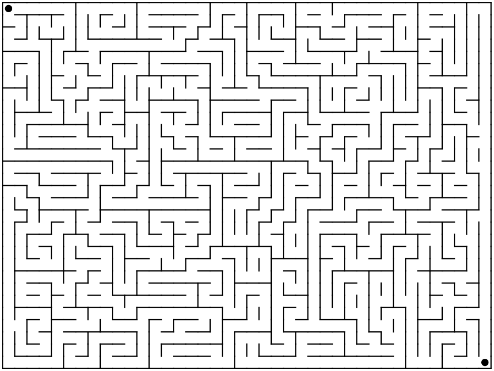
Grafové algoritmy

IB111 Programování a algoritmizace

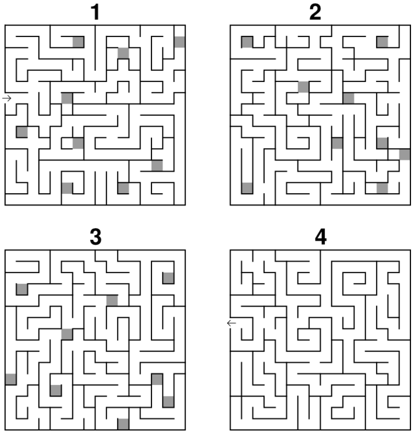
2011

Motivační příklad I



Kde prokopat zeď, aby se bludiště stalo průchodným?

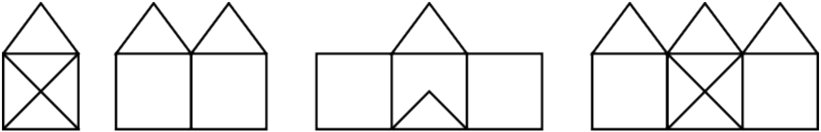
Motivační příklad II



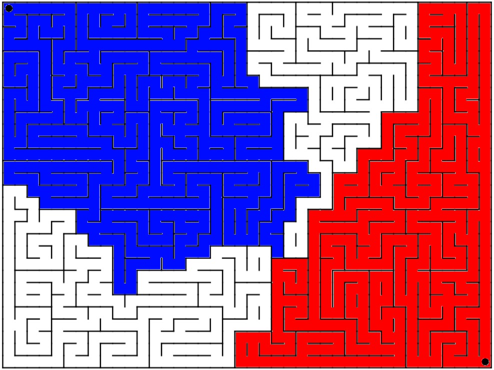
šedé pole = žebřík o patro nahoru

Motivační příklad III

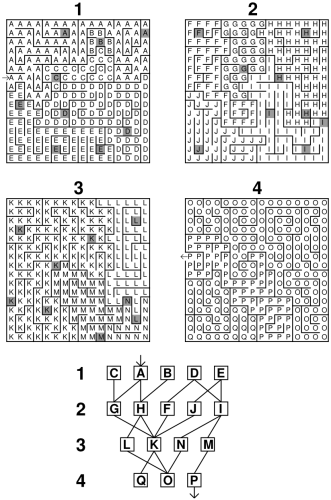
Lze obrázek nakreslit jedním (uzavřeným) tahem?



Motivační příklad I řešení

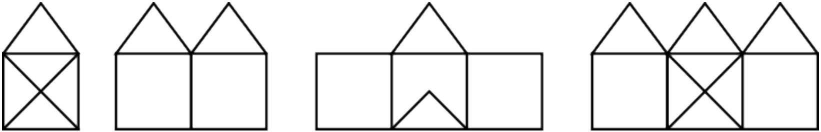


Motivační příklad II řešení



Motivační příklad III

Lze obrázek nakreslit jedním (uzavřeným) tahem?



- spočítat počty sousedů (stupeň vrcholu)
- řešitelné – maximálně dva stupně liché
- uzavřený tah – všechny stupně sudé

Graf

Základní pojmy:

- uzly (vrcholy)
- hrany: orientované, neorientované, vážené
- stupeň vrcholu
- cesta v grafu, dosažitelnost, cyklus
- souvislost, komponenty
- strom, klika (úplný graf)

Použití grafů

- dopravní sítě (silniční, vlaková, letecká, ...)
- elektrická síť
- Internet
- sociální sítě
- plánování: závislosti mezi úlohami
- stavový prostor logické úlohy

Potravní řetězce

uzly: zvířata, hrany: pokud jedno žere druhé

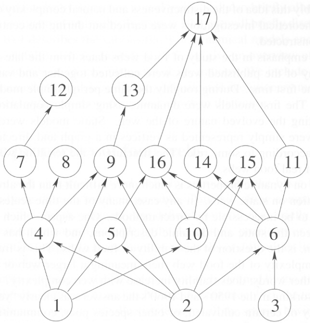
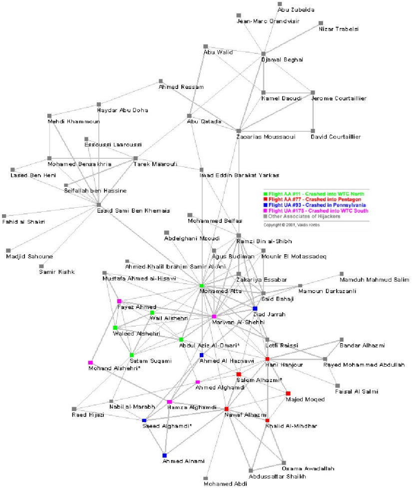


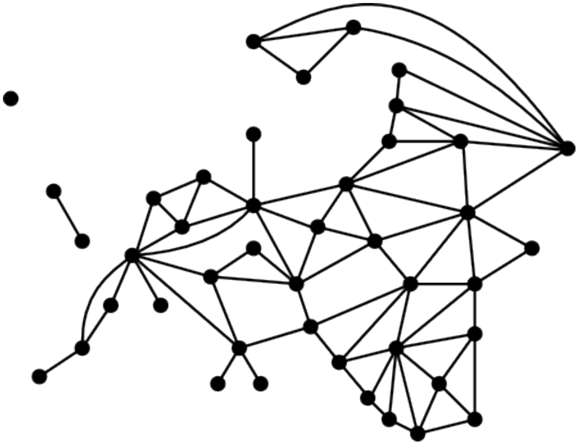
Figure 10.1: Narragansett Bay food web. 1=flagellates, diatoms; 2=particulate detritus; 3=macroalgae, eelgrass; 4=*Acartia*, other copepods; 5=sponges, clams; 6=benthic macrofauna; 7=ctenophores; 8=meroplankton, fish larvae; 9=pacific menhaden; 10=bivalves; 11=crabs, lobsters; 12=butterfish; 13=striped bass, bluefish, mackerel; 14=demersal species; 15=starfish; 16=flounder; 17=man. (After Yodzis 1989. *Introduction to theoretical ecology*, Harper and Row, New York.)

Teroristi



Abu Zubaida
Jean-Marc Grandvisir
Nizar Trabelsi
Abu Walid
Djamal Beghal
Ahmed Ressam
Kamel Daoudi
Jerome Courtaillier
Haydar Abu Doha
Mehdi Khammoun
Abu Qatada
Zacarias Moussaoui
David Courtaillier
Essoussi Laaroussi
Mohamed Bensakhria
Tarek Maaroufi
Lased Ben Heni
Imad Eddin Barakat Yarkas
Seifallah ben Hassine
Flight AA #11 - Crashed into WTC North
Flight AA #77 - Crashed into Pentagon
Flight UA #93 - Crashed in Pennsylvania
Flight UA #175 - Crashed into WTC South
Other Associates of Hijackers
Copyright © 2001, Valdis Krebs
Essid Sami Ben Khemais
Fahid al Shakri
Mohammed Belfas
Abdelghani Mzoudi
Ramzi Bin al-Shibh
Madjid Sahoune
Agus Budiman
Mounir El Motassadeq
Ahmed Khalil Ibrahim Samir Al-Ani
Samir Kishk
Mustafa Ahmed al-Hisawi
Zakariya Essabar
Mamduh Mahmud Salim
Mohamed Atta
Mamoun Darkazanli
Said Bahaji
Fayez Ahmed
Wail Alshehri
Ziad Jarrah
Marwan Al-Shehhi
Waleed Alshehri
Abdul Aziz Al-Omari*
Lotfi Raissi
Bandar Alhazmi
Satam Suqami
Ahmed Al Haznawi
Hani Hanjour
Rayed Mohammed Abdullah
Mohand Alshehri*
Salem Alhazmi*
Ahmed Alghamdi
Faisal Al Salmi
Majed Moqed
Nabil al-Marabh
Hamza Alghamdi
Raed Hijazi
Nawaf Alhazmi
Saeed Alghamdi*
Khalid Al-Mihdhar
Ahmed Alnami
Osama Awadallah
Abdussattar Shaikh
Mohamed Abdi

Co je to?



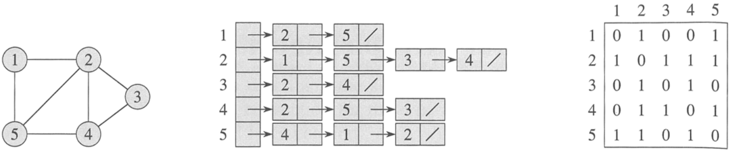
Logická úloha: Misionáři a kanibalové

- 3 misionáři, 3 kanibalové
- řeka, 1 loďka (max 2 lidé)
- víc kanibalů jak misionářů na jednom místě $\Rightarrow$ problém
- (jen jeden misionář a jeden kanibal umí pádlovat)
- zkuste:
    - najít řešení
    - „najít“ v tom graf

Reprezentace grafu v počítači

- seznamy sousedů
  - pro každý vrchol seznam jeho sousedů
  - vhodné pro řídké matice
- matice sousednosti
  - binární matice
  - pro každou dvojici vrcholů: sousedí (1) / nesousedí (0)
  - vhodné pro malé nebo husté matice

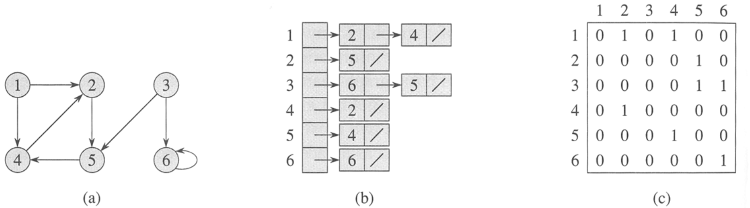
Reprezentace grafu: příklad neorientovaný graf



(a) (b) (c)

T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein: Introduction to Algorithms.

Reprezentace grafu: příklad orientovaný graf



T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein: Introduction to Algorithms.

Procházení grafu

procházení grafu = základní operace nad grafy

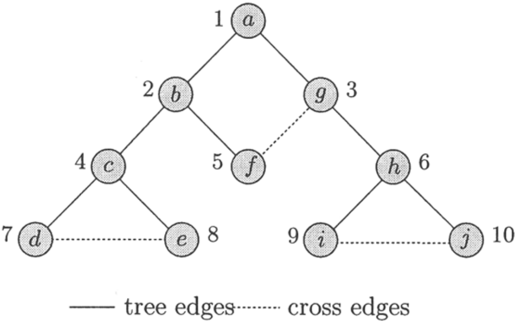
- procházení do šířky
- procházení do hloubky

Procházení do šířky

BFS = breadth-first search

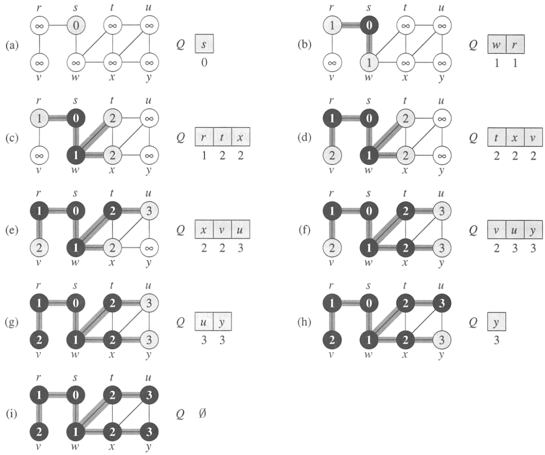
- „povodeň“ z počátečního vrcholu
- realizace pomocí fronty
- pro aktuální vrchol:
  - projdu všechny sousedy
  - pokud soused nebyl navštíven, dám jej do fronty

Ilustrace



M. H. Alsuwaiyel: Algorithms, Design Techniques and Analysis.

Ilustrace 2



T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein: Introduction to Algorithms.

Pseudokód

Algorithm 9.4 BFS

Input: A directed or undirected graph $G = (V, E)$.

Output: Numbering of the vertices in breadth-first search order.

```
bfn ← 0
for each vertex v ∈ V
    mark v unvisited
end for
for each vertex v ∈ V
    if v is marked unvisited then bfs(v)
end for
```

Procedure *bfs*(v)

```
Q ← {v}
mark v visited
while Q ≠ {}
    v ← Pop(Q)
    bfn ← bfn + 1
    for each edge (v, w) ∈ E
        if w is marked unvisited then
            Push(w, Q)
            mark w visited
        end if
    end for
end while
```

M. H. Alsuwaiyel: Algorithms, Design Techniques and Analysis.

Procházení do šířky – poznámky

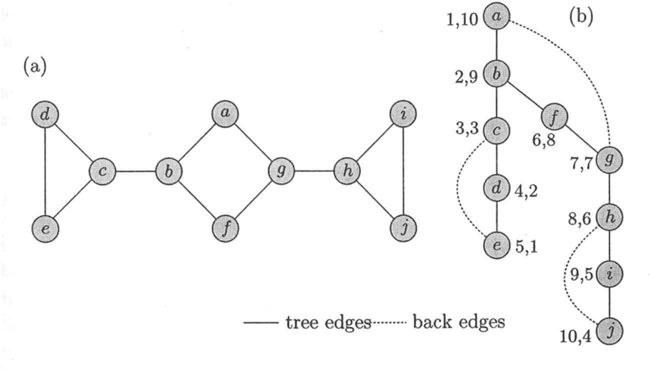
- procházím v pořadí podle vzdálenosti od zdroje
- jednoduché napočítat vzdálenosti (počet kroků)
- strom předchůdců – rekonstrukce nejkratších cest

Procházení do hloubky

DFS = depht-first search

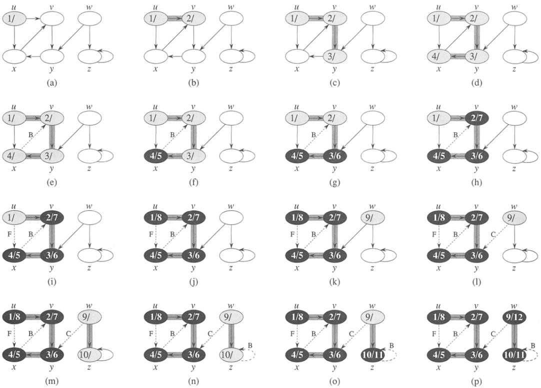
- „zavrtávání“ se do hloubky
- realizace pomocí rekurze (zásobníku)
- pokud najdu nenavštíveného souseda, začnu prohledávat z něho
- ostatní sousedy obsloužím až později

Ilustrace



M. H. Alsuwaiyel: Algorithms, Design Techniques and Analysis.

Ilustrace 2



T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein: Introduction to Algorithms.

Pseudokód

Algorithm 9.1 DFS

Input: A (directed or undirected) graph $G = (V, E)$.

Output: Preordering and postordering of the vertices in the corresponding depth-first search tree.

```
predfn ← 0;    postdfn ← 0
for each vertex v ∈ V
    mark v unvisited
end for
for each vertex v ∈ V
    if v is marked unvisited then dfs(v)
end for
```

Procedure *dfs*(v)

```
mark v visited
predfn ← predfn + 1
for each edge (v, w) ∈ E
    if w is marked unvisited then dfs(w)
end for
postdfn ← postdfn + 1
```

M. H. Alsuwaiyel: Algorithms, Design Techniques and Analysis.

Procházení do hloubky – poznámky

- jednoduché na implementaci pomocí rekurze
- užitečné vlastnosti využitelné pro aplikace, např.
  - detekce cyklů
  - (silné) komponenty souvislosti
  - topologické třídění

Aplikace prohledávání

- komponenty souvislosti
- hledání nejkratších cest
- detekce cyklů
- bipartita

Komponenty souvislosti

- komponenta souvislosti = maximální množina vrcholů $U \subseteq V$, každé dva vrcholy z U vzájemně dosažitelné
- detekce pomocí prohledávání (do šířky, do hloubky)

Hledání nejkratších cest

- pokud všechny hrany mají váhu 1
- hledání nejkratších cest pomocí BFS
- jednoduchá úprava

Detekce cyklu

- cyklus (kružnice) – cesta z vrcholu sama do sebe
- jak poznat, zda graf obsahuje cyklus?

Detekce cyklu

- cyklus (kružnice) – cesta z vrcholu sama do sebe
- jak poznat, zda graf obsahuje cyklus?
- neorientovaný, souvislý – spočítat počet hran a vrcholů
- orientovaný – pomocí DFS
- kontrola, zda je soused aktuálního vrcholu na zásobníku

Bipartitní graf

- bipartitní graf
  - existuje rozdělení množiny vrcholů na V_1, V_2
  - hrany vedou pouze mezi V_1 a V_2, nikoliv v rámci množin
- jak poznat, zda je graf bipartitní?

Bipartitní graf

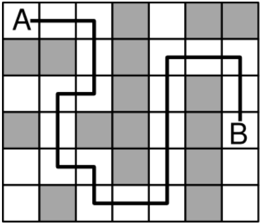
- bipartitní graf
  - existuje rozdělení množiny vrcholů na V_1, V_2
  - hrany vedou pouze mezi V_1 a V_2, nikoliv v rámci množin
- jak poznat, zda je graf bipartitní?
- pomocí BFS (či DFS) – průběžně přiřazuji do množiny 1/2 a kontroluji

Použití procházení grafu

- mnoho problémů lze řešit jednoduše pomocí aplikace procházení grafu
- klíčové správně „pojmenovat“ graf

Příklad: bludiště základní

Základní zadání



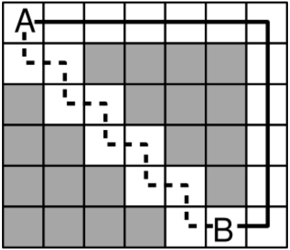
Textový zápis zadání

```
A..#.##
##.#...
...#.#.
#.##.#B
...#.#.
.#...#.
```

Textový zápis řešení

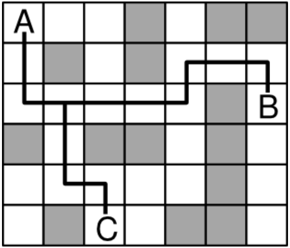
```
Axx#.##
##x#xxx
.xx#x#x
#x##x#B
.xx#x#.
.#xxx#.
```

Příklad: robot v bludišti



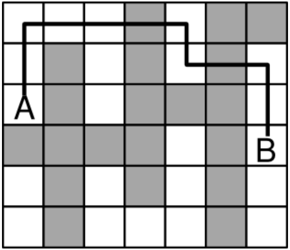
- čtverečkované bludiště
- robot s operacemi: krok, rotace vlevo, rotace vpravo
- jak se dostat na co nejméně operací z jednoho místa do druhého

Příklad: tři body v bludišti



- cíl: najít nejkratší spojnici tří bodů

Příklad: bludištěm s dynamitem



1. co nejméně použití dynamitu
2. co nejkratší cesta

Příklad: číselné bludiště

start

zadání:

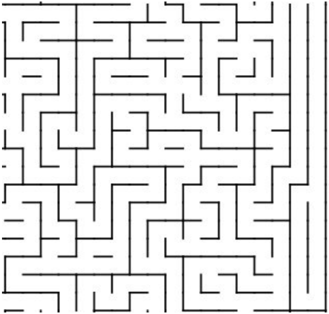
2	4	4	3	3
2	3	3	2	3
3	2	3	1	3
2	2	3	2	1
1	4	4	4	cíl

řešení: ↓ → ↑ ← → ↓

Logická úloha: Misionáři a kanibalové

- 3 misionáři, 3 kanibalové
- řeka, 1 loďka (max 2 lidé)
- víc kanibalů jak misionářů na jednom místě $\Rightarrow$ problém
- (jen jeden misionář a jeden kanibal umí pádlovat)
- zkuste:
  - najít řešení
  - „najít" v tom graf

Příklad: generování bludiště



- bludiště $\sim$ graf
- generování bludiště pomocí randomizovaného DFS – prokopávání zdí

Složitější algoritmy nad grafy

- nejkratší vzdálenosti
- kostra grafu
- eulerovská cesta – domečkologie
- hamiltonovská cesta – problém obchodního cestujícího
- toky v sítích

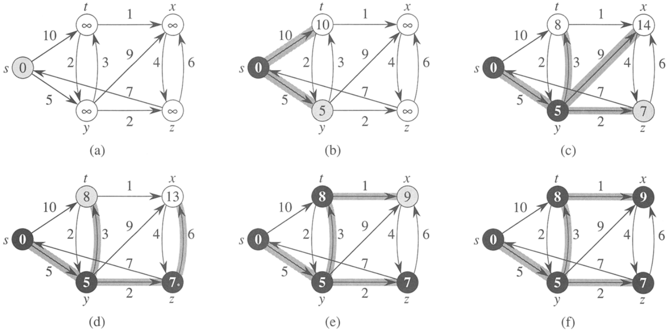
Nejkratší vzdálenosti

více různých problémů

- váhy hran:
  - konstantní (1)
  - přirozená čísla
  - celá čísla
- odkud kam:
  - z jednoho vrcholu do druhého (SSSP = single source shortest path)
  - mezi všemi dvojicemi vrcholů (APSP = all pairs shortest path)

Dijkstrův algoritmus

SSSP s kladnými hranami



T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein: Introduction to Algorithms.

Dijkstrův algoritmus

SSSP s kladnými hranami

- opakuj:
  1. vyber nezpracovaný vrchol s nejmenší vzdáleností od startu
  2. zpracuj vrchol: aktualizuj vzdálenost sousedů
- efektivní implementace: prioritní fronta

APSP s kladnými hranami

- naivně:
  - spustit SSSP z každého vrcholu
  - neefektivní
- Floyd-Warshalův algoritmus:
  - nejkratší vzdálenosti vedoucí přes vrchol 1
  - nejkratší vzdálenosti vedoucí přes vrcholy 1, 2
  - nejkratší vzdálenosti vedoucí přes vrcholy 1, 2, 3
  - ...

Floyd-Warshalův algoritmus

APSP s kladnými hranami

Algorithm 7.3 FLOYD

Input: An $n \times n$ matrix $l[1..n, 1..n]$ such that $l[i,j]$ is the length of the edge (i,j) in a directed graph $G = (\{1, 2, \ldots, n\}, E)$.

Output: A matrix D with $D[i,j] =$ the distance from i to j.

```
D ← l        {copy the input matrix l into D}
for k ← 1 to n
    for i ← 1 to n
        for j ← 1 to n
            D[i, j] = min{D[i, j], D[i, k] + D[k, j]}
        end for
    end for
end for
```

M. H. Alsuwaiyel: Algorithms, Design Techniques and Analysis.

Celočíselné hrany

- mohou být i záporné váhy hran, problémy:
  - nelze jednoduše najít ideální pořadí počítání vzdáleností
  - záporný cyklus
- obecný přístup:
  - opakovaná „relaxace“ hran
  - detekce záporných cyklů

Kostra grafu

- kostra grafu = minimální množina hran, tak že graf je souvislý
- ceny hran $\rightarrow$ nejlevnější kostra grafu
- aplikace: např. elektrická síť
- historie: prof. Borůvka

Kruskalův algoritmus

- setřídit hrany podle ceny
- postupně procházet hrany: vytvoří hrana cyklus?
  - ano $\rightarrow$ zahodit
  - ne $\rightarrow$ použít
- datová struktura: union-find

Kruskalův algoritmus

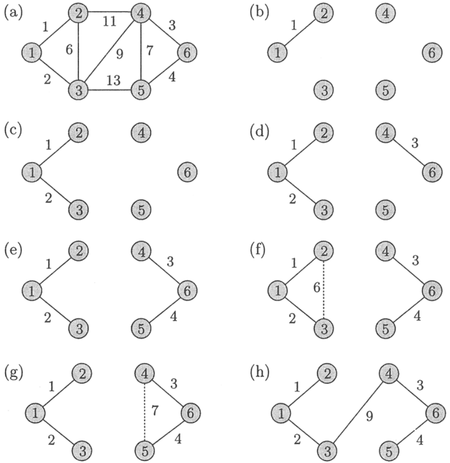
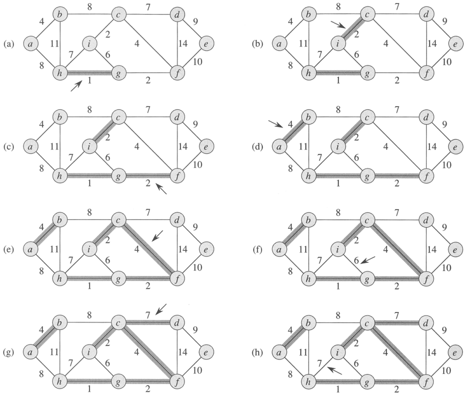


Fig. 8.4 An Example of Kruskal Algorithm.

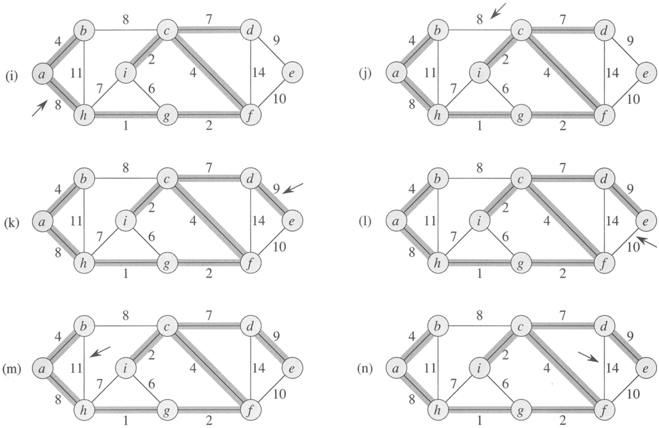
M. H. Alsuwaiyel: Algorithms, Design Techniques and Analysis.

Kruskalův algoritmus



T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein: Introduction to Algorithms.

Kruskalův algoritmus



T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein: Introduction to Algorithms.

Kruskalův algoritmus

Algorithm 8.3 KRUSKAL

Input: A weighted connected undirected graph $G = (V, E)$ with n vertices.

Output: The set of edges T of a minimum cost spanning tree for G.

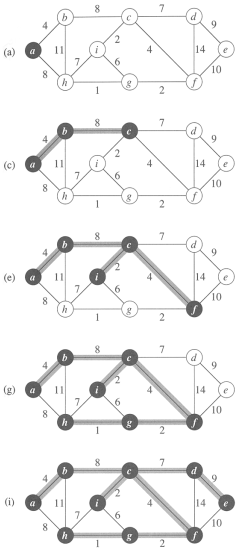
```
Sort the edges in E by nondecreasing weight.
for each vertex v ∈ V
    MAKESET({v})
end for
T = {}
while |T| < n − 1
    Let (x, y) be the next edge in E.
    if FIND(x) ≠ FIND(y) then
        Add (x, y) to T
        UNION(x, y)
    end if
end while
```

M. H. Alsuwaiyel: Algorithms, Design Techniques and Analysis.

Primův algoritmus

- začít od nejlevnější hrany
- postupně „přilepujeme“ další sousedící hrany
- datová struktura: prioritní fronta

Primův algoritmus



T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein: Introduction to Algorithms.

Primův algoritmus

Algorithm 8.4 PRIM

Input: A weighted connected undirected graph $G = (V, E)$, where $V = \{1, 2, \ldots, n\}$.

Output: The set of edges T of a minimum cost spanning tree for G.

```
T ← {};   X ← {1};   Y ← V − {1}
for y ← 2 to n
    if y adjacent to 1 then
        N[y] ← 1
        C[y] ← c[1, y]
    else C[y] ← ∞
    end if
end for
for j ← 1 to n − 1        {find n − 1 edges}
    Let y ∈ Y be such that C[y] is minimum
    T ← T ∪ {(y, N[y])}       {add edge (y, N[y]) to T}
    X ← X ∪ {y}               {add vertex y to X}
    Y ← Y − {y}               {delete vertex y from Y}
    for each vertex w ∈ Y that is adjacent to y
        if c[y, w] < C[w] then
            N[w] ← y
            C[w] ← c[y, w]
        end if
    end for
end for
```

M. H. Alsuwaiyel: Algorithms, Design Techniques and Analysis.

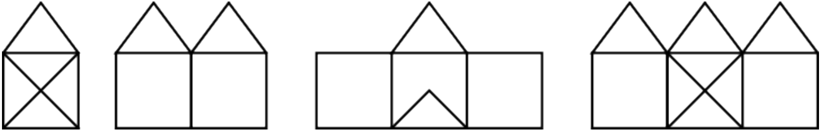
Nečekaná aplikace: generování bludiště

- další způsob jak generovat bludiště
- budujeme „kostru“ – bouráme zdi
- Kruskal, Prim – každý trochu jiná bludiště

Toky v sítích

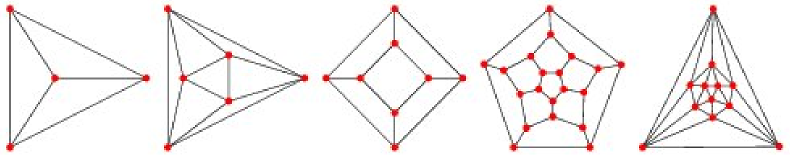
- tok v síti: např. voda, elektřina, energie (potravní řetězec)
- uzly: zdroj, dřez
- hrany: maximální kapacita
- podmínky:
  - zachování kapacity
  - konzistence uzlů: co přiteče, to odteče
- problém: hledání maximálního toku
- řešení: postupné „fecování“ toku

Eulerovský cyklus



- eulerovský cyklus = navštívit všechny hrany právě jednou
- problém 1: existuje eulerovský cyklus?
- problém 2: vypsat cyklus
- oboje jednoduše řešitelné

Hamiltonovský cyklus



- hamiltonovský cyklus = navštívit všechny uzly právě jednou
- problém obchodního cestujícího – vážené hrany
- obtížné (NP-úplné), hrubá síla, heuristiky

Shrnutí

- graf, reprezentace
- prohledávání do hloubky, do šířky
- aplikace prohledávání
- složitější algoritmy: cesty, kostry, toky, cykly, ...