech musí mít délku alespoň $H/\log D$. Navíc existuje takové jednoznačně dekódovatelné kódování, které má průměrnou délku slov menší než $1 + H/\log D$.
Rovnost $l(C) = H/\log D$ platí právě tehdy, když $p_i = D^{-l_i}$, kde l_i je délka zakódování slova w_i.

Důkaz Věty 4.1 - Kódování bez šumu

Předpokládejme, že máme jednoznačně dekódovatelné kódování C s délkami slov $l_1, \ldots, l_N$. Předpokládejme dále, že pravděpodobnosti emitovaných slov odpovídajících těmto délkám jsou $p_1, \ldots, p_N$.

Důkaz Věty 4.1 - Kódování bez šumu

Předpokládejme, že máme jednoznačně dekódovatelné kódování C s délkami slov $l_1, \ldots, l_N$. Předpokládejme dále, že pravděpodobnosti emitovaných slov odpovídajících těmto délkám jsou $p_1, \ldots, p_N$.

Tedy

$$H = -\sum p_i \log p_i$$

a průměrná délka kódování C je dána

$$l(C) = \sum p_i l_i.$$

Důkaz Věty 4.1 - Kódování bez šumu

Předpokládejme, že máme jednoznačně dekódovatelné kódování C s délkami slov $l_1, \ldots, l_N$. Předpokládejme dále, že pravděpodobnosti emitovaných slov odpovídajících těmto délkám jsou $p_1, \ldots, p_N$.

Tedy

$$H = -\sum p_i \log p_i$$

a průměrná délka kódování C je dána

$$l(C) = \sum p_i l_i.$$

Z Kraft–McMillanovy nerovnosti víme, že

$$G = \sum_{j=1}^{l} n_j D^{-j} \leq 1.$$

Důkaz Věty 4.1 - Kódování bez šumu

Definujme q_i $(1 \leq i \leq N)$ jako

$$q_i = D^{-l_i}/G,$$

tedy je $(q_1, \ldots, q_N)$ pravděpodobnostní rozdělení.

Důkaz Věty 4.1 - Kódování bez šumu

Definujme q_i $(1 \leq i \leq N)$ jako

$$q_i = D^{-l_i}/G,$$

tedy je $(q_1, \ldots, q_N)$ pravděpodobnostní rozdělení.

Aplikujme Klíčové Lemma a obdržíme pak

$$H = -\sum p_i \log p_i \leq -\sum p_i \log q_i.$$

Důkaz Věty 4.1 - Kódování bez šumu

Definujme q_i $(1 \leq i \leq N)$ jako

$$q_i = D^{-l_i}/G,$$

tedy je $(q_1, \ldots, q_N)$ pravděpodobnostní rozdělení.

Aplikujme Klíčové Lemma a obdržíme pak

$$H = -\sum p_i \log p_i \leq -\sum p_i \log q_i.$$

Ale

$$\log q_i = -l_i \log D - \log G.$$

Důkaz Věty 4.1 - Kódování bez šumu

Definujme q_i $(1 \leq i \leq N)$ jako

$$q_i = D^{-l_i}/G,$$

tedy je $(q_1, \ldots, q_N)$ pravděpodobnostní rozdělení.

Aplikujme Klíčové Lemma a obdržíme pak

$$H = -\sum p_i \log p_i \leq -\sum p_i \log q_i.$$

Ale

$$\log q_i = -l_i \log D - \log G.$$

Tedy

$$H \leq \left(\sum p_i l_i\right) \log D + \left(\sum p_i\right) \log G.$$

Důkaz Věty 4.1 - Kódování bez šumu

Ale $G \leq 1$ z Kraft-McMillanovy nerovnosti a tedy, jak je požadováno,

$$H \leq l(C) \log D.$$

Důkaz Věty 4.1 - Kódování bez šumu

Ale $G \leq 1$ z Kraft-McMillanovy nerovnosti a tedy, jak je požadováno,

$$H \leq l(C) \log D.$$

Dále platí

$$\begin{aligned} 0 \leq l(C) - H/\log D &= \sum (p_i/\log D) \log \frac{p_i}{q_i} - \log G/\log D \\ &= D(\mathbf{p}||\mathbf{q})/\log D + \log \frac{1}{G}/\log D. \end{aligned}$$

Pravá strana je rovna nule právě tehdy, když $\mathbf{p} = \mathbf{q}$.

Důkaz Věty 4.1 - Kódování bez šumu

Abychom dokázali horní hranici, vybereme naše délky slov $l_1, \ldots, l_N$ podle pravidla, že pro všechna i je délka l_i minimální přirozené číslo splňující

$$p_i^{-1} \leq D^{l_i}, \text{ tj } D^{-l_i} \leq p_i. \tag{4.1}$$

Důkaz Věty 4.1 - Kódování bez šumu

Abychom dokázali horní hranici, vybereme naše délky slov $l_1, \ldots, l_N$ podle pravidla, že pro všechna i je délka l_i minimální přirozené číslo splňující

$$p_i^{-1} \leq D^{l_i}, \text{ tj } D^{-l_i} \leq p_i. \tag{4.1}$$

Ale, protože $p_1 + \cdots + p_N = 1$, toto implikuje

$$\sum_{i=1}^{N} D^{-l_i} \leq 1.$$

Důkaz Věty 4.1 - Kódování bez šumu

Odtud víme, že existuje jednoznačně dekódovatelné (ve skutečnosti prefixové) kódování s těmito slovními délkami.

Důkaz Věty 4.1 - Kódování bez šumu

Odtud víme, že existuje jednoznačně dekódovatelné (ve skutečnosti prefixové) kódování s těmito slovními délkami. Ale, protože (4.1) je ekvivalentní s

$$l_i \log D \geq -\log p_i$$

a l_i je minimální vzhledem k této vlastnosti, víme, že

$$l_i < 1 - \log p_i / \log D.$$

Důkaz Věty 4.1 - Kódování bez šumu

Odtud víme, že existuje jednoznačně dekódovatelné (ve skutečnosti prefixové) kódování s těmito slovními délkami. Ale, protože (4.1) je ekvivalentní s

$$l_i \log D \geq -\log p_i$$

a l_i je minimální vzhledem k této vlastnosti, víme, že

$$l_i < 1 - \log p_i / \log D.$$

Máme tedy, že

$$l(C) = \sum p_i l_i < 1 + H/\log D.$$

Cvičení

Cvičení 4.2

1. *Zakódujeme-li n stejně pravděpodobných slov nad binární abecedou, věta o kódování bez šumu tvrdí, že průměrná délka slova $l(C)$ každého kompaktního jednoznačně dekódovatelného kódování splňuje*

$$\log_2 n \leq l(C),$$

pro které hodnoty n platí rovnost?

Cvičení

Cvičení 4.2

1. *Zakódujeme-li n stejně pravděpodobných slov nad binární abecedou, věta o kódování bez šumu tvrdí, že průměrná délka slova $l(C)$ každého kompaktního jednoznačně dekódovatelného kódování splňuje*

$$\log_2 n \leq l(C),$$

   pro které hodnoty n platí rovnost?

2. *Srovnejme hranice věty o kódování bez šumu s délkou kompaktního zakódování $2^k - 1$ stejně pravděpodobných slov nad $\{0, 1\}$.*

Shannonovo kódování

Předpokládejme, že máme dán zdroj $\mathcal{S}$ bez paměti ze slov $w_1, \ldots, w_N$ s pravděpodobnostmi $p_1, \ldots, p_N$ a že si přejeme najít kompaktní kódování $C_S^{\mathbf{p}}$ pro $\mathcal{S}$ nad abecedou $\{0, 1, \ldots, D-1\}$.

- bez újmy na obecnosti předpokládáme $p_i > 0$,

Shannonovo kódování

Předpokládejme, že máme dán zdroj $\mathcal{S}$ bez paměti ze slov $w_1, \ldots, w_N$ s pravděpodobnostmi $p_1, \ldots, p_N$ a že si přejeme najít kompaktní kódování $C_S^{\mathbf{p}}$ pro $\mathcal{S}$ nad abecedou $\{0, 1, \ldots, D-1\}$.

- bez újmy na obecnosti předpokládáme $p_i > 0$,
- položíme

$$l_i = \left\lceil \log_D p_i^{-1} \right\rceil,$$

Shannonovo kódování

Předpokládejme, že máme dán zdroj $\mathcal{S}$ bez paměti ze slov $w_1, \ldots, w_N$ s pravděpodobnostmi $p_1, \ldots, p_N$ a že si přejeme najít kompaktní kódování $C_S^{\mathbf{p}}$ pro $\mathcal{S}$ nad abecedou $\{0, 1, \ldots, D-1\}$.

- bez újmy na obecnosti předpokládáme $p_i > 0$,
- položíme
$$l_i = \left\lceil \log_D p_i^{-1} \right\rceil,$$
- tato čísla splňují Kraftovu nerovnost:
$$\sum_{i=1}^{N} D^{-\left\lceil \log_D p_i^{-1} \right\rceil} \leq \sum_{i=1}^{N} D^{-\log_D p_i^{-1}} = \sum_{i=1}^{N} p_i = 1,$$

Shannonovo kódování

Předpokládejme, že máme dán zdroj $\mathcal{S}$ bez paměti ze slov $w_1, \ldots, w_N$ s pravděpodobnostmi $p_1, \ldots, p_N$ a že si přejeme najít kompaktní kódování $C_S^{\mathbf{p}}$ pro $\mathcal{S}$ nad abecedou $\{0, 1, \ldots, D-1\}$.

- bez újmy na obecnosti předpokládáme $p_i > 0$,
- položíme
$$l_i = \left\lceil \log_D p_i^{-1} \right\rceil,$$
- tato čísla splňují Kraftovu nerovnost:
$$\sum_{i=1}^{N} D^{-\left\lceil \log_D p_i^{-1} \right\rceil} \leq \sum_{i=1}^{N} D^{-\log_D p_i^{-1}} = \sum_{i=1}^{N} p_i = 1,$$
- slova kódování $C_S^{\mathbf{p}}$ nalezneme pomocí konstrukce prefixového kódu se zadanými délkami l_i.

Meze Shannonova kódování

Použití Shannonova kódování může být mnohem horší než kompaktní kódování.

Meze Shannonova kódování

Použití Shannonova kódování může být mnohem horší než kompaktní kódování.

Například uvažme dva symboly, z nichž jeden se vyskytuje s pravděpodobností 0,9999 a druhý s pravděpodobností 0,0001.

Meze Shannonova kódování

Použití Shannonova kódování může být mnohem horší než kompaktní kódování.

Například uvažme dva symboly, z nichž jeden se vyskytuje s pravděpodobností 0,9999 a druhý s pravděpodobností 0,0001.

V případě Shannonova kódování použijeme délku kódového slova 1 bit, respektive 14 bitů. V případě kompaktního kódování to bude zjevně 1 bit pro oba symboly.

Meze Shannonova kódování

Použití Shannonova kódování může být mnohem horší než kompaktní kódování.

Například uvažme dva symboly, z nichž jeden se vyskytuje s pravděpodobností 0,9999 a druhý s pravděpodobností 0,0001.

V případě Shannonova kódování použijeme délku kódového slova 1 bit, respektive 14 bitů. V případě kompaktního kódování to bude zjevně 1 bit pro oba symboly.

Rozdíl průměrných délek pak bude 0,0013 ve prospěch kompaktního kódování.

Meze Shannonova kódování

Důsledek 4.3

Platí

$$H(\mathbf{p})/\log D \leq l(C_S^{\mathbf{p}}) < 1 + H(\mathbf{p})/\log D.$$

Meze Shannonova kódování

Důsledek 4.3

Platí

$$H(\mathbf{p})/\log D \leq l(C_S^{\mathbf{p}}) < 1 + H(\mathbf{p})/\log D.$$

Věta 4.4 (Divergence je cena za nevhodné kódování)

Buď dán zdroj $\mathcal{S}$ bez paměti ze slov $w_1, \ldots, w_N$ s (kladnými) pravděpodobnostmi $p_1, \ldots, p_N$. Nechť dále $\mathbf{q} = (q_1, \ldots, q_N)$ je (kladné) rozdělení pravděpodobností a $C_S^{\mathbf{q}}$ je Shannonův kód zkonstruovaný na základě rozdělení $\mathbf{q}$ v abecedě o D symbolech. Potom platí:

$$[H(X) + D(\mathbf{p}||\mathbf{q})]/\log D \leq l_{\mathbf{p}}(C_S^{\mathbf{q}}) < [H(X) + D(\mathbf{p}||\mathbf{q})]/\log D + 1.$$

Důkaz Věty 4.4 - Divergence

Nejprve dokažme horní mez. Platí:

$$\begin{aligned}
l_{\mathbf{p}}(C_S^{\mathbf{q}}) &= \sum_{i=1}^{N} p_i \left\lceil \log_D q_i^{-1} \right\rceil < \sum_{i=1}^{N} p_i \cdot (\log_D q_i^{-1} + 1) \\
&= \sum_{i=1}^{N} p_i \cdot \log_D \left(\frac{p_i}{q_i} \cdot \frac{1}{p_i} \right) + \sum_{i=1}^{N} p_i \\
&= \sum_{i=1}^{N} p_i \cdot \log_D \frac{p_i}{q_i} + \sum_{i=1}^{N} p_i \cdot \log_D \frac{1}{p_i} + 1 \\
&= \left[\sum_{i=1}^{N} p_i \cdot \log \frac{p_i}{q_i} + \sum_{i=1}^{N} p_i \cdot \log \frac{1}{p_i} \right] / \log D + 1 \\
&= [H(X) + D(\mathbf{p}||\mathbf{q})] / \log D + 1.
\end{aligned}$$

Důkaz Věty 4.4 - Divergence

Nyní obdobně dokažme dolní mez. Platí:

$$\begin{aligned}
l_{\mathbf{p}}(C_S^{\mathbf{q}}) &= \sum_{i=1}^{N} p_i \left\lceil \log_D q_i^{-1} \right\rceil \geq \sum_{i=1}^{N} p_i \cdot \log_D q_i^{-1} \\
&= \sum_{i=1}^{N} p_i \cdot \log_D \left(\frac{p_i}{q_i} \cdot \frac{1}{p_i} \right) \\
&= \sum_{i=1}^{N} p_i \cdot \log_D \frac{p_i}{q_i} + \sum_{i=1}^{N} p_i \cdot \log_D \frac{1}{p_i} \\
&= \left[\sum_{i=1}^{N} p_i \cdot \log \frac{p_i}{q_i} + \sum_{i=1}^{N} p_i \cdot \log \frac{1}{p_i} \right] / \log D \\
&= [H(X) + D(\mathbf{p}||\mathbf{q})] / \log D.
\end{aligned}$$

Obsah

Konstruování kompaktních kódování

FI Předpokládejme, že máme dán zdroj $\mathcal{S}$ bez paměti ze slov $w_1, \ldots, w_N$ s pravděpodobnostmi $p_1, \ldots, p_N$ a že si přejeme najít kompaktní kódování C pro $\mathcal{S}$ nad abecedou Σ.

Konstruování kompaktních kódování

FI Předpokládejme, že máme dán zdroj $\mathcal{S}$ bez paměti ze slov $w_1, \ldots, w_N$ s pravděpodobnostmi $p_1, \ldots, p_N$ a že si přejeme najít kompaktní kódování C pro $\mathcal{S}$ nad abecedou Σ.

Z věty o kódování bez šumu víme, že průměrná délka musí splňovat ohraničení

$$H/\log D \leq l(C) < 1 + H/\log D, \tag{5.1}$$

ale snadno se vidí, že dolní hranice lze dosáhnout pouze, když p_i jsou jisté celočíselné mocniny D.

Konstruování kompaktních kódování

FI Předpokládejme, že máme dán zdroj $\mathcal{S}$ bez paměti ze slov $w_1, \ldots, w_N$ s pravděpodobnostmi $p_1, \ldots, p_N$ a že si přejeme najít kompaktní kódování C pro $\mathcal{S}$ nad abecedou Σ.

Z věty o kódování bez šumu víme, že průměrná délka musí splňovat ohraničení

$$H/\log D \leq l(C) < 1 + H/\log D, \tag{5.1}$$

ale snadno se vidí, že dolní hranice lze dosáhnout pouze, když p_i jsou jisté celočíselné mocniny D.

Z Kraft-McMillanovy nerovnosti máme:
Jestliže existuje kompaktní jednoznačně dekódovatelné kódování o průměrné délce l, pak existuje kompaktní prefixové kódování o průměrné délce l.

Konstruování kompaktních kódování

Můžeme se tedy omezit na prefixová kódování. Nyní popíšeme metodu navrhnutou Huffmanem v roce 1952 pro konstruování kompaktníhu prefixového kódování pro výše uvedený zdroj $\mathcal{S}$ v případě, že Σ je binární abeceda.

Konstruování kompaktních kódování

Můžeme se tedy omezit na prefixová kódování. Nyní popíšeme metodu navrhnutou Huffmanem v roce 1952 pro konstruování kompaktníhu prefixového kódování pro výše uvedený zdroj $\mathcal{S}$ v případě, že Σ je binární abeceda.

Výsledný kód není určen jednoznačně (např. bitovou inverzí nebo permutací na abecedě Σ získáme jiný optimální kód).

Konstruování kompaktních kódování

Můžeme se tedy omezit na prefixová kódování. Nyní popíšeme metodu navrhnutou Huffmanem v roce 1952 pro konstruování kompaktníhu prefixového kódování pro výše uvedený zdroj $\mathcal{S}$ v případě, že Σ je binární abeceda.

Výsledný kód není určen jednoznačně (např. bitovou inverzí nebo permutací na abecedě Σ získáme jiný optimální kód).

Jednoznačně není zadána ani délka kódových slov.

Konstruování kompaktních kódování

Můžeme se tedy omezit na prefixová kódování. Nyní popíšeme metodu navrhnutou Huffmanem v roce 1952 pro konstruování kompaktníhu prefixového kódování pro výše uvedený zdroj $\mathcal{S}$ v případě, že Σ je binární abeceda.

Výsledný kód není určen jednoznačně (např. bitovou inverzí nebo permutací na abecedě Σ získáme jiný optimální kód).

Jednoznačně není zadána ani délka kódových slov.

Huffmanovo kódování se používá na závěrečné zpracování formátů JPEG, MP3, DEFLATE, PKZIP resp. na předzpracování souboru pro aritmetické kódování.

Konstruování kompaktních kódování

Nejprve dokážeme některé vlastnosti kompaktního prefixového kódování C nad $\Sigma = \{0, 1\}$. Budeme užívat $l(w)$ k označení délky slova w v C.

Lemma 5.1

Kompaktní kódování pro zdroj s právě dvěma slovy w_1 a w_2 je

$$w_1 \to 0, \qquad w_2 \to 1.$$

Konstruování kompaktních kódování

Nejprve dokážeme některé vlastnosti kompaktního prefixového kódování C nad $\Sigma = \{0, 1\}$. Budeme užívat $l(w)$ k označení délky slova w v C.

Lemma 5.1

Kompaktní kódování pro zdroj s právě dvěma slovy w_1 a w_2 je

$$w_1 \to 0, \qquad w_2 \to 1.$$

Důkaz je zřejmý.

Konstruování kompaktních kódování

Lemma 5.2

Je-li C prefixové a kompaktní kódování a $p_i > p_j$, pak $l(w_i) \leq l(w_j)$.

Konstruování kompaktních kódování

Lemma 5.2

Je-li C prefixové a kompaktní kódování a $p_i > p_j$, pak $l(w_i) \leq l(w_j)$.

Důkaz. Pokud tomu tak není, vytvořme nový kód C' z C záměnou zakódování w_i a w_j. Pak průměrná délka je zmenšena a stále máme prefixové kódování.

Konstruování kompaktních kódování

Lemma 5.2

Je-li C prefixové a kompaktní kódování a $p_i > p_j$, pak $l(w_i) \leq l(w_j)$.

Důkaz. Pokud tomu tak není, vytvořme nový kód C' z C záměnou zakódování w_i a w_j. Pak průměrná délka je zmenšena a stále máme prefixové kódování.

Lemma 5.3

Je-li C prefixové a kompaktní kódování, pak mezi všemi kódovými slovy v C maximální délky musí existovat alespoň dvě lišící se pouze v poslední číslici.

Důkaz. Nechť tomu tak není; pak můžeme odebrat poslední číslici z těchto všech kódových slov maximální délky a stále máme prefixové kódování, což je spor s kompaktností C.

HUFFMANŮV KÓDOVACÍ ALGORITMUS

Beze ztráty obecnosti můžeme předpokládat, že zdroj $\mathcal{S}$ má svůj systém zdrojových slov $\{w_1, \ldots, w_N\}$ uspořádaných tak, že pravděpodobnosti p_i vyslání w_i splňují

$$p_1 \geq p_2 \geq \cdots \geq p_N.$$

HUFFMANŮV KÓDOVACÍ ALGORITMUS

Beze ztráty obecnosti můžeme předpokládat, že zdroj $\mathcal{S}$ má svůj systém zdrojových slov $\{w_1, \ldots, w_N\}$ uspořádaných tak, že pravděpodobnosti p_i vyslání w_i splňují

$$p_1 \geq p_2 \geq \cdots \geq p_N.$$

Huffmanova procedura konstruuje rekurzivně posloupnost zdrojů $S_0, S_1, \ldots, S_{N-2}$ tak, že $S_0 = S$ a S_k je získáno z S_{k-1} ztotožněním dvou nejméně pravděpodobných symbolů z S_{k-1} s jediným symbolem σ z S_k. Pravděpodobnost, že symbol σ je vyslán z S_k je brána jako součet pravděpodobností jeho dvou vytvářejících symbolů v S_{k-1}.

HUFFMANŮV KÓDOVACÍ ALGORITMUS

Tedy S_1 je získán z S_0 ztotožněním w_N a w_{N-1} do jednoho symbolu w_{N-1} vyskytujícího se s pravděpodobností $p_N + p_{N-1}$.

HUFFMANŮV KÓDOVACÍ ALGORITMUS

Tedy S_1 je získán z S_0 ztotožněním w_N a w_{N-1} do jednoho symbolu w_{N-1} vyskytujícího se s pravděpodobností $p_N + p_{N-1}$.

V každém stavu máme zdroj s o jeden méně symboly.

HUFFMANŮV KÓDOVACÍ ALGORITMUS

Tedy S_1 je získán z S_0 ztotožněním w_N a w_{N-1} do jednoho symbolu w_{N-1} vyskytujícího se s pravděpodobností $p_N + p_{N-1}$.

V každém stavu máme zdroj s o jeden méně symboly.

Po $N - 2$ takovýchto redukcích dospějeme k zdroji S_{N-2}, který má pouze dva symboly.

Přechod mezi S_{j-1} a S_j lze nejlépe vidět na následujícím obrázku.

HUFFMANŮV KÓDOVACÍ ALGORITMUS

Zakódování	Pravděpodobnost	Slovo	Slovo	Pravděpodobnost	Zakódování
	q_1	v_1	u_1	q_1	
	q_2	v_2	u_2	q_2	
	q_{k-1}	v_{k-1}	u_{k-1}	q_{k-1}	
	q_k	v_k	u_k	$q_t + q_{t+1}$	
	q_{k+1}	v_{k+1}	u_{k+1}	q_k	
	q_{t-1}	v_{t-1}			
	q_t	v_t	u_t	q_{t-1}	
	q_{t+1}	v_{t+1}			
	S_{j-1}			S_j	

HUFFMANŮV KÓDOVACÍ ALGORITMUS

Je-li dáno zakódování $\sigma_1, \ldots, \sigma_t$ zdroje S_j, Huffmanova procedura pro nalezení zakódování zdroje S_{j-1} je následující velmi snadné pravidlo.
Předpokládejme pravděpodobnosti $q_1 \geq q_2 \geq \cdots \geq q_{t+1}$ slov z S_{j-1} jsou takové, že slovo vytvořené z v_t a v_{t+1} je slovo u_k z S_j. Pak by Huffmanovo zakódování S_{j-1} mělo být, jak je ukázáno v levém sloupci předchozí tabulky. Formálně, mělo by být zadáno pravidlem

$$v_i \to \sigma_i \qquad (1 \leq i \leq k-1), v_i \to \sigma_{i+1} \qquad (k \leq i \leq t-1),$$
$$v_t \to (\sigma_k, 0), \qquad v_{t+1} \to (\sigma_k, 1).$$

HUFFMANŮV KÓDOVACÍ ALGORITMUS

Zakódování	Pravděpodobnost	Slovo	Slovo	Pravděpodobnost	Zakódování
σ_1	q_1	v_1	u_1	q_1	σ_1
σ_2	q_2	v_2	u_2	q_2	σ_2
σ_{k-1}	q_{k-1}	v_{k-1}	u_{k-1}	q_{k-1}	σ_{k-1}
σ_{k+1}	q_k	v_k	u_k	$q_t + q_{t+1}$	σ_k
σ_{k+2}	q_{k+1}	v_{k+1}	u_{k+1}	q_k	σ_{k+1}
σ_t	q_{t-1}	v_{t-1}			
$(\sigma_k, 0)$	q_t	v_t	u_t	q_{t-1}	σ_t
$(\sigma_k, 1)$	q_{t+1}	v_{t+1}			
	S_{j-1}			S_j	

HUFFMANŮV KÓDOVACÍ ALGORITMUS

Zpětným zpracováním nastartujeme naši zakódovací proceduru zakódováním dvou slov z S_{N-2} s dvěma kódovými slovy 0 a 1; pak S_{N-3} bude mít tři kódová slova atd. a budeme pokračovat ve výše uvedené proceduře, až dosáhneme Huffmanova kódu pro $S = S_0$.

HUFFMANŮV KÓDOVACÍ ALGORITMUS

Zpětným zpracováním nastartujeme naši zakódovací proceduru zakódováním dvou slov z S_{N-2} s dvěma kódovými slovy 0 a 1; pak S_{N-3} bude mít tři kódová slova atd. a budeme pokračovat ve výše uvedené proceduře, až dosáhneme Huffmanova kódu pro $S = S_0$.

Příklad 5.4

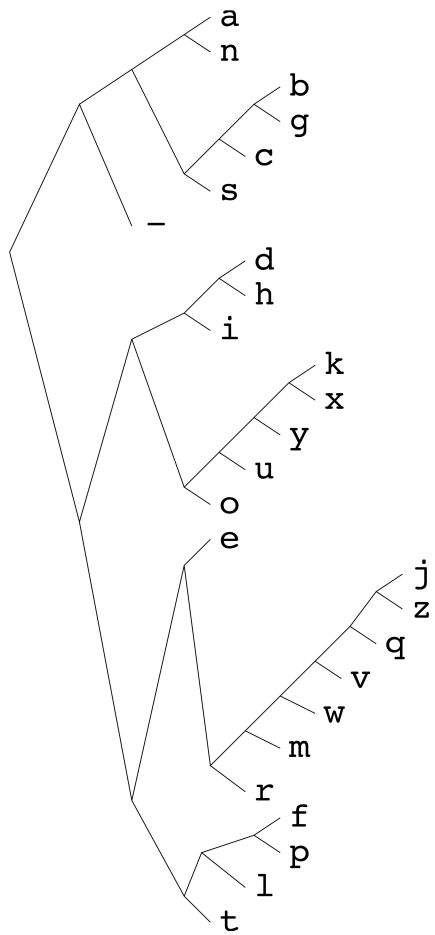
V následující tabulce je uvedeno Huffmanovo kódování francouzského a anglického jazyka. Frekvence udává, kolikrát se písmeno objevilo ve vzorku 1000 slov.
Další tabulka ukazuje Huffmanovo kódování anglického jazyka (včetně mezery) založené na pravděpodobnostech z jiného korpusu.

HUFFMANŮV KÓDOVACÍ ALGORITMUS

$Char$	$F_{F_{st}}$	$Code_{F_{st}}$	$F_{E_{st}}$	$Code_{E_{st}}$
E	850	00	591	100
A	395	1110	368	1111
S	388	1101	275	0110
I	366	1100	286	0111
T	356	1011	473	001
N	355	1010	320	1100
R	305	1001	308	1011
U	295	0111	111	00010
L	278	0101	153	10101
O	255	0100	360	1110
D	200	11110	171	11011
C	148	10000	124	01010
M	145	01101	114	00011
P	144	01100	89	00000
V	75	100010	41	1101001
Q	61	1111110	5	1101000101
G	51	1111101	90	00001
F	46	1111100	132	01011
B	42	1000111	65	101000
H	35	11111111	237	0100
J	29	11111110	6	1101000110
X	17	10001100	7	1101000111
Y	10	100011010	89	110101
Z	8	1000110111	3	1101000100
K	2	10001101101	19	11010000
W	1	10001101100	68	101001

HUFFMANŮV KÓDOVACÍ ALGORITMUS

a_i	p_i	$\log_2 \frac{1}{p_i}$	l_i	$c(a_i)$
a	0.0575	4.1	4	0000
b	0.0128	6.3	6	001000
c	0.0263	5.2	5	00101
d	0.0285	5.1	5	10000
e	0.0913	3.5	4	1100
f	0.0173	5.9	6	111000
g	0.0133	6.2	6	001001
h	0.0313	5.0	5	10001
i	0.0599	4.1	4	1001
j	0.0006	10.7	10	1101000000
k	0.0084	6.9	7	1010000
l	0.0335	4.9	5	11101
m	0.0235	5.4	6	110101
n	0.0596	4.1	4	0001
o	0.0689	3.9	4	1011
p	0.0192	5.7	6	111001
q	0.0008	10.3	9	110100001
r	0.0508	4.3	5	11011
s	0.0567	4.1	4	0011
t	0.0706	3.8	4	1111
u	0.0334	4.9	5	10101
v	0.0069	7.2	8	11010001
w	0.0119	6.4	7	1101001
x	0.0073	7.1	7	1010001
y	0.0164	5.9	6	101001
z	0.0007	10.4	10	1101000001
–	0.1928	2.4	2	01



HUFFMANŮV KÓDOVACÍ ALGORITMUS

Budeme ilustrovat Huffmanovu metodu na skutečně malém příkladě.

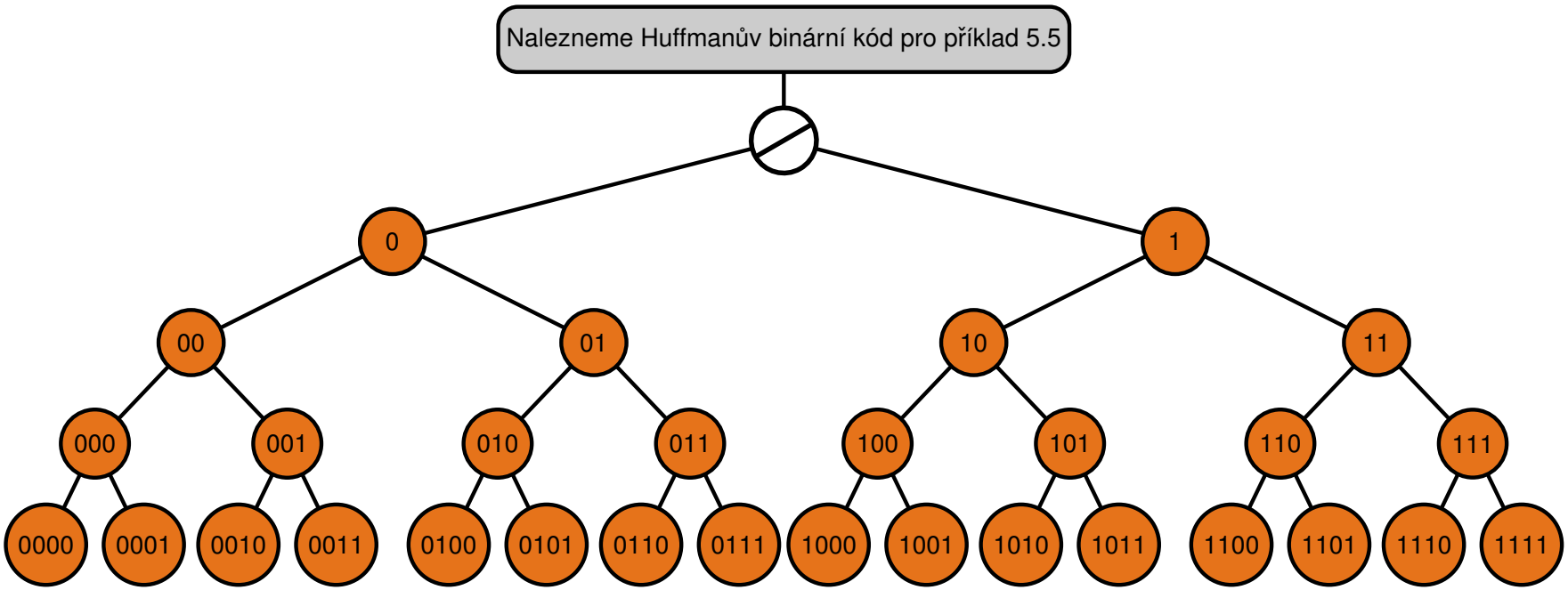
Příklad 5.5

Předpokládejme, že $\mathcal{S}$ je zdroj s pěti zdrojovými slovy a pravděpodobnostmi (viz níže). Pak vývoj Huffmanova zakódování lze považovat za procházení šipek dopředu a pak zakódování zpátky.

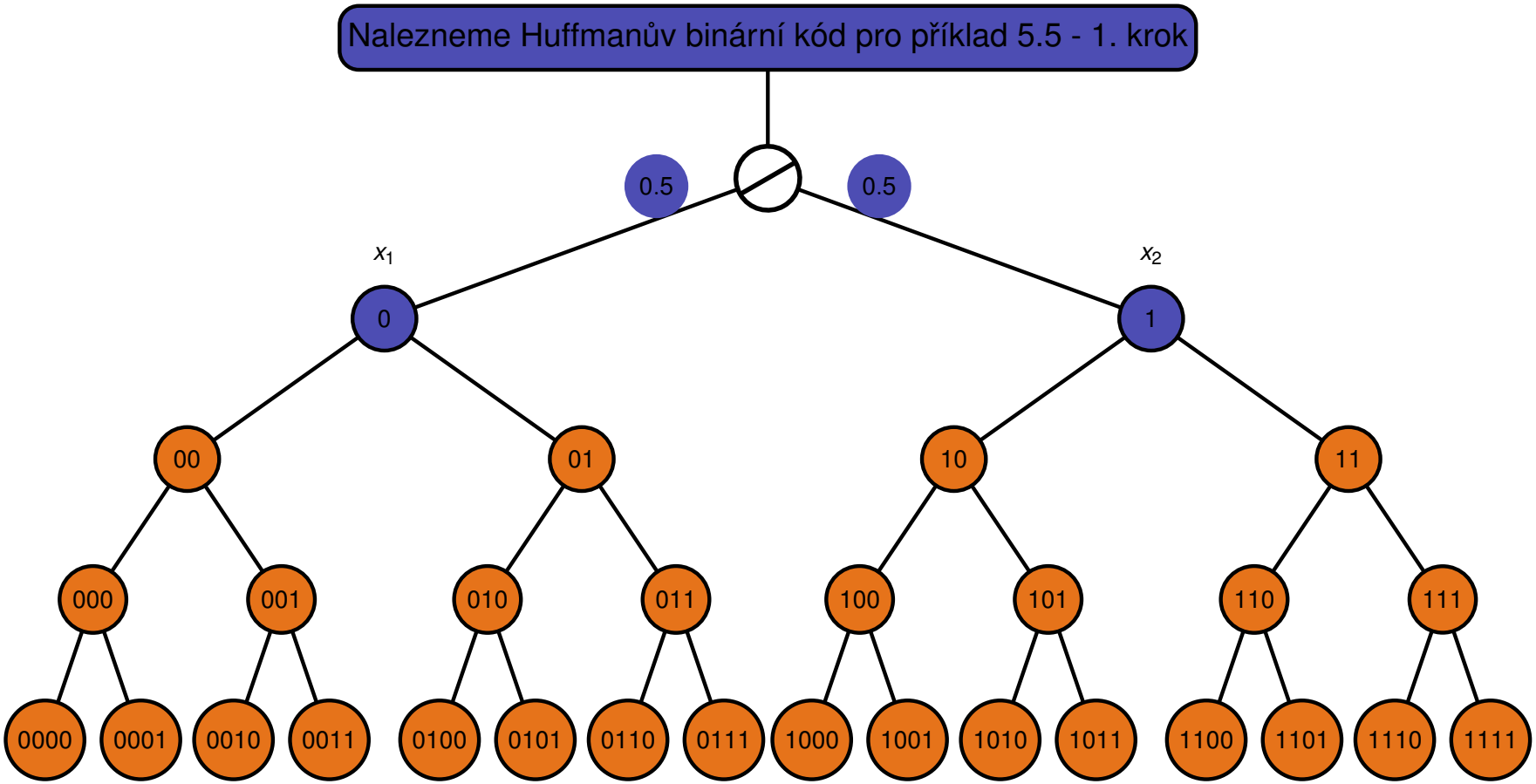
W	P	C	W	P	C	W	P	C	W	P	C
w_1	0.5	1	v_1	0.5	1	u_1	0.5	1	x_1	0.5	0
w_2	0.2	01	v_2	0.2	01	u_2	0.3	00	x_2	0.5	1
w_3	0.15	001	v_3	0.15	000	u_3	0.2	01			
w_4	0.1	0000	v_4	0.15	001						
w_5	0.05	0001									

W: slovo, P: pravděpodobnost C: kódování

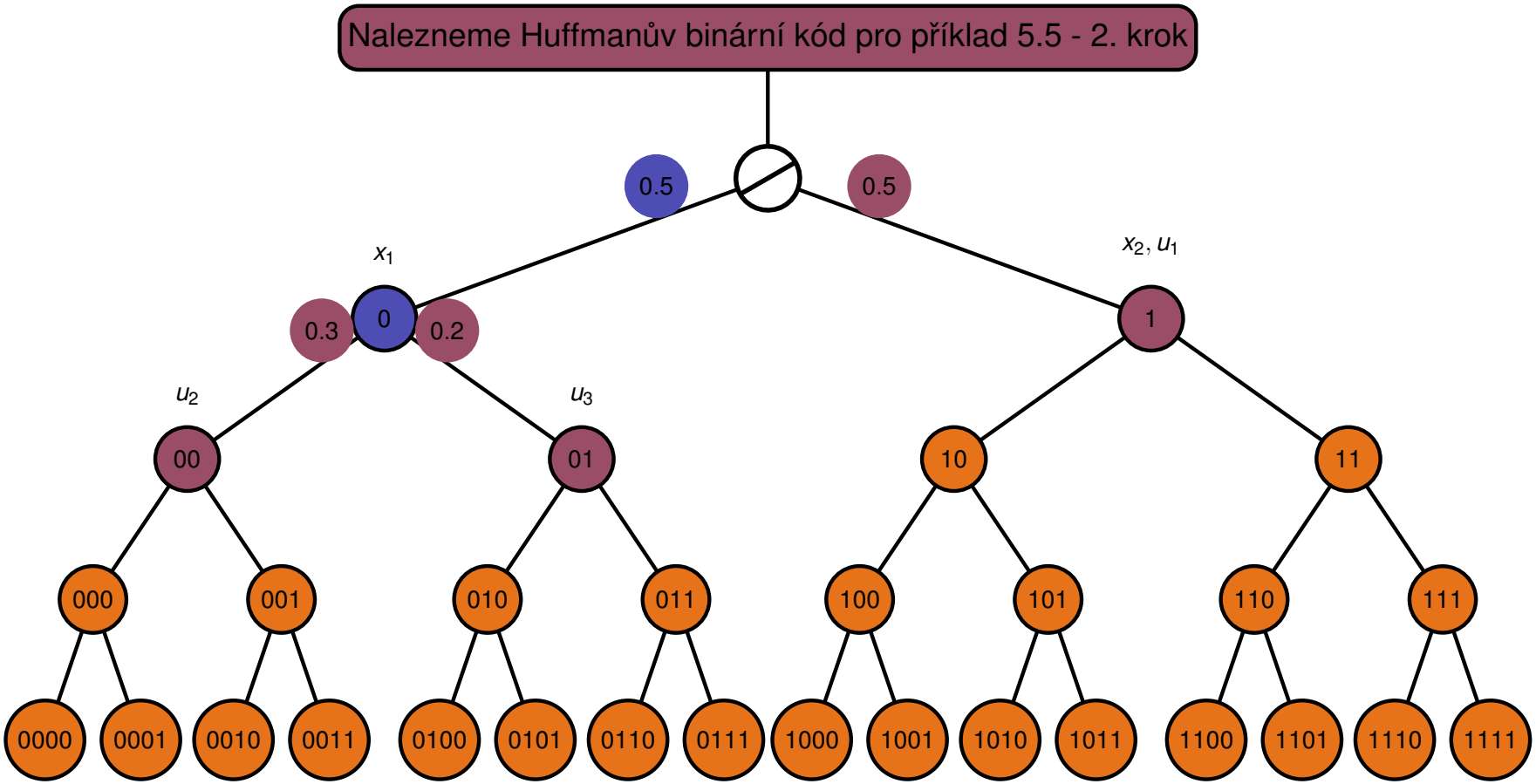
HUFFMANŮV KÓDOVACÍ ALGORITMUS



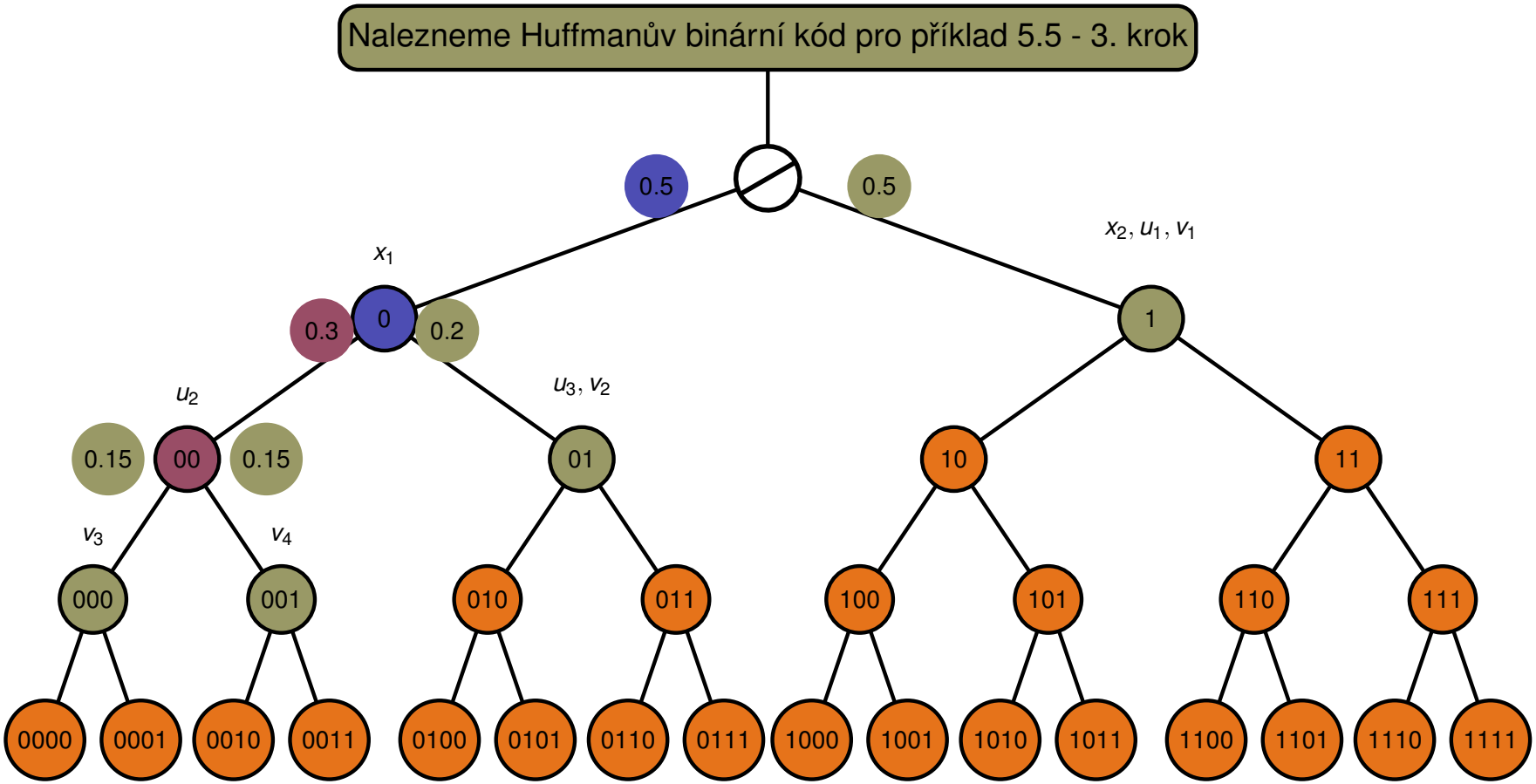
HUFFMANŮV KÓDOVACÍ ALGORITMUS



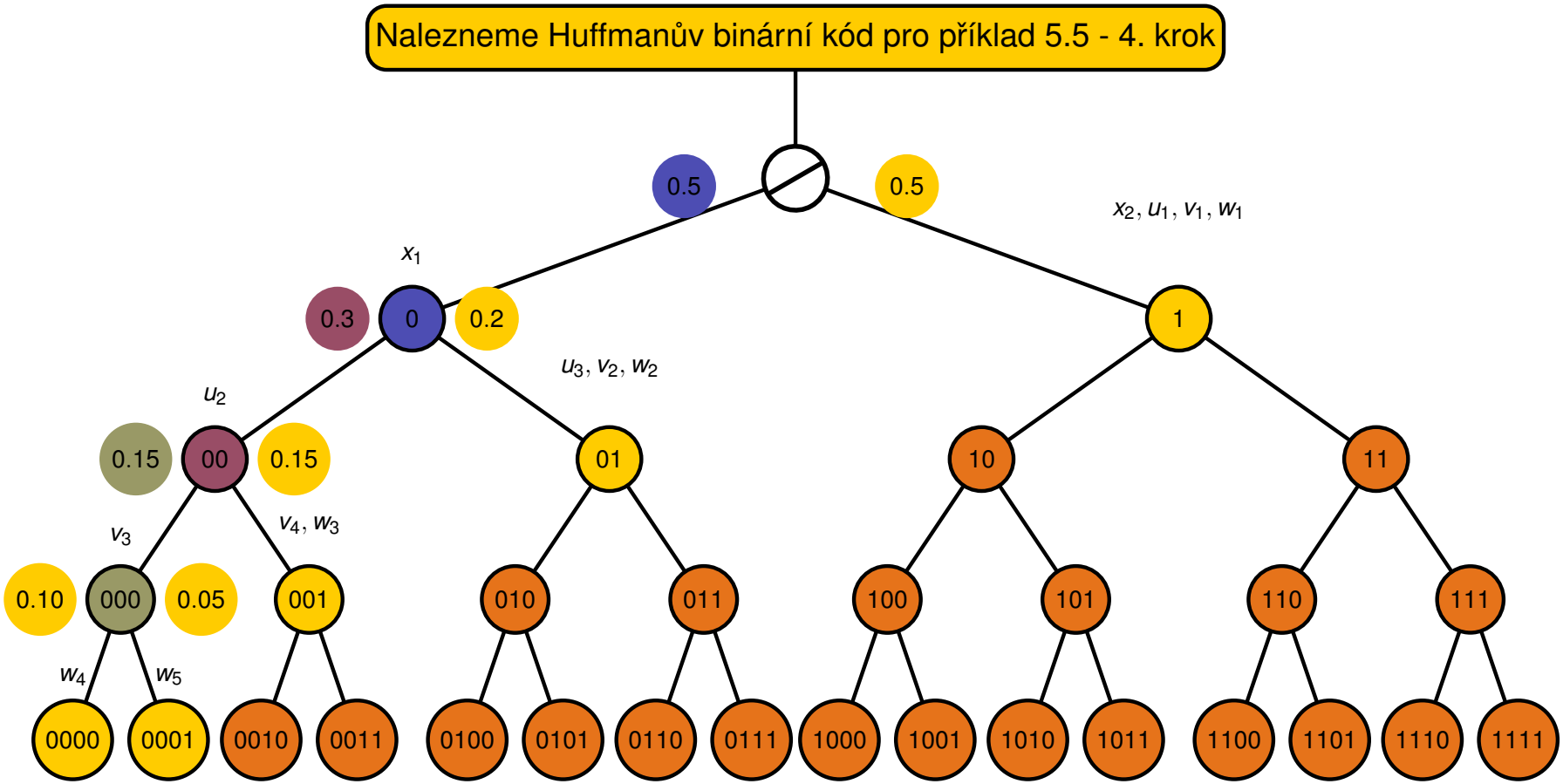
HUFFMANŮV KÓDOVACÍ ALGORITMUS



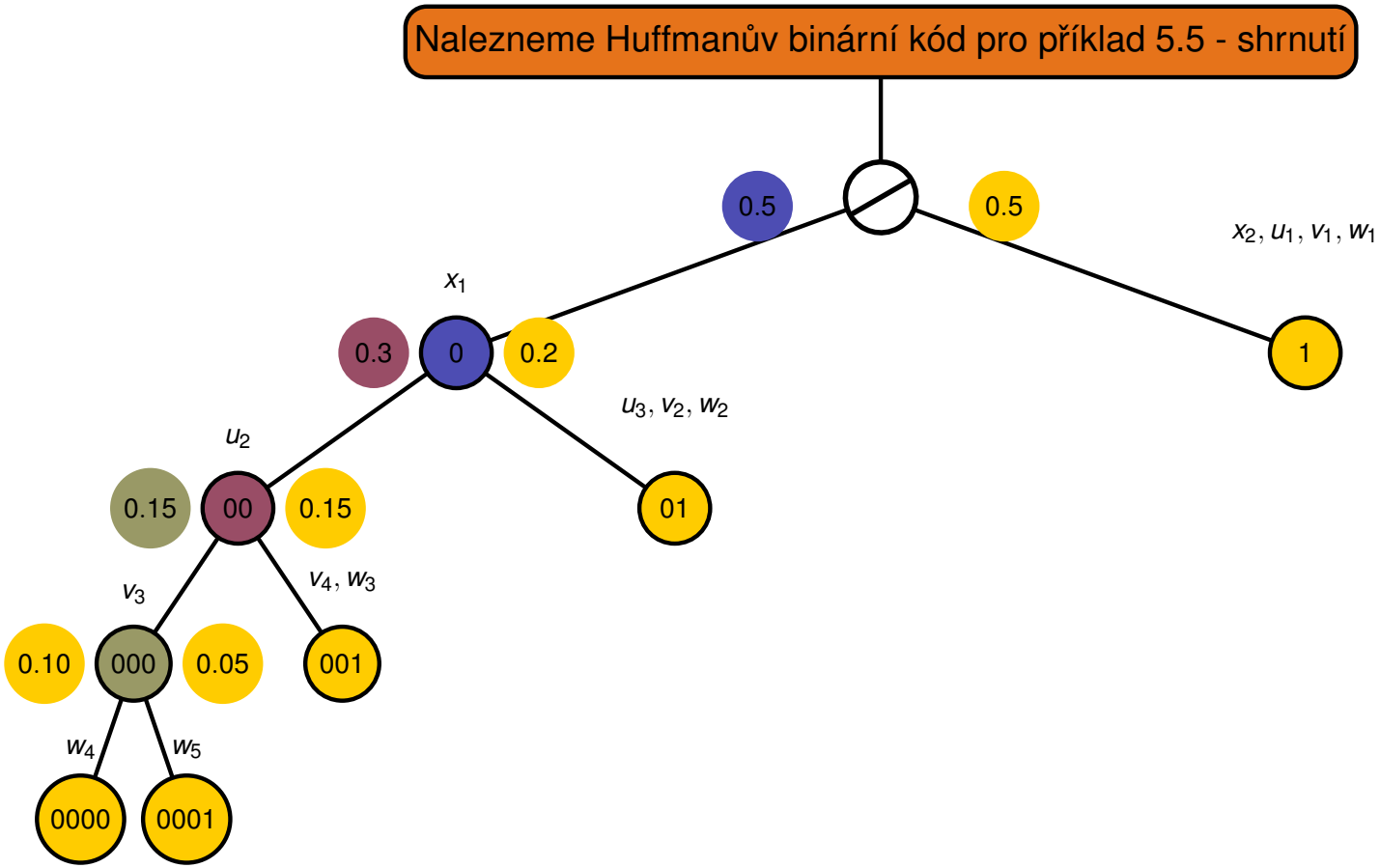
HUFFMANŮV KÓDOVACÍ ALGORITMUS



HUFFMANŮV KÓDOVACÍ ALGORITMUS



HUFFMANŮV KÓDOVACÍ ALGORITMUS



HUFFMANŮV KÓDOVACÍ ALGORITMUS

Příklad 5.5 (Pokračování)

Výsledné zakódování

$$w_1 \to 1, w_2 \to 01, w_3 \to 001, w_4 \to 0000, w_5 \to 0001$$

má průměrnou délku 1.95 na zdrojové slovo.

HUFFMANŮV KÓDOVACÍ ALGORITMUS

Příklad 5.5 (Pokračování)

Výsledné zakódování

$$w_1 \to 1, w_2 \to 01, w_3 \to 001, w_4 \to 0000, w_5 \to 0001$$

má průměrnou délku 1.95 na zdrojové slovo.

Poznámka 5.6

Minimálně dvakrát v horním příkladu jsme schopni provést výběr, protože dvě slova mají stejné pravděpodobnosti. Pokud tento případ nastane, dostaneme různá zakódování.

DŮKAZ, ŽE HUFFMANŮV ALGORITMUS JE KOREKTNÍ

Z Lemmatu 5.1 víme, že zakódování S_{N-2} dvěma symboly 0 a 1 je optimální. Důkaz bude tedy úplný, jestliže budeme schopni dokázat, že kompaktnost je zachovávána při přechodu ze zdroje S_j ke zdroji S_{j-1}.

DŮKAZ, ŽE HUFFMANŮV ALGORITMUS JE KOREKTNÍ

Z Lemmatu 5.1 víme, že zakódování S_{N-2} dvěma symboly 0 a 1 je optimální. Důkaz bude tedy úplný, jestliže budeme schopni dokázat, že kompaktnost je zachovávána při přechodu ze zdroje S_j ke zdroji S_{j-1}.

Předpokládejme tedy, že S_j je kompaktně zakódován a že $l_1, \ldots, l_t$ jsou délky slov $\sigma_1, \ldots, \sigma_t$, ale že Huffmanovo zakódování C_{j-1} zdroje S_{j-1} není kompaktní.

DŮKAZ, ŽE HUFFMANŮV ALGORITMUS JE KOREKTNÍ

Z Lemmatu 5.1 víme, že zakódování S_{N-2} dvěma symboly 0 a 1 je optimální. Důkaz bude tedy úplný, jestliže budeme schopni dokázat, že kompaktnost je zachovávána při přechodu ze zdroje S_j ke zdroji S_{j-1}.

Předpokládejme tedy, že S_j je kompaktně zakódován a že $l_1, \ldots, l_t$ jsou délky slov $\sigma_1, \ldots, \sigma_t$, ale že Huffmanovo zakódování C_{j-1} zdroje S_{j-1} není kompaktní.

Existuje tedy prefixové kompaktní kódování C zdroje S_{j-1} tak, že

$$l(C) < l(C_{j-1}).$$

DŮKAZ, ŽE HUFFMANŮV ALGORITMUS JE KOREKTNÍ

Dle lemmatu 5.1 můžeme přeuspořádat kódová slova kódování C maximální délky tak, že má-li C kódová slova $v'_1, \ldots, v'_{t+1}$ s délkami $l'_1, \ldots, l'_{t+1}$, pak $l'_1 \leq l'_2 \cdots \leq l'_{t+1}$ a $v'_t = (\sigma, 0)$, $v'_{t+1} = (\sigma, 1)$, kde σ je nějaké kódové slovo z Σ^*.

DŮKAZ, ŽE HUFFMANŮV ALGORITMUS JE KOREKTNÍ

Dle lemmatu 5.1 můžeme přeuspořádat kódová slova kódování C maximální délky tak, že má-li C kódová slova $v'_1, \ldots, v'_{t+1}$ s délkami $l'_1, \ldots, l'_{t+1}$, pak $l'_1 \leq l'_2 \cdots \leq l'_{t+1}$ a $v'_t = (\sigma, 0)$, $v'_{t+1} = (\sigma, 1)$, kde σ je nějaké kódové slovo z Σ^*.

Nechť kódování C' zdroje S_j sestává ze slov

$$v'_1, v'_2, \ldots, v'_{k-1}, \sigma, v'_k, \ldots, v'_{t-1};$$

pak C' je prefixové zakódování zdroje S_j a má průměrnou délku

$$\begin{aligned} l(C') &= q_1 l'_1 + \cdots + q_{k-1} l'_{k-1} + (q_t + q_{t+1})|\sigma| + q_k l'_k \\ &\quad + q_{k+1} l'_{k+1} + \cdots + q_{t-1} l'_{t-1} \\ &= l(C) - (q_t + q_{t+1}). \end{aligned}$$

DŮKAZ, ŽE HUFFMANŮV ALGORITMUS JE KOREKTNÍ

Ale zároveň

$$l(C_j) = l(C_{j-1}) - (q_t + q_{t+1}).$$

DŮKAZ, ŽE HUFFMANŮV ALGORITMUS JE KOREKTNÍ

Ale zároveň

$$l(C_j) = l(C_{j-1}) - (q_t + q_{t+1}).$$

Tudíž, jestliže $l(C) < l(C_{j-1})$, pak

$$l(C') < l(C_j),$$

což je spor s kompaktností C_j.

HUFFMANOVA KÓDOVÁNÍ NAD NEBINÁRNÍMI ABECEDAMI

Předpokládejme, že pracujeme namísto s binární abecedou $\{0, 1\}$ s abecedou Σ o r symbolech.

HUFFMANOVA KÓDOVÁNÍ NAD NEBINÁRNÍMI ABECEDAMI

Předpokládejme, že pracujeme namísto s binární abecedou $\{0, 1\}$ s abecedou Σ o r symbolech.

Budeme postupovat stejným způsobem. Stejně jako v binárním případě, začněme s $S = S_0$ (původní zdroj) a budeme konstruovat posloupnost zdrojů $S_0, S_1, \ldots, S_t$ až skončíme u zdroje S_t obsahujícím právě r symbolů.

HUFFMANOVA KÓDOVÁNÍ NAD NEBINÁRNÍMI ABECEDAMI

Předpokládejme, že pracujeme namísto s binární abecedou $\{0, 1\}$ s abecedou Σ o r symbolech.

Budeme postupovat stejným způsobem. Stejně jako v binárním případě, začněme s $S = S_0$ (původní zdroj) a budeme konstruovat posloupnost zdrojů $S_0, S_1, \ldots, S_t$ až skončíme u zdroje S_t obsahujícím právě r symbolů.

Tento zdroj má kompaktní zakódování (jednoduše máme bijekci mezi S_t a abecedou Σ).

HUFFMANOVA KÓDOVÁNÍ NAD NEBINÁRNÍMI ABECEDAMI

Musíme aplikovat následující dva body:

1. Když budeme konstruovat S_{j+1} z S_j, ztotožníme ne 2, ale r nejméně pravděpodobných symbolů z S_j do jednoho symbolu z S_{j+1}. Tedy S_{j+1} má o $r-1$ symbolů méně než S_j.
2. Budeme potřebovat pro závěrečný zdroj S_t, abychom měli právě r symbolů. Abychom toho dosáhli, je nutno začít se zdrojem $\mathcal{S}$ s právě $r + t(r-1)$ symboly. Protože obecně je málo pravděpodobné, že $\mathcal{S}$ bude mít právě tento počet slov, uměle přidáme k $\mathcal{S}$ množinu S', která bude disjunktní s $\mathcal{S}$, a slova v ní obsažená budou mít nulovou pravděpodobnost. Pak klademe $S_0 = S \cup S'$ a máme $|S'| = r + t(r-1) - |S|$.

Cvičení

Cvičení 5.7

1. *Jaká je průměrná délka slova kompaktního kódování nad* $\{0, 1\}$*, jestliže máme 5 stejně pravděpodobných zdrojových slov?*

Cvičení

Cvičení 5.7

1. *Jaká je průměrná délka slova kompaktního kódování nad* $\{0,1\}$*, jestliže máme 5 stejně pravděpodobných zdrojových slov?*
2. *Najděte kompaktní kódování nad* $\{0,1\}$ *pro zdroj vysílající slova* $w_1, \ldots, w_6$ *s pravděpodobnostmi*

$$P(w_1) = \frac{1}{3}, P(w_2) = \frac{1}{4}, P(w_3) = \frac{1}{6}, P(w_4) = P(w_5) = P(w_6) = \frac{1}{12}$$

a porovnejte jejich průměrnou délku s horní a dolní hranicí dle věty o kódování bez šumu pro zdroje bez paměti.

Cvičení

Cvičení 5.7

1. *Jaká je průměrná délka slova kompaktního kódování nad* $\{0,1\}$*, jestliže máme 5 stejně pravděpodobných zdrojových slov?*
2. *Najděte kompaktní kódování nad* $\{0,1\}$ *pro zdroj vysílající slova* $w_1, \ldots, w_6$ *s pravděpodobnostmi*

$$P(w_1) = \frac{1}{3}, P(w_2) = \frac{1}{4}, P(w_3) = \frac{1}{6}, P(w_4) = P(w_5) = P(w_6) = \frac{1}{12}$$

   a porovnejte jejich průměrnou délku s horní a dolní hranicí dle věty o kódování bez šumu pro zdroje bez paměti.
3. *Najděte kompaktní kódování nad* $\{0,1,2\}$ *pro zdroj z předchozího příkladu.*

Obsah

Generování diskrétních rozdělení vrhem ideální mince

FI Uvažme následující otázku.

Generování diskrétních rozdělení vrhem ideální mince

FI Uvažme následující otázku.

Kolik vrhů ideální mince je potřeba, abychom vygenerovali náhodnou proměnnou X vzhledem k předem zadanému pravděpodobnostnímu rozdělení?

Generování diskrétních rozdělení vrhem ideální mince

FI Uvažme následující otázku.

Kolik vrhů ideální mince je potřeba, abychom vygenerovali náhodnou proměnnou X vzhledem k předem zadanému pravděpodobnostnímu rozdělení?

Zkusme si to nejprve ukázat na jednoduchém příkladě.

Generování diskrétních rozdělení vrhem ideální mince

Příklad 6.1

Buď dána posloupnost vrhů ideální mincí (tj. ideální bity). Dále předpokládejme, že si přejeme vygenerovat náhodnou proměnnou X s rozdělením

$$X = \begin{cases} a & \text{s pravděpodobností } \frac{1}{2}, \\ b & \text{s pravděpodobností } \frac{1}{4}, \\ c & \text{s pravděpodobností } \frac{1}{4}. \end{cases} \tag{6.1}$$

Generování diskrétních rozdělení vrhem ideální mince

Příklad 6.1

Buď dána posloupnost vrhů ideální mincí (tj. ideální bity). Dále předpokládejme, že si přejeme vygenerovat náhodnou proměnnou X s rozdělením

$$X = \begin{cases} a & \text{s pravděpodobností } \frac{1}{2}, \\ b & \text{s pravděpodobností } \frac{1}{4}, \\ c & \text{s pravděpodobností } \frac{1}{4}. \end{cases} \tag{6.1}$$

Není žádný problém uhodnout odpověď. Je-li první bit 0, položíme $X = a$. Jsou-li první dva bity 10, položíme $X = b$. Pokud vidíme 11, položíme $X = c$. Je jasné, že X má požadované rozdělení.

Generování diskrétních rozdělení vrhem ideální mince

Příklad 6.1

Buď dána posloupnost vrhů ideální mincí (tj. ideální bity). Dále předpokládejme, že si přejeme vygenerovat náhodnou proměnnou X s rozdělením

$$X = \begin{cases} a & \text{s pravděpodobností } \frac{1}{2}, \\ b & \text{s pravděpodobností } \frac{1}{4}, \\ c & \text{s pravděpodobností } \frac{1}{4}. \end{cases} \tag{6.1}$$

Není žádný problém uhodnout odpověď. Je-li první bit 0, položíme $X = a$. Jsou-li první dva bity 10, položíme $X = b$. Pokud vidíme 11, položíme $X = c$. Je jasné, že X má požadované rozdělení.
Určeme dále průměrný počet ideálních bitů potřebných pro vygenerování této náhodné proměnné. V tomto případě to je $\frac{1}{2}(1) + \frac{1}{4}(2) + \frac{1}{4}(2) = 1,5$ bitů.

Generování diskrétních rozdělení vrhem ideální mince

Obecný problém můžeme pak formulovat následovně. Mějme dánu posloupnost vrhů ideální mincí $Z_1, Z_2, \ldots,$.

Generování diskrétních rozdělení vrhem ideální mince

Obecný problém můžeme pak formulovat následovně. Mějme dánu posloupnost vrhů ideální mincí $Z_1, Z_2, \ldots,$.

Přejeme si vygenerovat diskrétní náhodnou proměnnou X s oborem hodnot $\{a_1, a_2, \ldots, a_m\}$ s pravděpodobnostním rozdělením $p_1, p_2, \ldots, p_m$.

Generování diskrétních rozdělení vrhem ideální mince

Obecný problém můžeme pak formulovat následovně. Mějme dánu posloupnost vrhů ideální mincí $Z_1, Z_2, \ldots,$.

Přejeme si vygenerovat diskrétní náhodnou proměnnou X s oborem hodnot $\{a_1, a_2, \ldots, a_m\}$ s pravděpodobnostním rozdělením $p_1, p_2, \ldots, p_m$.

Důležité!!! X má konečný obor hodnot !!!

Generování diskrétních rozdělení vrhem ideální mince

Obecný problém můžeme pak formulovat následovně. Mějme dánu posloupnost vrhů ideální mincí $Z_1, Z_2, \ldots,$.

Přejeme si vygenerovat diskrétní náhodnou proměnnou X s oborem hodnot $\{a_1, a_2, \ldots, a_m\}$ s pravděpodobnostním rozdělením $p_1, p_2, \ldots, p_m$.

Důležité!!! X má konečný obor hodnot !!!

Nechť náhodná proměnná T označuje počet vrhů mince použitých v algoritmu.

Generování diskrétních rozdělení vrhem ideální mince

Obecný problém můžeme pak formulovat následovně. Mějme dánu posloupnost vrhů ideální mincí $Z_1, Z_2, \ldots,$.

Přejeme si vygenerovat diskrétní náhodnou proměnnou X s oborem hodnot $\{a_1, a_2, \ldots, a_m\}$ s pravděpodobnostním rozdělením $p_1, p_2, \ldots, p_m$.

Důležité!!! X má konečný obor hodnot !!!

Nechť náhodná proměnná T označuje počet vrhů mince použitých v algoritmu.

Můžeme popsat algoritmus zobrazující řetězce bitů $Z_1, Z_2, \ldots,$ na možné výstupy náhodné proměnné X pomocí binárního stromu.

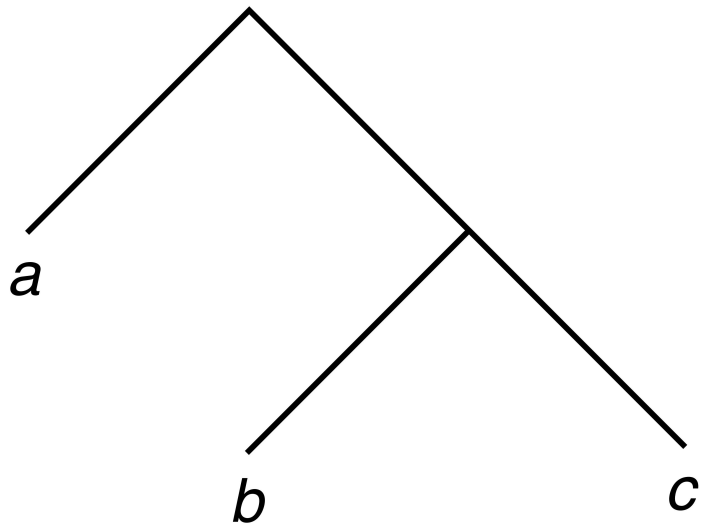
Generování diskrétních rozdělení vrhem ideální mince

Listy stromu jsou označeny výstupními symboly X a cesta k listům je zadána posloupností bitů generovanou vrhem ideální mincí.

Generování diskrétních rozdělení vrhem ideální mince

Listy stromu jsou označeny výstupními symboly X a cesta k listům je zadána posloupností bitů generovanou vrhem ideální mincí.

Například strom pro generování pravděpodobnostního rozdělení $\frac{1}{2}, \frac{1}{4}, \frac{1}{4}$ z příkladu 6.1 je uveden na sousedním obrázku.



Strom reprezentující algoritmus musí splňovat jisté vlastnosti:

Generování diskrétních rozdělení vrhem ideální mince

1. Strom musí být úplný, tj. každý uzel je buď list nebo má dva následníky ve stromu. Strom může být nekonečný.

Generování diskrétních rozdělení vrhem ideální mince

1. Strom musí být úplný, tj. každý uzel je buď list nebo má dva následníky ve stromu. Strom může být nekonečný.
2. Pravděpodobnost listu hloubky k je 2^{-k}. Mnoho listů může být označeno stejným výstupním symbolem - celková pravděpodobnost všech těchto listů je pak rovna požadované pravděpodobnosti výstupního symbolu.

Generování diskrétních rozdělení vrhem ideální mince

1. Strom musí být úplný, tj. každý uzel je buď list nebo má dva následníky ve stromu. Strom může být nekonečný.
2. Pravděpodobnost listu hloubky k je 2^{-k}. Mnoho listů může být označeno stejným výstupním symbolem - celková pravděpodobnost všech těchto listů je pak rovna požadované pravděpodobnosti výstupního symbolu.
3. Průměrný počet ideálních bitů ET požadovaných na vygenerování X je roven střední hodnotě hloubky tohoto stromu.

Generování diskrétních rozdělení vrhem ideální mince

1. Strom musí být úplný, tj. každý uzel je buď list nebo má dva následníky ve stromu. Strom může být nekonečný.
2. Pravděpodobnost listu hloubky k je 2^{-k}. Mnoho listů může být označeno stejným výstupním symbolem - celková pravděpodobnost všech těchto listů je pak rovna požadované pravděpodobnosti výstupního symbolu.
3. Průměrný počet ideálních bitů ET požadovaných na vygenerování X je roven střední hodnotě hloubky tohoto stromu.

Generování diskrétních rozdělení vrhem ideální mince

1. Strom musí být úplný, tj. každý uzel je buď list nebo má dva následníky ve stromu. Strom může být nekonečný.
2. Pravděpodobnost listu hloubky k je 2^{-k}. Mnoho listů může být označeno stejným výstupním symbolem - celková pravděpodobnost všech těchto listů je pak rovna požadované pravděpodobnosti výstupního symbolu.
3. Průměrný počet ideálních bitů ET požadovaných na vygenerování X je roven střední hodnotě hloubky tohoto stromu.

Samozřejmě máme mnoho možných algoritmů, které generují stejné rozdělení pravděpodobnosti.

Generování diskrétních rozdělení vrhem ideální mince

Například, zobrazení $00 \mapsto a$, $01 \mapsto b$, $10 \mapsto c$, $11 \mapsto a$ nám rovněž poskytne rozdělení $\frac{1}{2}, \frac{1}{4}, \frac{1}{4}$ z příkladu 6.1.

Generování diskrétních rozdělení vrhem ideální mince

Například, zobrazení $00 \mapsto a$, $01 \mapsto b$, $10 \mapsto c$, $11 \mapsto a$ nám rovněž poskytne rozdělení $\frac{1}{2}, \frac{1}{4}, \frac{1}{4}$ z příkladu 6.1.

Ale průměrný počet ideálních bitů potřebných pro vygenerování této náhodné proměnné je $\frac{1}{4}(2) + \frac{1}{4}(2) + \frac{1}{4}(2) + \frac{1}{4}(2) = 2$ bity.

Generování diskrétních rozdělení vrhem ideální mince

Například, zobrazení $00 \mapsto a$, $01 \mapsto b$, $10 \mapsto c$, $11 \mapsto a$ nám rovněž poskytne rozdělení $\frac{1}{2}, \frac{1}{4}, \frac{1}{4}$ z příkladu 6.1.

Ale průměrný počet ideálních bitů potřebných pro vygenerování této náhodné proměnné je $\frac{1}{4}(2) + \frac{1}{4}(2) + \frac{1}{4}(2) + \frac{1}{4}(2) = 2$ bity.

Není tedy tak efektivní jako přiřazení z příkladu 6.1, kde jsme potřebovali pouze $1,5$ bitů.

Generování diskrétních rozdělení vrhem ideální mince

Například, zobrazení $00 \mapsto a$, $01 \mapsto b$, $10 \mapsto c$, $11 \mapsto a$ nám rovněž poskytne rozdělení $\frac{1}{2}, \frac{1}{4}, \frac{1}{4}$ z příkladu 6.1.

Ale průměrný počet ideálních bitů potřebných pro vygenerování této náhodné proměnné je $\frac{1}{4}(2) + \frac{1}{4}(2) + \frac{1}{4}(2) + \frac{1}{4}(2) = 2$ bity.

Není tedy tak efektivní jako přiřazení z příkladu 6.1, kde jsme potřebovali pouze $1,5$ bitů.

Můžeme tedy očekávat, že budeme potřebovat alespoň tolik nahodilosti v ideálních bitech, kolik vytvoříme ve výstupních symbolech.

Generování diskrétních rozdělení vrhem ideální mince

Například, zobrazení $00 \mapsto a$, $01 \mapsto b$, $10 \mapsto c$, $11 \mapsto a$ nám rovněž poskytne rozdělení $\frac{1}{2}, \frac{1}{4}, \frac{1}{4}$ z příkladu 6.1.

Ale průměrný počet ideálních bitů potřebných pro vygenerování této náhodné proměnné je $\frac{1}{4}(2) + \frac{1}{4}(2) + \frac{1}{4}(2) + \frac{1}{4}(2) = 2$ bity.

Není tedy tak efektivní jako přiřazení z příkladu 6.1, kde jsme potřebovali pouze $1,5$ bitů.

Můžeme tedy očekávat, že budeme potřebovat alespoň tolik nahodilosti v ideálních bitech, kolik vytvoříme ve výstupních symbolech.

Protože entropie je míra nahodilosti a každý ideální bit má entropii jednoho bitu, můžeme čekat, že počet použitých ideálních bitů bude alespoň tolik jako entropie výstupu.

Generování diskrétních rozdělení vrhem ideální mince

Pro další úvahy označme $\mathcal{Y}$ množinu listů úplného stromu. Dále uvažujme pravděpodobnostní rozdělení na listech tak, že pravděpodobnost listu hloubky k stromu bude 2^{-k}. Buď dále Y náhodná proměnná s touto distribucí. Máme pak následující lemma.

Generování diskrétních rozdělení vrhem ideální mince

Pro další úvahy označme $\mathcal{Y}$ množinu listů úplného stromu. Dále uvažujme pravděpodobnostní rozdělení na listech tak, že pravděpodobnost listu hloubky k stromu bude 2^{-k}. Buď dále Y náhodná proměnná s touto distribucí. Máme pak následující lemma.

Lemma 6.2

Pro každý úplný strom uvažujeme pravděpodobnostní rozdělení na listech tak, že pravděpodobnost listu hloubky k stromu bude 2^{-k}. Pak střední hodnota hloubky stromu je rovna entropii tohoto rozdělení.

Důkaz Lemmatu 6.2

Lemma 6.2

Pro každý úplný strom uvažujeme pravděpodobnostní rozdělení na listech tak, že pravděpodobnost listu hloubky k stromu bude 2^{-k}. Pak střední hodnota hloubky stromu je rovna entropii tohoto rozdělení.

MA Střední hodnota hloubky stromu je rovna

$$ET = \sum_{y \in \mathcal{Y}} k(y) 2^{-k(y)} \tag{6.2}$$

a entropie rozdělení náhodné proměnné Y je

$$H(Y) = -\sum_{y \in \mathcal{Y}} \frac{1}{2^{k(y)}} \log_2 \frac{1}{2^{k(y)}} = \sum_{y \in \mathcal{Y}} k(y) 2^{-k(y)}, \tag{6.3}$$

kde $k(y)$ značí hloubku listu $y \in \mathcal{Y}$. Tedy $H(Y) = ET$.

Střední hodnota hloubky stromu

Věta 6.3

FI *Pro každý algoritmus generující náhodnou proměnnou X je střední hodnota počtu ideálních bitů alespoň rovna entropii $H(X)$, tj.*

$$ET \geq H(X). \tag{6.4}$$

Střední hodnota hloubky stromu

Věta 6.3

FI *Pro každý algoritmus generující náhodnou proměnnou X je střední hodnota počtu ideálních bitů alespoň rovna entropii $H(X)$, tj.*

$$ET \geq H(X). \tag{6.4}$$

MA Každý algoritmus generující náhodnou proměnnou X pomocí ideálních bitů můžeme reprezentovat pomocí úplného binárního stromu.

Střední hodnota hloubky stromu

Věta 6.3

FI *Pro každý algoritmus generující náhodnou proměnnou X je střední hodnota počtu ideálních bitů alespoň rovna entropii $H(X)$, tj.*

$$ET \geq H(X). \tag{6.4}$$

MA Každý algoritmus generující náhodnou proměnnou X pomocí ideálních bitů můžeme reprezentovat pomocí úplného binárního stromu.

Označme všechny listy tohoto stromu různými symboly $y \in \mathcal{Y} = \{1, 2, \dots\}$.

Střední hodnota hloubky stromu

Věta 6.3

FI *Pro každý algoritmus generující náhodnou proměnnou X je střední hodnota počtu ideálních bitů alespoň rovna entropii $H(X)$, tj.*

$$ET \geq H(X). \tag{6.4}$$

MA Každý algoritmus generující náhodnou proměnnou X pomocí ideálních bitů můžeme reprezentovat pomocí úplného binárního stromu.

Označme všechny listy tohoto stromu různými symboly $y \in \mathcal{Y} = \{1, 2, \dots\}$.

Je-li strom nekonečný, pak je i abeceda $\mathcal{Y}$ nekonečná.

Střední hodnota hloubky stromu

Zde je nutno být opatrný - zatím jsme pracovali pouze s náhodnými proměnnými s konečným oborem hodnot, ale snadno nahlédneme, že v tomto případě všechny úvahy zůstanou stejné!!!

Střední hodnota hloubky stromu

Zde je nutno být opatrný - zatím jsme pracovali pouze s náhodnými proměnnými s konečným oborem hodnot, ale snadno nahlédneme, že v tomto případě všechny úvahy zůstanou stejné!!!

Uvažme nyní náhodnou proměnnou Y definovanou na listech tohoto stromu tak, že pro každý list $y \in \mathcal{Y}$ hloubky $k(y)$ je pravděpodobnost $P(Y = y) = 2^{-k(y)}$.

Střední hodnota hloubky stromu

Zde je nutno být opatrný - zatím jsme pracovali pouze s náhodnými proměnnými s konečným oborem hodnot, ale snadno nahlédneme, že v tomto případě všechny úvahy zůstanou stejné!!!

Uvažme nyní náhodnou proměnnou Y definovanou na listech tohoto stromu tak, že pro každý list $y \in \mathcal{Y}$ hloubky $k(y)$ je pravděpodobnost $P(Y = y) = 2^{-k(y)}$.

Dle Lemmatu 6.2 je střední hodnota hloubky stromu rovna entropii $H(Y)$. Ale náhodná proměnná X je funkcí náhodné proměnné Y a tedy

$$H(X) \leq H(Y) = ET. \tag{6.5}$$

Dyadické rozdělení

Řešme nyní otázku optimality pro dyadické rozdělení.

Dyadické rozdělení

Řešme nyní otázku optimality pro dyadické rozdělení.

Věta 6.4

FI *Nechť náhodná proměnná X má dyadické rozdělení. Pak optimální algoritmus pro vygenerování X pomocí vrhu ideální mincí má střední hodnotu počtu vrhu mincí rovnu entropii, tj.*

$$H(X) = ET. \tag{6.6}$$

Důkaz Věty 6.4

MA Z věty 6.3 máme nerovnost $H(X) \leq ET$. Věnujme se nyní obrácené nerovnosti. Pro dyadické rozdělení pak Huffmanovo kódování C splňuje $l(C) = H(X)$.

Důkaz Věty 6.4

MA Z věty 6.3 máme nerovnost $H(X) \leq ET$. Věnujme se nyní obrácené nerovnosti. Pro dyadické rozdělení pak Huffmanovo kódování C splňuje $l(C) = H(X)$.

Pro každé $x \in \mathcal{X}$ je hloubka listu $k(x)$ v kódovacím stromu odpovídajícímu x rovna délce odpovídajícího kódového slova, což je $k(x) = \log_2 \frac{1}{p(x)}$.

Důkaz Věty 6.4

MA Z věty 6.3 máme nerovnost $H(X) \leq ET$. Věnujme se nyní obrácené nerovnosti. Pro dyadické rozdělení pak Huffmanovo kódování C splňuje $l(C) = H(X)$.

Pro každé $x \in \mathcal{X}$ je hloubka listu $k(x)$ v kódovacím stromu odpovídajícímu x rovna délce odpovídajícího kódového slova, což je $k(x) = \log_2 \frac{1}{p(x)}$.

Použijeme-li toto kódování pro vygenerování náhodné proměnné X, tento list bude mít pravděpodobnost $2^{-k(x)} = 2^{-\log_2 \frac{1}{p(x)}} = p(x)$.

Důkaz Věty 6.4

MA Z věty 6.3 máme nerovnost $H(X) \leq ET$. Věnujme se nyní obrácené nerovnosti. Pro dyadické rozdělení pak Huffmanovo kódování C splňuje $l(C) = H(X)$.

Pro každé $x \in \mathcal{X}$ je hloubka listu $k(x)$ v kódovacím stromu odpovídajícímu x rovna délce odpovídajícího kódového slova, což je $k(x) = \log_2 \frac{1}{p(x)}$.

Použijeme-li toto kódování pro vygenerování náhodné proměnné X, tento list bude mít pravděpodobnost $2^{-k(x)} = 2^{-\log_2 \frac{1}{p(x)}} = p(x)$.

Střední hodnota počtu vrhů mincí je pak rovna střední hodnotě hloubky stromu, což je následně rovno entropii (protože rozdělení je dyadické). Máme tedy okamžitě, že pro dyadické rozdělení optimální generující algoritmus splňuje $H(X) = ET$.

Dyadické rozdělení

FI Co se stane, pokud pravděpodobnostní rozdělení nebude dyadické?

Dyadické rozdělení

FI Co se stane, pokud pravděpodobnostní rozdělení nebude dyadické?

V tomto případě pak nelze použít tutéž myšlenku, protože Huffmanovo kódování bude generovat dyadické pravděpodobnostní rozdělení na listech, nikoliv rozdělení, se kterým jsme začali.

Dyadické rozdělení

FI Co se stane, pokud pravděpodobnostní rozdělení nebude dyadické?

V tomto případě pak nelze použít tutéž myšlenku, protože Huffmanovo kódování bude generovat dyadické pravděpodobnostní rozdělení na listech, nikoliv rozdělení, se kterým jsme začali.

Protože všechny listy z našeho stromu mají pravděpodobnost tvaru $2^{-k(z)}$, je jasné, že bychom měli rozdělit každou pravděpodobnost p_i, která není dyadická, do atomů, které už dyadické jsou.

Dyadické rozdělení

FI Co se stane, pokud pravděpodobnostní rozdělení nebude dyadické?

V tomto případě pak nelze použít tutéž myšlenku, protože Huffmanovo kódování bude generovat dyadické pravděpodobnostní rozdělení na listech, nikoliv rozdělení, se kterým jsme začali.

Protože všechny listy z našeho stromu mají pravděpodobnost tvaru $2^{-k(z)}$, je jasné, že bychom měli rozdělit každou pravděpodobnost p_i, která není dyadická, do atomů, které už dyadické jsou.

Můžeme pak připojit tyto atomy k listům našeho stromu.

Dyadické rozdělení

Například, má-li jeden z výstupů x pravděpodobnost $p(x) = \frac{1}{4}$, budeme potřebovat pouze jeden atom (list stromu v úrovni 2), ale pokud $p(x) = \frac{7}{8} = \frac{1}{2} + \frac{1}{4} + \frac{1}{8}$, budeme potřebovat tři atomy, jeden na úrovni 1, druhý na úrovni 2 a třetí na úrovni 3 našeho stromu.

Dyadické rozdělení

Například, má-li jeden z výstupů x pravděpodobnost $p(x) = \frac{1}{4}$, budeme potřebovat pouze jeden atom (list stromu v úrovni 2), ale pokud $p(x) = \frac{7}{8} = \frac{1}{2} + \frac{1}{4} + \frac{1}{8}$, budeme potřebovat tři atomy, jeden na úrovni 1, druhý na úrovni 2 a třetí na úrovni 3 našeho stromu.

Abychom minimalizovali střední hodnotu hloubky stromu, měli bychom použít atomy s co největšími pravděpodobnostmi.

Dyadické rozdělení

Například, má-li jeden z výstupů x pravděpodobnost $p(x) = \frac{1}{4}$, budeme potřebovat pouze jeden atom (list stromu v úrovni 2), ale pokud $p(x) = \frac{7}{8} = \frac{1}{2} + \frac{1}{4} + \frac{1}{8}$, budeme potřebovat tři atomy, jeden na úrovni 1, druhý na úrovni 2 a třetí na úrovni 3 našeho stromu.

Abychom minimalizovali střední hodnotu hloubky stromu, měli bychom použít atomy s co největšími pravděpodobnostmi.

Tedy, je-li dána pravděpodobnost p_i, která není dyadická, najdeme největší atom tvaru 2^{-k}, který je menší než p_i a připojíme tento atom ke stromu. Pak spočítáme zbytek a najdeme největší atom, který je pod zbytkem.

Dyadické rozdělení

Například, má-li jeden z výstupů x pravděpodobnost $p(x) = \frac{1}{4}$, budeme potřebovat pouze jeden atom (list stromu v úrovni 2), ale pokud $p(x) = \frac{7}{8} = \frac{1}{2} + \frac{1}{4} + \frac{1}{8}$, budeme potřebovat tři atomy, jeden na úrovni 1, druhý na úrovni 2 a třetí na úrovni 3 našeho stromu.

Abychom minimalizovali střední hodnotu hloubky stromu, měli bychom použít atomy s co největšími pravděpodobnostmi.

Tedy, je-li dána pravděpodobnost p_i, která není dyadická, najdeme největší atom tvaru 2^{-k}, který je menší než p_i a připojíme tento atom ke stromu. Pak spočítáme zbytek a najdeme největší atom, který je pod zbytkem.

Tento proces opakujeme a pak lze rozdělit všechny pravděpodobnosti do dyadických atomů.

Dyadické rozdělení

Zřejmě je tento proces ekvivalentní nalezení binárních rozvojů pravděpodobností p_i. Buď tedy binární rozvoj pravděpodobnosti p_i tvaru

$$p_i = \sum_{j \geq 1} p_i^{(j)}, \tag{6.7}$$

kde $p_i^{(j)} = 2^{-j}$ nebo 0.

Dyadické rozdělení

Zřejmě je tento proces ekvivalentní nalezení binárních rozvojů pravděpodobností p_i. Buď tedy binární rozvoj pravděpodobnosti p_i tvaru

$$p_i = \sum_{j \geq 1} p_i^{(j)}, \tag{6.7}$$

kde $p_i^{(j)} = 2^{-j}$ nebo 0. Pak atomy rozvoje jsou

$$\{p_i^{(j)} \mid i = 1, 2, \ldots, m, j \geq 1\}.$$

Protože $\sum_{i=1}^{m} p_i = 1$, bude i součet těchto atomů $\sum_{i,j \geq 1} p_i^{(j)} = 1$. Přitom připojíme atom s pravděpodobností 2^{-j} jako list hloubky j ke stromu.

Dyadické rozdělení

Zřejmě je tento proces ekvivalentní nalezení binárních rozvojů pravděpodobností p_i. Buď tedy binární rozvoj pravděpodobnosti p_i tvaru

$$p_i = \sum_{j \geq 1} p_i^{(j)}, \tag{6.7}$$

kde $p_i^{(j)} = 2^{-j}$ nebo 0. Pak atomy rozvoje jsou

$$\{p_i^{(j)} \mid i = 1, 2, \ldots, m, j \geq 1\}.$$

Protože $\sum_{i=1}^{m} p_i = 1$, bude i součet těchto atomů $\sum_{i,j \geq 1} p_i^{(j)} = 1$. Přitom připojíme atom s pravděpodobností 2^{-j} jako list hloubky j ke stromu.

Hloubky atomů splňují Kraftovu nerovnost, můžeme tedy zkonstruovat binární strom, kde atomy mají požadovanou hloubku.

Dyadické rozdělení - Příklad

Příklad 6.5

Nechť náhodná proměnná X má pravděpodobnostní rozdělení

$$X = \begin{cases} a & \text{s pravděpodobností } \frac{2}{3}, \\ b & \text{s pravděpodobností } \frac{1}{3}. \end{cases} \tag{6.8}$$

Dyadické rozdělení - Příklad

Příklad 6.5

Nechť náhodná proměnná X má pravděpodobnostní rozdělení

$$X = \begin{cases} a & \text{s pravděpodobností } \frac{2}{3}, \\ b & \text{s pravděpodobností } \frac{1}{3}. \end{cases} \tag{6.8}$$

Nejprve určíme binární rozvoje těchto pravděpodobností:

$$\frac{2}{3} = 0.10101010\ldots_2, \qquad \frac{1}{3} = 0.01010101\ldots_2. \tag{6.9}$$

Dyadické rozdělení - Příklad

Příklad 6.5

Nechť náhodná proměnná X má pravděpodobnostní rozdělení

$$X = \begin{cases} a & \text{s pravděpodobností } \frac{2}{3}, \\ b & \text{s pravděpodobností } \frac{1}{3}. \end{cases} \tag{6.8}$$

Nejprve určíme binární rozvoje těchto pravděpodobností:

$$\frac{2}{3} = 0.10101010\ldots_2, \qquad \frac{1}{3} = 0.01010101\ldots_2. \tag{6.9}$$

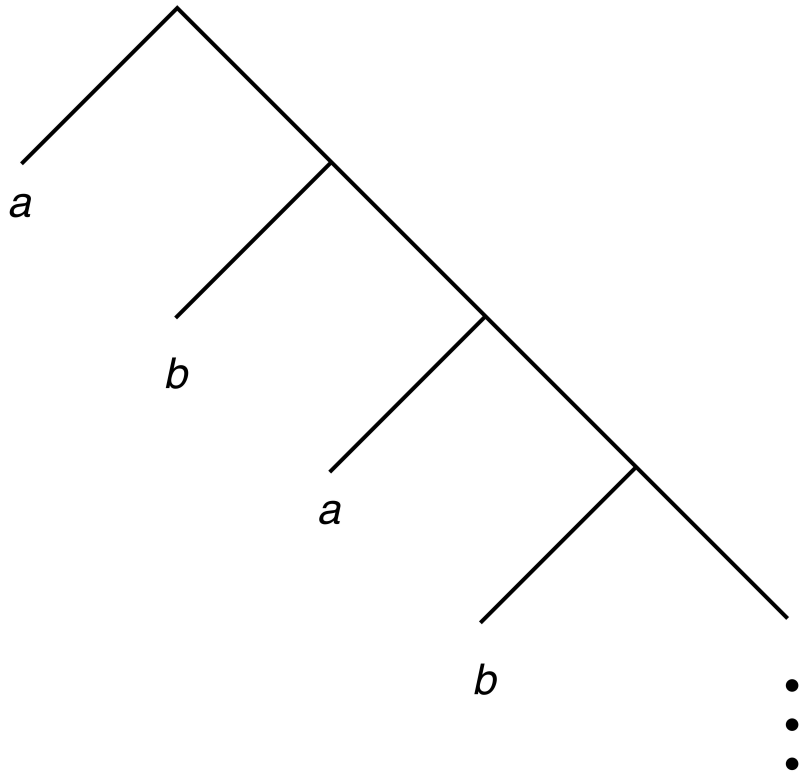
Pak příslušné atomy pro rozvoj jsou:

$$\frac{2}{3} \mapsto \left(\frac{1}{2}, \frac{1}{8}, \frac{1}{32}, \ldots\right), \qquad \frac{1}{3} \mapsto \left(\frac{1}{4}, \frac{1}{16}, \frac{1}{64}, \ldots\right). \tag{6.10}$$

Dyadické rozdělení - Příklad

Příklad 6.5 (Pokračování)

Tyto atomy můžeme připojit ke stromu (viz sousední obrázek), který generuje pravděpodobnostní rozdělení $\frac{2}{3}, \frac{1}{3}$.



Dyadické rozdělení

Tato procedura nám poskytne binární strom, který generuje náhodnou proměnnou X.

Dyadické rozdělení

Tato procedura nám poskytne binární strom, který generuje náhodnou proměnnou X.

V předchozím jsme tvrdili, že tato procedura je optimální v tom smyslu, že náš binární strom bude mít minimální střední hodnotu hloubky.

Dyadické rozdělení

Tato procedura nám poskytne binární strom, který generuje náhodnou proměnnou X.

V předchozím jsme tvrdili, že tato procedura je optimální v tom smyslu, že náš binární strom bude mít minimální střední hodnotu hloubky.

Nepodáme zde úplný formální důkaz, ale namísto toho ohraničíme střední hodnotu hloubky stromu vygenerovaného tímto postupem.

Dyadické rozdělení - Odhad hloubky stromu

Věta 6.6

Pro optimální algoritmus generující náhodnou proměnnou X střední hodnota počtu ideálních bitů mezi $H(X)$ a $H(X)+2$, tj.

$$H(X) \leq ET < H(X) + 2. \tag{6.11}$$

Dyadické rozdělení - Odhad hloubky stromu

Věta 6.6

Pro optimální algoritmus generující náhodnou proměnnou X střední hodnota počtu ideálních bitů mezi $H(X)$ a $H(X)+2$, tj.

$$H(X) \leq ET < H(X) + 2. \tag{6.11}$$

Důkaz. **MA** Dle věty 6.3 máme $H(X) \leq ET$. Abychom určili horní hranici, určeme pro každou pravděpodobnost p_i binární rozvoj

$$p_i = \sum_{j \geq 1} p_i^{(j)}, \tag{6.12}$$

kde $p_i^{(j)} = 2^{-j}$ nebo 0.

Dyadické rozdělení - Odhad hloubky stromu

Věta 6.6

Pro optimální algoritmus generující náhodnou proměnnou X střední hodnota počtu ideálních bitů mezi $H(X)$ a $H(X)+2$, tj.

$$H(X) \leq ET < H(X) + 2. \tag{6.11}$$

Důkaz. **MA** Dle věty 6.3 máme $H(X) \leq ET$. Abychom určili horní hranici, určeme pro každou pravděpodobnost p_i binární rozvoj

$$p_i = \sum_{j \geq 1} p_i^{(j)}, \tag{6.12}$$

kde $p_i^{(j)} = 2^{-j}$ nebo 0.
Zkonstruujeme pomocí těchto atomů binární strom tak, že jeho listy odpovídají atomům našeho dyadického rozdělení.

Důkaz Věty 6.6 - Odhad hloubky stromu

Zkonstruujeme pomocí těchto atomů binární strom tak, že jeho listy odpovídají atomům našeho dyadického rozdělení.

Důkaz Věty 6.6 - Odhad hloubky stromu

Zkonstruujeme pomocí těchto atomů binární strom tak, že jeho listy odpovídají atomům našeho dyadického rozdělení.

Pak počet vrhů ideální mincí potřebných pro vygenerování každého atomu je jeho hloubka v našem stromu.

Důkaz Věty 6.6 - Odhad hloubky stromu

Zkonstruujeme pomocí těchto atomů binární strom tak, že jeho listy odpovídají atomům našeho dyadického rozdělení.

Pak počet vrhů ideální mincí potřebných pro vygenerování každého atomu je jeho hloubka v našem stromu.

Tedy **střední hodnota počtu vrhů ideální mincí** je rovna **střední hodnotě hloubky našeho stromu**, která je okamžitě rovna entropii dyadického rozdělení atomů.

Důkaz Věty 6.6 - Odhad hloubky stromu

Zkonstruujeme pomocí těchto atomů binární strom tak, že jeho listy odpovídají atomům našeho dyadického rozdělení.

Pak počet vrhů ideální mincí potřebných pro vygenerování každého atomu je jeho hloubka v našem stromu.

Tedy **střední hodnota počtu vrhů ideální mincí** je rovna **střední hodnotě hloubky našeho stromu**, která je okamžitě rovna entropii dyadického rozdělení atomů.

Tudíž

$$ET = H(Y), \tag{6.13}$$

kde náhodná proměnná Y má pravděpodobnostní rozdělení $p_1^{(1)}, p_1^{(2)}, \ldots, p_2^{(1)}, p_2^{(2)}, \ldots, p_m^{(1)}, p_m^{(2)}, \ldots$.

Důkaz Věty 6.6 - Odhad hloubky stromu

Protože náhodná proměnná X je funkcí náhodné proměnné Y, z řetězcového pravidla okamžitě obdržíme

$$H(Y) = H(Y, X) = H(X) + H(Y|X), \tag{6.14}$$

stačí ověřit, že $H(Y|X) < 2$.

Důkaz Věty 6.6 - Odhad hloubky stromu

Protože náhodná proměnná X je funkcí náhodné proměnné Y, z řetězcového pravidla okamžitě obdržíme

$$H(Y) = H(Y, X) = H(X) + H(Y|X), \tag{6.14}$$

stačí ověřit, že $H(Y|X) < 2$.

Uvažujme nyní příslušný podstrom a podterm T_i pro každou pravděpodobnost p_i. Pak

$$T_i = \sum_{j \geq 1 \,:\, p_i^{(j)} > 0} j 2^{-j} \qquad \text{a} \qquad H(Y) = \sum_{i=1}^{m} T_i. \tag{6.15}$$

Důkaz Věty 6.6 - Odhad hloubky stromu

Můžeme najít takové přirozené číslo n_i, že $2^{-n_i} \leq p_i < 2^{-n_i+1}$, neboli

$$n_i - 1 < -\log p_i \leq n_i. \tag{6.16}$$

Důkaz Věty 6.6 - Odhad hloubky stromu

Můžeme najít takové přirozené číslo n_i, že $2^{-n_i} \leq p_i < 2^{-n_i+1}$, neboli

$$n_i - 1 < -\log p_i \leq n_i. \tag{6.16}$$

Nutně tedy $p_i^{(j)} > 0$ pouze pokud $j \geq n$. Můžeme tedy přepsat (6.15) do tvaru

$$T_i = \sum_{j \geq n_i \,:\, p_i^{(j)} > 0} j 2^{-j} \qquad \text{a} \qquad H(Y) = \sum_{i=1}^{m} T_i. \tag{6.17}$$

Důkaz Věty 6.6 - Odhad hloubky stromu

Můžeme najít takové přirozené číslo n_i, že $2^{-n_i} \leq p_i < 2^{-n_i+1}$, neboli

$$n_i - 1 < -\log p_i \leq n_i. \tag{6.16}$$

Nutně tedy $p_i^{(j)} > 0$ pouze pokud $j \geq n$. Můžeme tedy přepsat (6.15) do tvaru

$$T_i = \sum_{j \geq n_i \,:\, p_i^{(j)} > 0} j 2^{-j} \qquad \text{a} \qquad H(Y) = \sum_{i=1}^{m} T_i. \tag{6.17}$$

Zároveň dle definice atomu můžeme zapsat p_i jakožto

$$p_i = \sum_{j \geq n_i \,:\, p_i^{(j)} > 0} 2^{-j}. \tag{6.18}$$

Důkaz Věty 6.6 - Odhad hloubky stromu

Ukažme nejprve, že platí $T_i < -p_i \log p_i + 2p_i$. Zřejmě

$$T_i + p_i \log p_i - 2p_i \overset{(a)}{<} T_i - p_i(n_i - 1) - 2p_i = T_i - (n_i + 1)p_i$$

Důkaz Věty 6.6 - Odhad hloubky stromu

Ukažme nejprve, že platí $T_i < -p_i \log p_i + 2p_i$. Zřejmě

$$T_i + p_i \log p_i - 2p_i \overset{(a)}{<} T_i - p_i(n_i - 1) - 2p_i = T_i - (n_i + 1)p_i$$
$$= \sum_{j \geq n_i : p_i^{(j)} > 0} j 2^{-j} - (n_i + 1) \sum_{j \geq n_i : p_i^{(j)} > 0} 2^{-j}$$

Důkaz Věty 6.6 - Odhad hloubky stromu

Ukažme nejprve, že platí $T_i < -p_i \log p_i + 2p_i$. Zřejmě

$$T_i + p_i \log p_i - 2p_i \overset{(a)}{<} T_i - p_i(n_i - 1) - 2p_i = T_i - (n_i + 1)p_i$$
$$= \sum_{j \geq n_i : p_i^{(j)} > 0} j 2^{-j} - (n_i + 1) \sum_{j \geq n_i : p_i^{(j)} > 0} 2^{-j}$$
$$= \sum_{j \geq n_i : p_i^{(j)} > 0} (j - n_i - 1) 2^{-j}$$

Důkaz Věty 6.6 - Odhad hloubky stromu

Ukažme nejprve, že platí $T_i < -p_i \log p_i + 2p_i$. Zřejmě

$$T_i + p_i \log p_i - 2p_i \overset{(a)}{<} T_i - p_i(n_i - 1) - 2p_i = T_i - (n_i + 1)p_i$$
$$= \sum_{j \geq n_i : p_i^{(j)} > 0} j 2^{-j} - (n_i + 1) \sum_{j \geq n_i : p_i^{(j)} > 0} 2^{-j}$$
$$= \sum_{j \geq n_i : p_i^{(j)} > 0} (j - n_i - 1) 2^{-j}$$
$$= -2^{-n_i} + 0 + \sum_{j \geq (n_i + 2) : p_i^{(j)} > 0} (j - n_i - 1) 2^{-j}$$

Důkaz Věty 6.6 - Odhad hloubky stromu

Ukažme nejprve, že platí $T_i < -p_i \log p_i + 2p_i$. Zřejmě

$$
\begin{aligned}
T_i + p_i \log p_i - 2p_i &\overset{(a)}{<} T_i - p_i(n_i - 1) - 2p_i = T_i - (n_i + 1)p_i \\
&= \sum_{j \geq n_i \,:\, p_i^{(j)} > 0} j 2^{-j} - (n_i + 1) \sum_{j \geq n_i \,:\, p_i^{(j)} > 0} 2^{-j} \\
&= \sum_{j \geq n_i \,:\, p_i^{(j)} > 0} (j - n_i - 1) 2^{-j} \\
&= -2^{-n_i} + 0 + \sum_{j \geq (n_i+2) \,:\, p_i^{(j)} > 0} (j - n_i - 1) 2^{-j} \\
&\overset{(b)}{=} -2^{-n_i} + \sum_{k \geq 1 \,:\, p_i^{(k+n_i+1)} > 0} k 2^{-(k+n_i+1)}
\end{aligned}
$$

Důkaz Věty 6.6 - Odhad hloubky stromu

Ukažme nejprve, že platí $T_i < -p_i \log p_i + 2p_i$. Zřejmě

$$
\begin{aligned}
T_i + p_i \log p_i - 2p_i &\overset{(a)}{<} T_i - p_i(n_i - 1) - 2p_i = T_i - (n_i + 1)p_i \\
&= \sum_{j \geq n_i \,:\, p_i^{(j)} > 0} j 2^{-j} - (n_i + 1) \sum_{j \geq n_i \,:\, p_i^{(j)} > 0} 2^{-j} \\
&= \sum_{j \geq n_i \,:\, p_i^{(j)} > 0} (j - n_i - 1) 2^{-j} \\
&= -2^{-n_i} + 0 + \sum_{j \geq (n_i+2) \,:\, p_i^{(j)} > 0} (j - n_i - 1) 2^{-j} \\
&\overset{(b)}{=} -2^{-n_i} + \sum_{k \geq 1 \,:\, p_i^{(k+n_i+1)} > 0} k 2^{-(k+n_i+1)} \\
&\overset{(c)}{\leq} -2^{-n_i} + \sum_{k \geq 1} k 2^{-(k+n_i+1)} = -2^{-n_i} + -2^{-(n_i+1)} 2 = 0,
\end{aligned}
$$

Důkaz Věty 6.6 - Odhad hloubky stromu

Ukažme nejprve, že platí $T_i < -p_i \log p_i + 2p_i$. Zřejmě

$$
\begin{aligned}
T_i + p_i \log p_i - 2p_i &\overset{(a)}{<} T_i - p_i(n_i - 1) - 2p_i = T_i - (n_i + 1)p_i \\
&= \sum_{j \geq n_i \,:\, p_i^{(j)} > 0} j 2^{-j} - (n_i + 1) \sum_{j \geq n_i \,:\, p_i^{(j)} > 0} 2^{-j} \\
&= \sum_{j \geq n_i \,:\, p_i^{(j)} > 0} (j - n_i - 1) 2^{-j} \\
&= -2^{-n_i} + 0 + \sum_{j \geq (n_i+2) \,:\, p_i^{(j)} > 0} (j - n_i - 1) 2^{-j} \\
&\overset{(b)}{=} -2^{-n_i} + \sum_{k \geq 1 \,:\, p_i^{(k+n_i+1)} > 0} k 2^{-(k+n_i+1)} \\
&\overset{(c)}{\leq} -2^{-n_i} + \sum_{k \geq 1} k 2^{-(k+n_i+1)} = -2^{-n_i} + -2^{-(n_i+1)} 2 = 0,
\end{aligned}
$$

přičemž (a) plyne ze vztahu (6.16), (b) plyne ze změny proměnných v rámci sumy a (c) plyne z rozšíření oboru sumace.

Důkaz Věty 6.6 - Odhad hloubky stromu

Ukázali jsme tedy, že

$$T_i < -p_i \log p_i + 2p_i. \tag{6.19}$$

Důkaz Věty 6.6 - Odhad hloubky stromu

Ukázali jsme tedy, že

$$T_i < -p_i \log p_i + 2p_i. \tag{6.19}$$

Protože $ET = \sum_{i=1}^{m} T_i$, máme okamžitě, že

$$ET = \sum_{i=1}^{m} T_i < -\sum_{i=1}^{m} p_i \log p_i + 2\sum_{i=1}^{m} p_i = H(X) + 2. \tag{6.20}$$

Problémy k řešení

Problémy 2

1. *Ve hře na šachovnici má jeden z hráčů (A) uhádnout, kam jeho protivník umístil královnu. Hráči A je povoleno šest otázek, které musí být pravdivě zodpovězeny odpovědí ano/ne. Dokažte, že existuje strategie, při které může hráč A vždy vyhrát tuto hru, ale že nelze zajistit výhru, pokud má povoleno pouze pět otázek.*
2. *Je-li hra z předchozího příkladu hraná na šachovnici o rozměrech $n \times n$, kolik otázek potřebuje hráč A, aby bezpečně vyhrál.*

Problémy k řešení

Problémy 2

3. *Najděte průměrnou délku optimálního (kompaktního) jednoznačně dekódovatelného binárního kódu pro zdroj bez paměti, který vysílá šest slov s pravděpodobnostmi*

$$0.25, 0.10, 0.15, 0.05, 0.20, 0.25.$$

Analyzováním Huffmanova algoritmu ukažme, že zdroj bez paměti vysílá N slov a jestliže $l_1, \ldots, l_N$ jsou délky kódových slov optimálního kódování nad binární abecedou, pak $l_1 + \cdots + l_N \leq \frac{1}{2}(N^2 + N - 2)$.

Problémy k řešení

Problémy 2

4. *Máte k dispozici rovnovážnou váhu a devět zdánlivě identických mincí. Bylo Vám sděleno, že jedna mince je různá od zbývajících stejných mincí. Máte za úkol najít, o kterou minci se jedná a zda je těžší nebo lehčí. Navrhněte strategii o nejvýše 3 váženích pro řešení tohoto problému. Abychom obecně vyřešili tentýž problém pro n mincí v k váženích, je nutno, aby platilo, že* $k \log 3 \geq \log 2n$. *Dokažte.*
5. *V Huffmanově algoritmu pro binární abecedu aplikovaném na N zdrojových slov jsou délky slov pro konečné optimální zakódování* $l_1, \ldots, l_N$. *Dokažte, že*

$$l_1 + \cdots + l_N \geq N \log_2 N.$$

Problémy k řešení

Problémy 2

6. *Mějme dva hráče, hráče A a hráče B. Tito hráči hrají hru s náhodnou kostkou, která má n stran a nabývá hodnot* $1, \ldots, n$ *s pravděpodobnostmi* $p_1, p_2, \ldots, p_n$.
Hráč B vrhá kostkou za závěsem a hráč A má zjistit hodnotu po vržení kostky co možná nejrychleji. Hráč A se může ptát hráče B a ten mu musí pravdivě odpovídat buď ano nebo ne. Ukažte, že průměrný počet otázek pro úspěšnou strategii hráče A musí být alespoň entropie $H = -\sum p_j \log p_j$.

Problémy k řešení

Problémy 2

7. *Ukažte, že optimální zakódování zdroje s N stejně pravděpodobnými zdrojovými slovy v abecedě o D písmenech, je*

$$max\{(D^{r+1} - N - b)/(D-1), N\}$$

slov délky r, kde b je určeno vztahem

$$N + b = D + k(D-1), \quad 0 \leq b < D - 1,$$

a r je největší přirozené číslo tak, že

$$D^r \leq N + b.$$

Problémy k řešení

Problémy 2

8. *Dokažte, že následující metoda nám dává jednoznačně dekódovatelný kód pro zdroj $\mathcal{S}$ (mluvíme o tzv.* ***Shannonově kódování****).*
Předpokládejme, že $\mathcal{S}$ má N zdrojových slov $s_1, s_2, \ldots, s_N$ a p_i je pravděpodobnost, že je vysláno slovo s_i, přičemž platí $p_i \geq p_{i+1}$. Nechť navíc

$$a_1 = 0, \quad a_2 = p_1, \quad a_3 = p_1 + p_2, \ldots,$$
$$a_N = p_1 + p_2 + \cdots + p_{N-1}.$$

Nechť m_i $(1 \leq i \leq N)$ je definováno jako nejmenší takové přirozené číslo m_i splňující $2^{-m_i} \leq p_i$, tj. $m_i = \lceil \log_2 \frac{1}{p_i} \rceil$.

Problémy k řešení

Problémy 2

8. *Je-li pak a_j^* binární rozvoj čísla a_j na m_j desetinných míst, je kódování*

$$s_j \mapsto a_j^*, \quad 1 \leq j \leq N$$

jednoznačně dekódovatelné pro zdroj $\mathcal{S}$. Ukažte, že se nejedná o optimální zakódování, ale že průměrná délka $\hat{l}$ kódování splňuje

$$H(S) \leq \hat{l} \leq H(S) + 1.$$

Problémy k řešení

Problémy 2

9. *Jsou-li $l_1, \ldots, l_n$ délky slov binárního Huffmanova kódování zdrojových slov majících pravděpodobnost $p_1, \ldots, p_n$, pak* redundance *kódování je definována jako*

$$r = \sum_{k=1}^{n} p_k l_k - H(p_1, \ldots, p_n).$$

Ukažte, že platí

$$r \leq p_{max} + \log[2(\log e)/e] = p_{max} + 0.086,$$

kde $p_{max} = \max_i p_i$.

Problémy k řešení

Problémy 2

10. *Buď C prefixové kódování zdroje s N zdrojovými slovy v abecedě Σ o D písmenech se slovními délkami $l_1, \ldots, l_N$ takové, že platí*

$$\sum_{i=1}^{N} D^{-l_i} < 1.$$

Ukažte, že existuje libovolně dlouhá posloupnost v Σ^*, kterou nelze dekódovat.