C2142 Návrh algoritmů pro přírodovědce

7. Grafy.

Tomáš Raček

Jaro 2021

Vyhledávání v databázích I

Opakování. Umíme efektivně vyhledávat a řadit objekty podle různých klíčů v případě, že je na těchto klíčích definováno uspořádání ($\leq$).

Vyhledávání molekul. Mějme molekulu a chtějme zjistit, zdali se již vyskytuje v dané sadě sloučenin (= databázi).

- Záznamy o molekulách často obsahují jednoznačné identifikátory (řetězce znaků) $\rightarrow$ umíme.

Problém. Tyto informace ale nemusí být dostupné. K dispozici máme však minimálně:

- údaje o atomech (pozice, typy)
- vazby mezi atomy

Příklad molekuly – formát MOL

```
702
  -OEChem-03301510303D

  9  8  0     0  0  0  0  0  0999 V2000
   -1.1712    0.2997    0.0000 O   0  0  0  0  0  0
   -0.0463   -0.5665    0.0000 C   0  0  0  0  0  0
    1.2175    0.2668    0.0000 C   0  0  0  0  0  0
   -0.0958   -1.2120    0.8819 H   0  0  0  0  0  0
   -0.0952   -1.1938   -0.8946 H   0  0  0  0  0  0
    2.1050   -0.3720   -0.0177 H   0  0  0  0  0  0
    1.2426    0.9307   -0.8704 H   0  0  0  0  0  0
    1.2616    0.9052    0.8886 H   0  0  0  0  0  0
   -1.1291    0.8364    0.8099 H   0  0  0  0  0  0
  1  2  1  0  0  0  0
  1  9  1  0  0  0  0
  2  3  1  0  0  0  0
  2  4  1  0  0  0  0
  2  5  1  0  0  0  0
  3  6  1  0  0  0  0
  3  7  1  0  0  0  0
  3  8  1  0  0  0  0
```

Vyhledávání v databázích II

Intuice. Porovnání na základě pozic jednotlivých atomů a vazeb představuje výpočetně netriviální problém. Současné databáze navíc obsahují stovky tisíc až miliony struktur.

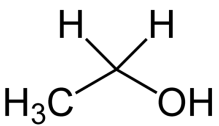
Návrh řešení. Proveďme vyhledávání v několika fázích, které budou postupně omezovat množinu přípustných struktur. Postupujme od nejjednodušších metod po složitější.

1. jednoduché deskriptory (př. sumární vzorec)
2. využití znalosti topologie, podstruktur
3. porovnání pozic atomů v prostoru

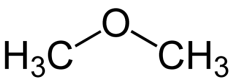
Příklad. Porovnání molekul podle sumárních vzorců v rozumném čase výrazně redukuje množinu kandidátů. Nicméně samo o sobě nestačí.

Vyhledávání v databázích III

Omezení. Pomocí sumárního vzorce nelze rozlišit izomery.



Ethanol (Alcohol)



Dimethylether

Topologie. Je nutné přidat další informace o struktuře sloučenin – propojení vazbami.

Problém. Potřebujeme nalézt vhodnou datovou strukturu pro reprezentaci molekuly.

3. fáze. Ani toto rozlišení obecně nestačí (stereoizomery), ale získané výsledky lze použít jako výchozí bod pro další algoritmy.

Graf

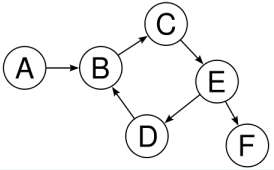
Definice. Graf $G = (V, E)$, kde V je množina uzlů (vrcholů) a E je množina hran.

Typy grafů

- orientovaný – hrany jsou uspořádané dvojice (u, v)
- neorientovaný – hrany jsou dvouprvkové podmnožiny $\{u, v\}$

Příklad orientovaného grafu

- $G = (V, E)$
- $V = \{A, B, C, D, E, F\}$
- $E = \{(A, B), (B, C), (C, E), (D, B), (E, D), (E, F)\}$



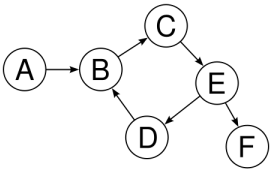
Reprezentace grafu

Minimální požadavky na datovou strukturu

- dotaz na existenci hrany v grafu
- sousedé daného vrcholu

Triviální řešení představuje obyčejný seznam (pole) hran. Nicméně výše zmíněné operace pak nelze implementovat efektivně.

- $G = (V, E)$
- $V = \{A, B, C, D, E, F\}$
- $E = \{(A, B), (B, C), (C, E), (D, B), (E, D), (E, F)\}$



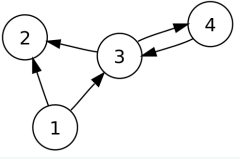
Matice sousednosti

Matice sousednosti. Vytvořme pro graf $G = (V, E)$ matici A o rozměrech $|V| \times |V|$ s vlastností:

$$A_{i,j} = 1 \leftrightarrow (i, j) \in E$$

Příklad

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$



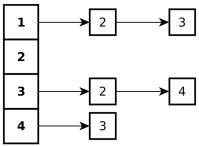
Vlastnosti

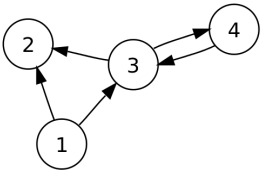
- dotaz na přítomnost hrany je konstatní operace
- seznam následníků daného vrcholu v lineárním čase
- potřeba $|V|^2$ paměti $\rightarrow$ vhodné pro husté grafy ($|E| \approx |V|^2$)

Seznam následníků

Seznam následníků. Uvažme pole ukazatelů na seznamy následníků daných vrcholů.

Příklad





Vlastnosti

- dotaz na přítomnost hrany je lineární operace
- seznam následníků v lineárním čase
- pouze $|V| + |E|$ paměti $\rightarrow$ vhodné pro řídké grafy ($|E| \approx |V|$)

Procházení grafu

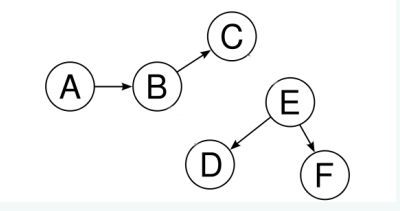
Cíl. Projít všechny vrcholy grafu dostupné ze zvoleného výchozího.

Naivní řešení. Projít postupně seznam vrcholů od začátku do konce (podobně jako u obyčejného pole).

- zjevně lineární operace
- nerespektuje strukturu grafu
- graf nemusí být souvislý → projdeme i jeho nedosažitelné části

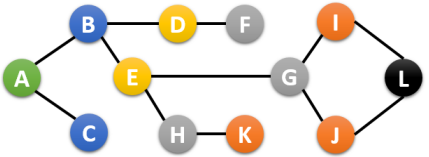
Ideální řešení

- zachová lineární složitost
- každý vrchol projde právě jednou
- odstraní výše uvedené nedostatky



Procházení do šířky

Breadth First Search (BFS) prochází graf po jednotlivých úrovních – než projde vrcholy vzdálené (co do počtu hran) n od výchozího, projde předtím všechny vrcholy vzdálené $n-1$.



Breadth-First Search (BFS)

Vlastnosti

- procházíme nejdříve všechny přímé následníky vrcholů
- pro uložení pořadí, ve kterém vrcholy prohledáváme, používáme frontu
- lineární složitost vzhledem k velikosti grafu – $\mathcal{O}(|V| + |E|)$

Procházení do šířky – pseudokód

```
1: function BFS(G,u) is
2:    Nechť Q je prázdná fronta
3:    Enqueue(Q,u)
4:    Označ u jako navštívený
5:    while Q není prázdná do
6:       v ← Dequeue(Q)
7:       for all (v,w) ∈ E do
8:          if w není navštívený then
9:             Označ w jako navštívený
10:            Enqueue(Q,w)
11:         fi
12:      done
13:   done
14: end
```

Procházení do hloubky

Depth First Search (DFS) prochází graf „dokud to jde“, pak se vrací do posledního místa, kde existuje neprozkoumaná cesta, kterou pak pokračuje dále (= obvyklé prohledávání bludiště).

Vlastnosti

- lineární algoritmus – $\mathcal{O}(|V| + |E|)$
- často v rekurzivní podobě, iterativní využívá zásobník

```
1: function DFS(G,u) is
2:    Označ u jako navštívený
3:    for all (u,v) ∈ E do
4:       if v není navštívený then
5:          DFS(G,v)
6:       fi
7:    done
8: end
```

Procházení do hloubky (iterativně) – pseudokód

```
 1: function DFS(G,u) is
 2:    Nechť S je prázdný zásobník
 3:    Push(S,u)
 4:    Označ u jako navštívený
 5:    while S není prázdný do
 6:       v ← Pop(S)
 7:       for all (v,w) ∈ E do
 8:          if w není navštívený then
 9:             Označ w jako navštívený
10:             Push(S,w)
11:          fi
12:       done
13:    done
14: end
```

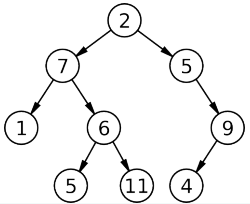
Otázka. Čím se liší pseudokód pro BFS a DFS?

Procházení binárního stromu

BFS → procházení po úrovních

Varianty DFS

- pre-order → 1. uzel, 2. levý podstrom, 3. pravý podstrom
- in-order → 1. levý podstrom, 2. uzel, 3. pravý podstrom
- post-order → 1. levý podstrom, 2. pravý podstrom, 3. uzel



Pořadí procházení vrcholů

- BFS: 2, 7, 5, 1, 6, 9, 5, 11, 4
- DFS pre-order: 2, 7, 1, 6, 5, 11, 5, 9, 4
- DFS in-order: 1, 7, 5, 6, 11, 2, 5, 4, 9
- DFS post-oder: 1, 5, 11, 6, 7, 4, 9, 5, 2

Otázka. Co kdybychom použili DFS in-order na BST?