Algoritmy & programovací techniky

11. června 2005

1 Složitost

- velikost vstup. dat, 1 krok algoritmu (op. v konst. čase)
- zrychlení výpočtu v záv. na rychlosti HW
- Asymptotická složitost - $f(n)$=$O(g(n))$, $f(n)$=$\Omega(g(n))$, $f(n)$= $\Theta(g(n))$, & o, w.

2 Dyn. množiny

- def. dyn. množiny, operace - find, insert, delete, min, max, succ, pred

2.1 Bin. vyhl. stromy

- definice bin. stromu, bin. vyhl. stromu
- všechny operace
- nevýhoda: negarantuje výšku

2.2 Červeno-černé stromy

- interní, externí uzly
- 4 vlastnosti:
  1. ∀uzel červený/černý
  2. ∀ext. uzel černý
  3. ∀červený vrchol musí mít oba syny černé
  4. ∀cesta od lib. vrcholu x k listu v jeho podstromě obs. stejný počet černých uzlů.
- výška uzlu, černá výška
- **L1: Nechť x je lib. uzel, pak jeho podstrom obs. aspoň** $2^{bh(x)}-1$**interních uzlů.** (indukcí podle h(x))
- **L2: Č-Č strom s n interními uzly má výšku nejvýše** $2\log_2(n+1)$**.** (vl. 3, lemma 1).
- garantovaná složitost
- rotace - levá, pravá; kořen vždy černý
- vkládání - jako BVS, pak obarvit nový červeně (když je otec červený - err), upravit:
  1. otec a strýc červený - přebarvit, přenos chyby
  2. strýc černý - je pravým synem ⇒L, převést na 3.
  3. strýc černý, je levým synem ⇒P, přebarvit
- delete - jako BVS, pak (vyhodím y - má max. 1 syna x, černý) x:= dvojitě černý, úpravy:
  1. bratr x (w) je červený (má 2 černé syny) - L, přebarvit, převést na 2,3,4
  2. w má 2 černé syny - odstranit černou z x, w →červený, A - černý nebo 2x černý (přenos chyby)
  3. levý syn w červený (B), pravý černý - vyměnit barvy B a w, P(w) převést na 4.
  4. pravý syn w (C) je červený -L(w), přebarvit - C černé, w(nový kořen) buď černý nebo červený

2.3 AVL stromy

- definice
- **V1: Výška AVL stromu s n vrcholy je O(log n)**. - nechť minimální strom, velikost roste jako fib. - lemma : AVL ≥fib, důkaz indukcí, z toho pro obecný AVL.

3 Dolní odhady složitosti problémů

- horní - jednodušší, třeba dokázat neexistenci rychlejšího alg.
- triviální odhady - velikost vstupu, výstupu
- **V1: Na určení k-tého z n prvků je třeba aspoň (n-1) porovnání.**
- **V2: Pro ∀ tříďák zal. na porovnávání ex. vstupní posloupnost, t.ž. alg. provede** $\Omega(n\ log_2 n)$**porovnání.** (rozhodovací strom, n! listů, $n! \leq 2^p$(p pater), odhad faktoriálu: $(\frac{n}{2})^{\frac{n}{2}}$.

4 Alg. „Rozděl a panuj“

- fungování - rekurze: rozděl, vyřeš podúlohy, sjednoť
- mergesort, binsearch

4.1 Analýza složitosti

- D(n) - rozdělení **n** na **a** podúloh velikosti $\frac{n}{b}$, T(n) - zprac. úlohy vel. n (pro n<c: T(n) = O(1)), S(n) - sjednocení na řeš. pův. úlohy vel. n.
- vzorec $T(n)= D(n) + aT(\frac{n}{b}) + S(n)$ (pro n<c, jinak $T(n)= \Theta(1)$).
- metody řešení - substituční, master theorem:
  - Substituce: uhodnout řešení (asymptotické), indukcí dokázat (zvlášť pro horní a dolní odhad).
  - **Master Theorem: nechť a≥1, c>1, d≥0** $\in R$**, nechť** $T: N \to N$ **je nekl. fce, t.ž. ∀n (n =** c^k**, k**$\in N$**) platí:**

$$T(n) = aT(\frac{n}{c}) + F(n)$$

kde $F(n) = \Theta(n^d)$**. Nechť x:=** $log_c a$**. Potom:**

1. x < d: $T(n) = \Theta(n^d)$.
2. x = d: $T(n) = \Theta(n^d \log n)$
3. x > d: $T(n) = \Theta(n^x)$.

4.2 příklady :

- násobení komplexních čísel - $((ac-bd), (ad+bc)i)$ - c(b-a), a(d+c), d(a-b)
- násobení matic (normální rekurzivní, Strassenův algoritmus).
- hledání k-tého z n prvků
  - lze setřídit - $\Theta(n \log n)$
  - rozděl &panuj - chceme lépe: z quicksortu: pivot, roztřídit na m prvků menších než pivot, pivot a (n-m-1) větších. (při špatném výběru pivota - kvadratická složitost).
- **Blum et al.** - dobrý výběr pivota:
  1. rozděl posl. na $\frac{n}{5}$pětic
  2. ∀ pětici najdi medián (7 porovnání: 4; vítězové ven, přidám pátý, každý s každým)
  3. rekurzivně najdi medián z mn. $\frac{n}{5}$ mediánů
  4. použij ho jako pivot k rozdělení posl. (obě poloviny zaručený počet prvků $\frac{3}{10}n$.
  5. pokud medián mediánů není hledaný prvek, hledej rekurzivně v mn. větších / menších prvků

  - důkaz subst. metodou, T(n) $\leq \frac{12}{5}n + k \cdot \frac{9}{10}n$ (a chci) $\leq k \cdot n$

- analýza složitosti Quicksortu
  - nejlepší, nejhorší, **průměrný** případ: (intuitivně - konst. poměr), korektně: předp. stejnou pravděpodobnost ∀permutací, pivot první prvek úseku, dělení zachovává náhodnost. Z toho: $T(n) = \frac{1}{n}\sum_{i=0}^{n-1}(T(i) + T(n-1-i)) + (n-1)$, sečíst vždy 2 spolu, vyjádřit pomocí T(n-1) - $T(n) \cdot n$ & $T(n-1) \cdot (n-1)$ odečíst; vypsat rovnice pro n...1 a sečíst. Vyjde $2(n+1)\sum_{j=2}^{n}\frac{j-1}{(j+1)j}$, rozepsat jako $\frac{A}{j} + \frac{B}{j+1}$, spočíst, rozepsat, vyjde $2(n+1) \cdot [\frac{2}{n+1} + \sum_{j=3}^{n}\frac{1}{j} - \frac{1}{2}]$ - harm. řada ~ $n \cdot \log n$.

5 Třídění v lin. čase

- není zal. na porovnávání, zpravidla indexuje pole tříděnými čísly ⇒omezení hodnot tříděných čísel
- **Counting sort** - n čísel ∈ [1..k], k = $O(n)$. 2 pole A,B &1 pomocné - C, 1. init C (nulování), 2. do C dám #vstup. čísel rovných i, 3. v C je #čísel ≤i, 4. do B (downto) vkládám na pozice určené v C prvky A (stabilní).
- **Radix sort** - podle nižšího, pak vyšších řádů; stabilní průchody, 1 řád - jakýkoliv alg. (často Counting - pak lineární).

6 Zákl. grafové algoritmy

- graf, vrcholy (n), hrany (m), orientovaný graf
- reprezentace: matice sousednosti, seznam sousedů (spoják)

6.1 Prohledávání do šířky - BFS

- po vrstvách podle vzdálenosti; používá FIFO
- zač - obarví vše bíle, předch. := NIL, vzdál. := ∞; do fronty dá 1. vrchol (obarvený šedě, vzdál. := 0).
- dokud není prázdná fronta - vyberu vrchol, vezmu sousedy (bílé), přebarvím šedě, dám jim vzdál. + 1, nast. předchůdce, dám je do fronty. Vrchol přebarvím na černo a vyhodím z fronty.
- užití - Dijkstr, Prim, testování souvislosti, počítání komponent; běží v $\Theta(m+n)$

6.2 Prohl. do hloubky - DFS

- používá zásobník/rekurzi
- ∀vrchol - projdu sousedy, na nenavštívené rekurzivně volám „navštiv"; ukládám čas-navšť (d(i)).
- **klasifikace hran:** stromová (j objeven z i - bílý), zpáteční (j předchůdce i - šedý), dopředná (i předch. j, ale ne přímý rodič - j černý, d(i)<d(j)), příčná (zbytek - j černý, d(i)>d(j)).
- **vlastnosti:** 1. stromové hrany tvoří orientovaný DFS les (ukázat že nemohou tvořit orient. cyklus, orient. vidličku); 2. j následník i v DFS stromě ⇔v čase d(i) ∃z i do j cesta z výlučně bílých vrcholů (⇒: indukcí; ⇐ex. cesta - stane se cestou ze strom. hran), 3. intervaly (d(i), f(i)) tvoří „dobré uzávorkování" (z rekurzivnosti)

6.3 Topologické číslování

- def. topolog. čísl. ($t: V \to \{1...n\}$, t.ž. $\forall(i,j): t(i) < t(j)$.) , pouze acyklické.
- hloupý alg. - najdi vrchol bez výst. hran, přiřaď posl. volné číslo; odstraň ho ze seznamu vrcholů a opakuj. ($\Theta(n(m+n))$.
- lepší - modifikace DFS (lin.čas):
  - **lemma** - v G cyklus ⇔DFS objeví zpět. hranu (⇒vezmu i z cyklu objevené jako první, j předchůdce i v cyklu, (j,i) se stane zpět. hranou; ⇐z def. zpět. hrany)
  - **věta: Očíslování vrcholů acyklického grafu podle klesajících časů opuštění f(i) je topologické** (stačí dokázat: (i,j) ∈E ⇒ f(i)>f(j) - podle barvy j při prohlížení (i,j) - j bílý - stromová hrana, j šedý - lemma, spor, j černý - f(j) už nastalo)

6.4 Transitivní uzávěr orient. grafu

- definice: G' = (V,E') trans. uzávěr G = (V,E) $\Leftrightarrow \forall i,j \in V,\ i \neq j$: z i do j vede v G orientovaná cesta $\Rightarrow (i,j) \in E$.
- reprezentace maticí sousednosti $\rightarrow$ matice dosažitelnosti
- lze získat v $\Theta(n(m+n))$ pomocí n-krát použitého DFS.

6.5 Silně souviské komponenty orient. grafu

- def. $K \subseteq V$ je s.s.k. G, pokud (i.) $\forall i,j \in K, i \neq j$: $\exists$ *orient.* cesta z i do j i zpět, (ii.) neexistuje množina vrcholů, která by byla ostrou nadmnožinou a splňovala (i).
- hloupý alg. - vytvořit matici dosažitelnosti a z ní v čase $O(n^2)$ přečíst s.s.k.. Složitost stejná jako transitivní uzávěr.
- **lineární alg.** - „kondenzovaný" graf - jeho vrcholy jsou souvislé komp. původního; prakticky - posl. opuštěný vrchol DFS leží v topolog. první komponentě:
  1. DFS(G), vytv. spojáku vrcholů podle klesajicích časů f(i).
  2. Vytvoření G^T(G i G^Tmají stejné s.s.k., vytvoření v $O(m+n)$- procházím spojáky G a přidávám zdroj. vrcholy do spojáků G^T).
  3. DFS(G^T), kdy vrcholy jsou hl. cyklem procházeny v pořadí podle fáze 1. (vždy dostanu vrchol z topologicky první komp. G (topolog. poslední komp. v G^T)
- celý běží v $\Theta(m+n)$
- důkaz:
  - lemma: nechť K je s.s.k. v G, po DFS(G) platí: (i) K je podmn. jediného DFS stromu, (ii) v daném DFS tvoří K podstrom ((i) - T_j, T_i - nechť $T_i$vybudován dřív než T_j, potom příčné hrany vedou jen z $T_j$do T_i. (ii) - a)neex. spol. předek $\forall$vrcholů v K, který je z K - nemůže nastat, viz (i). b) musí tvořit podstrom (cesta zač. i končící v K musí celá patřit do K))
  - ve fázi 1 - nějaké DFS stromy. fáze 3 - další DFS stromy: $T_1, T_2, ...$ Indukcí podle i - T_i tvoří s.s.k.. Nechť x kořen T_1, potom je posl. opuštěný ve fázi1, kořen posl. stromu z fáze1. y lib. $\in T_1$. Z x vede cesta do y v $G^T \Rightarrow$ z y vede cesta do x v G. y nemůže být v G v jiném DFS stromě než x (z vl. DFS), z x musí vést cesta do $y \Rightarrow x \& y$ jsou ve stejné s.s.k.. Tato s.s.k. obsahuje celý T_1, ale nic jiného (prohledávám $\forall$ možné cesty - z ost. vrcholů nevede cesta do x).
    Dle lemmatu T_1 tvoří kořenový podstrom v posl. stromě z fáze1. Po jeho odstranění se tento strom rozpadne na množinu stromů. Kořen T_2 je nyní opět kořen posl. stromu z fáze1 $\Rightarrow$ opak. pro $\forall$ T_i.

7 Extremální cesty v orient. grafu

- graf. bez ohodnocení - stačí BFS.
- def. ohodnoceného grafu - $w : E \rightarrow R$ - váhová funkce. $w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$, kde p je orient. cesta
- def. váha nejkr. cesty - $\delta(u,v) = min\{w(p) |\ p$ je cesta z u do $v\}$ pokud $\exists$cesta, jinak $\delta(u,v) = \infty$.
- nejkr. cesta; záporné cykly, dodef. $\delta(u,v) := -\infty$.

7.1 Nejkr. cesty z jednoho zdroje

- úloha: spočítat pro dané s$\in$V $\delta(s,v)$ pro $\forall v \in V \backslash \{s\}$.
- **Vl.1.** je-li p nejkr. cesta z v_0 do v_k, pak $\forall i,j :\ 0 \leq i,j \leq k$ je p_{ij} nejkr. cesta (sporem - mohu nahr. kratší)
- **Vl.2.** je-li p nejkr. cesta a navíc lze p rozložit na $p' = p_{su} + (u,v)$, pak $\delta(s,v) = \delta(s,u) + w(u,v)$.
- **Vl.3.** $(u,v) \in E \Rightarrow \delta(s,v) \leq \delta(s,u) + w(u,v)$
- zpřesňování odhadů - $d(v)$, $d(v) \geq \delta(s,v)$; inicializace $(+\infty)$, Relax(u,v)

- **Vl.4.** $(u,v) \in E$ $\Rightarrow$po provedení Relax(u,v) platí $d(v) \leq d(u) + w(u,v)$ (z vl. Relax, neplatí-li, nastaví rovnost)
- **Vl.5.** Po Init je $d(v) \geq \delta(s,v)$, platí stále po lib. posl. volání Relax. Pokud dosáhne $\delta(s,v)$, tak už se nemění. (vím že platí po Init, sporem - v 1., pro které Relax poruší $d(v) \geq \delta(s,v)$. Pak po Relax(u,v) platí $d(u) + w(u,v) = d(v) < \delta(s,v) \leq$(vl.3) $\delta(s,u) + w(u,v)$ - spor).
- **Vl.6.** Pokud z s do v nevede orient. cesta, tak od Init platí, že $d(v) = \delta(s,v) = \infty$ (z vl. 5)
- **Vl.7.** Nechť nejkr. cesta z s do v končí hranou (u,v). Po provedení Init a sekvence Relax, obs. Relax(u,v), tak pokud d(u) = $\delta(s,u)$ platí někdy před voláním Relax(u,v), tak d(v) = $\delta(s,v)$ platí kdykoliv potom. (vl.5 - d(u)=$\delta(s,u)$ platí stále. Potom $d(v) \leq$(vl.4)$d(u) + w(u,v) = \delta(s,u) + w(u,v) =$(vl.2)$\delta(s,v)$.)

7.2 DAG

- Directed Acyclic Graph, Alg. kritické cesty. (jen pro acyklické grafy)
  1. topolog. setřiď vrcholy G
  2. Inicializace
  3. $\forall(u \in V)$ v topolog. pořadí udělej pro $\forall(v \in V; (u,v) \in E)$ Relax(u,v).
- **V1: Alg. je korektní** - v lib. a) nedosažitelný - vl.6; b) nechť p je nejkr. cesta z s do v. Platí, že t(v_i) < t(v_{i+1}), Relax topologicky - indukcí podle i: $d(v_i) = \delta(s,v_i)$ (ind. krok = vl.7).
- složitost $\Theta(n+m)$ topolog. třídění, alg. trvá $\Theta(1)$ na vrchol i hranu (předp. přístupu k vrcholu v konst. čase).
- aplikace - plánování činností - cesta: posl. prováděných jedna po druhé

7.3 Dijkstrův alg.

- nezáp. váhy hran; vrcholy - 2 množiny: vyřízené S, nevyřízené Q
  1. Init
  2. S:=$\emptyset$, Q := V(G)
  3. while (Q$\neq \emptyset$) do u:= Extract-Min(Q); S:= S$\cup\{u\}$; $\forall(v \in V, (u,v) \in E)$ do Relax(u,v).
- **V2: ALg. je korektní** - sporem, nechť u je 1. vrchol, t.ž. $d(u) \neq \delta(s,u)$ v čase přeřazení z Q do S. $u \neq s$ - Init, u dosažitelný (vl.6); nechť p nejkr. cesta z s do u, y 1. vrchol na p mimo S, x jeho předch. Nezáp.hrany - $\delta(s,y) \leq \delta(s,u)$. nejkr. cesta - $d(y) = \delta(s,y)$. Podle výběru - $d(u) \leq d(y)$, spor - $d(u) = \delta(s,u)$ (musí být rovno).).
- čas. složitost - Pole: Extract-Min (n^2), Init (n), Relax (max. m-krát) - m, $m < n^2$. Celkem $\Theta(n^2)$. Halda: Init - $n \log n$, Extract-Min $n \cdot \log n$, Decrease-Key $m \cdot \log m$, Celkem $\Theta((m+n)\log n)$.

7.4 Bellman-Ford

- nejobecnější, nejpomalejší, Relax i víckrát na 1 hranu, dovoluje záp. cykly (vrací false)
  1. Init
  2. for i:= 1 to (|V| - 1) do for $\forall$ ((u,v)$\in$E) do Relax (u,v);
  3. for $\forall$((u,v)$\in$E) do if (d(v) > d(u) + w(u,v)) then return FALSE;
  4. return TRUE;
- složitost - $\Theta(n \cdot m)$
- **L1: Nechť G, s. Předp., že G neobs. záp. cykly dostupné z s. Pak v době ukončení práce Bellman-Ford platí $d(v) = \delta(s,v)$ pro $\forall$vrcholy dosažitelné z s.** - Nechť v dosažitelný, $p = (v_0, ..., v_k)$ nejkr. cesta. Nezáp. cykly - p neobs. cyklus, proto $k \leq (n-1)$. Indukcí podle k dokázat, že po k-té iteraci platí $d(v_k) = \delta(s,v_k)$.
- **V3: Nechť G, s. Pak po ukončení B-F: a) pokud obs. záp. cyklus, vrátí FALSE. b) jinak alg. vrátí TRUE,** $\forall v : d(v) = \delta(s,v)$**.** a) $v_0, ..v_k$ záp. cyklus, předp. TRUE - $d(v_i) \leq d(v_{i-1}) + w(v_{i-1}, v_i)$ $\Rightarrow \sum_{i=1}^{k} d(v_i) \leq \sum_{i=1}^{k} (d(v_{i-1}) + w(v_{i-1}, v_i)) \Rightarrow 0 \leq \sum_{i=1}^{k} w(v_{i-1}, v_i)$. b) dosažitelné - L1, nedosaž. - Vl.6, kontrola na konci (Vl.3).

7.5 Nejkr. cesty pro všechny páry vrcholů

- předp. zadání maticí sousednosti W^G_{uv} =(0 pro u=v) (w(u,v) pro (u,v)∈E) (∞ pro (u,v)∉E), předp. nezáp. cykly, cíl - konstrukce D^G, kde $D_{uv} = \delta(u, v)$.
- možno n-krát spustit předch. alg.
- **Alg. „násobení matic“**
  - indukcí podle počtu hran na nejkr. cestě. $d_{uv}(k)$ - nejkr. cesta s max. k hranami
    1. k=1 ... $d_{uv}(1) = W^G_{uv}$, matice $D^G(1) = W^G$.
    2. k-1 →k ... nejkr. cesta vede přes mezivrchol (cesta z u do i & (i, v) - násobím vždy $D_G(1)$) - maticové nás (sčítání místo násobení, výběr minima místo sčítání). (pokud je nejkr. kratší než k hran, zůstane zachována - i=v).
  - bez záp. cyklů - nejkr. cesta max. (n-1) - další „násobení“ nevadí.
  - složitost - asociativně $\Theta(n^3 \log_2 n)$.
- **Floyd-Warshallův algoritmus**
  - podobně, ale indukce podle *množiny povolených* vrcholů jako mezivrcholy.
  - $d_{u,v}$= min. váha cesty z u do v s vnitř. vrcholy z množiny $\{1..k\}$
    1. k=0 - $D^G = W^G$.
    2. k-1 → k - $d_{uv} = min\{d_{uv}(k-1),\ d_{uk}(k-1) + d_{kv}(k-1)\}$ - stačí porovnání 2 čísel.
  - složitost - $\Theta(n^3)$, navíc malá konstanta (jen 3 for-cykly), rychlejší než B-F; pro záp. cykly se na diagonále za čas objeví záp. číslo - netřeba testovat předem.

8 Minimální kostra

- vstup - orientovaný G, váhová fce; úkol - nalézt kostru s min. celk. vahou; platí |T| = |V|-1, BÚNO nezáp. hrany (lze přičíst konstantu)
- idea - post. přidávat hrany, do mn. A, která je pořád podmn. nějaké min. kostry.
- bezpečná hrana ($A \cup \{e\}$je taky podmn. kostry), řez (rozklad V na 2 části $S, V\backslash S$), hrana kříží řez (|{$(u, v) \cup S$|=1), řez respektuje mn. A (žádná hran nekříží řez), lehká hrana pro řez (ze všech hran křících řez má nejmenší váhu).
- **V1: Nechť G,w. Nechť A⊆E podmn. min. kostry,** $(S, V\backslash S)$ **řez resp. A. Potom pokud** $(u, v) \in E$ **lehká pro řez, pak bezpečná pro A.** Nechť T min. kostra, $A \subseteq T$, pokud $(u, v) \in T$ - OK; jinak - p jediná cesta z u do v v T, (u, v) kříží řez - na cestě je hrana $(x, y) \notin A$ co kříží řez. T' = $\{T\backslash(x, y) \cup (u, v)\}$; w(T)=w(T').
- Důsledek - C souv. komp. podgrafu zadaného A. Potom pokud (u,v) je hrana min. váhy, spojující C s jinými komp. podgrafu zadaného A, pak je (u,v) bezpečná pro A. (Řez (C,V\C) resp. A, (u,v) lehká).

8.1 Borůvka - Kruskal

- vždy hrana s nejm. váhou ze všech hran mezi stávajícími komp. podgrafu zadaného A; vždy sloučí 2 komp. v 1 strom
  1. setřiď hrany podle w; $A := \emptyset$
  2. $\forall v \in V$:Make-Set(v) {samost. komp.}
  3. $\forall (u, v) \in E$:(v pořadí podle třídění) if Find-Set(u) ≠Find-Set(v) then A:=A $\cup (u, v)$; Union(u,v)
  4. return A.
- Složitost - Setřídění $\Theta(m \log m)$; Make-Set - $\Theta(n)$; Find-Set - konst. čas (pole pointrů, z něj přes vrchol na hlavu seznamu - číslo mn.) - $\Theta(m)$; Union - kratší seznam za delší, přepsat aliasy, délku sezn. Max. přealiasování - $\log_2 n$ pro 1 vrchol - $O(n \log n)$.

8.2 Jarník - Prim

- hrany A vždy 1 strom, vždy vybere hranu s nejm. vahou ze všech, které vedou mezi A a okolím

1. Q:= V(G); for $\forall v \in V$ od klíč(v):= ∞; klíč(r) := 0; (start. vrchol) předch(r):=NIL;
2. while (Q$\neq \emptyset$) do
u:= Extract-Min(Q); for$\forall(v \in V : (u,v) \in E)$ do if ($v \in Q$ && klíč(v)> $w(u,v)$) then klíč(v):=w(u,v); předch(v):=u;

- Složitost - v poli $\Theta(n^2)$ - vybr. minima - n-krát n; v bin. haldě: Init - $\Theta(n)$; Extract-min - $n \cdot \log n$; Nejvýše m-krát decrease-key - $O(m \log n)$; Celkem nejhůře $\Theta(m \log n)$.

9 Hashování

- vhodné pro reprezentaci dyn. množiny, kde potřebuji jen Insert, Delete, Search
- přímo adresovatelná tabulka - pole - velikost: počet možných klíčů, předp. různých klíčů pro všechny prvky; buď celá data nebo pointry; neefektivní.
- hash-tabulka - menší, úměrná skutečnému počtu použitých klíčů, počítá se pomocí **hashovací funkce**.
- problém - **kolize** - řešení: řetězení - ve spojáku. (Insert & Delete - $\Theta(1)$, Search: jednoduché uniformní hashování (stejná pravděpodobnost $\forall$adresu, nezáv. na ost. klíčích) - faktor zaplnění tabulky - $\alpha = \frac{n}{m}$.)
- **V1:** Neúsp. Search trvá průměrně $\Theta(1+\alpha)$. - spočítám hash, projdu celý spoják (prům. α prvků).
- **V2:** Úsp. hledání trvá průměrně $\Theta(1+\alpha)$ za předp. viz výše. - Předp. že Insert strká vše na konec spojáku, že hledám od zač.; Očekávaný počet navštívených je o 1 větší než při vložení prvku. Průměr přes všech n uložených: (1 + očekávaná délka seznamu při vložení i-tého): $(1+\frac{i-1}{m})$. Průměr: $1+\frac{n-1}{2m} = 1+\frac{\alpha}{2}+\frac{1}{2m} = \Theta(1+\alpha)$.
- Důsledek: Pokud $n = O(m)$, tak $\alpha = O(1)$ - Search je průměrně $\Theta(1)$.

9.1 Hash-funkce

- Chceme co nejrovnoměrnější rozdělení, předp. číselné klíče (stringy - konverze)

1. Hashování dělením
   - hash(k) = k mod m
   - nevhodné m=2^p, 10^p, $2^p - 1$, vhodné - m prvočíslo daleko od mocnin 2
2. Hashování násobením
   - volím $A \in (0,1)$, Pro klíč k spočítám $\lfloor m \cdot (kA \, mod \, 1) \rfloor$
   - m většinou použitá mocnina 2 - jednouší počítání - m=2^p, k - w bitů - vynásobení k(A shl w), vezme se z dolní poloviny výsledku prvních p bitů (prvních p bitů desetiné části kA).
   - vhodné A - např. $\frac{\sqrt{5}-1}{2}$.
3. Univerzální hashování
   - ke každé fci se dá zkonstruovat posloupnost, kde se n klíčů nahashuje do stejného místa (když universum $\geq n^2$). Možnost obejít - random vybrat z více hash-fcí - nelze vytvořit špatná data.
   - volíme náhodně a nezávisle z vhodné množiny fcí
   - **univerzální** mn. hash-fcí H- pokud $\forall$2 klíče $x, y \in U$ platí, že počet hash-fcí, kde hash(x)=hash(y) je max. $\frac{H}{m}$.
   - Před hashováním zvolím náhodně h:$U \rightarrow \{1...m\}$z H (univ.) $\Rightarrow$pravděpodobnost kolize je $\frac{1}{m}$ - to je jednoduché uniformní hashování.
   - **V1: Nechť je h náh. vybraná z univ. mn. hash-fcí, nechť je použitá k hashování n klíčů do tabulky velikosti m, kde $n \leq m$, potom očekávaný počet kolizí libovolného klíče x je menší než 1.** (def. c_{yz} =(1 pro h(y)=h(z), 0 jinak; h z univ. mn. $\rightarrow E[c_{yz}] = \frac{1}{m}$, c_x :=celk. počet kolizí x. $E[c_x] = E[\sum_{y \neq x} c_{xy}] = \sum_{y \neq x} E[c_{xy}] = \frac{n-1}{m}$.
   - Důsledek: prům. délka seznamu je < 1.

- Jak zkonstruovat univ. množinu? - zvolit m prvočíslo, $\forall x$ rozdělit do $(r+1)$ částí ($x_i < m$); a náh. klíč, potom $h_a(x) = (\sum_{i=0}^{r} a_i x_i) \bmod m$. $H = \cup_a \{h_a\}$, $|H| = m^{(r+1)}$.
- **V2: Taková *H* je univ. množina hash-fcí.** (Nechť 2 různé klíče x,y se liší v k-tém podklíči. Pro každou posl. $a_0, ..., a_{k-1}, a_{k+1}, ..., a_r$ ex. právě 1 a_k, t.ž. $h_a(x) = h_a(y)$ - $a_k(x_k - y_k) \equiv= (-\sum_{i \neq k} a_i(x_i - y_i)) \bmod m$. Možných voleb posl. je $m^r, \forall \exists! a_k$, pro které nastává kolize.

9.2 Otevřené adresování

- při kolizi se spočítá nová adresa - další pokus o zápis do tabulky ($\forall$ klíče přímo v 1 tabulce). Hash-fce: $h : U \times \{0, .., m-1\} \to \{0, ..., m-1\}$. Pro klíč máme posloupnost $\{h(x,0), h(x,1) ... h(x,m-1)\}$- musí být permutací $\{0, ..., m-1\}$.

1. Lineární zkoušení - $h(x,i) = (h'(x) + i) \bmod m$ - problém - primární clusterování (jen m posloupností zkoušených pozic)
2. Kvadratické zkoušení - $h(x,i) = (h'(x) + c_1 i + c_2 i^2) \bmod m$ - musím vhodně zvolit c_1, c_2, (pořád ale jen m posloupností zkoušených pozic).
3. Dvojité hashování - $h(x,i) = (h_1(x) + i \cdot h_2(x)) \bmod m$ - nezávislé fce, potřebuji h_2 nesoudělné s m. Správně vybraná funkce - je-li $h_1(x) = h_1(y)$, pak většinou $h_2(x) \neq h_2(y)$ - $\Theta(n^2)$ posl. zkoušených pozic.