Písemka – 4 různé příklady – těžší než zápočtové; vymyslete variantu algoritmu, dokažte vlastnost, datové str.
1 ústní otázka – dokažte/ ukažte

Asymptotická notace

Časová (počet kroků ideálního stroje), prostorová složitost (obvykle počet bitů); závísí na velikosti vstupních dat
Odstranění závislosti na konkrétních datech – v nejhorším, v průměrném, v nejlepším případě
Asymptotická složitost – chování algoritmu na velkých datech; zanedbává multiplikativná a aditivní konst.
$\Omega(f) \leq \Theta(f) \leq O(f)$
Amortizovaná složitost – nepočítáme dobu jedné operace, ale průměrnou dobu operace

Binární vyhledávací stromy

Vrcholy obsahují klíč, pravého následíka a levého následníka, rodiče; klíč(l)≤klíč(v)≤klíč(p)
Operace: najít; minimum; maximum; otec; syn; vložit; odstranit; úměrné výšce stromu;

Červeno-černé stromy

Každý vrchol červený/ černý; každý list černý; červený vrchol má oba potomky černé; z každého vrcholu vede do všech jeho podřízených listů cesta přes stejný počet černých vrcholů
Výška maximálně 2 log(n+1)

AVL stromy

Pro každý uzel platí $|h(l)-h(r)| \leq 1$
Výška stromu je O(log$n$)

B-stromy

Klíče v neklesající posloupnosti; má-li vrchol n klíčů, má n+1 pointerů; klíče ve vrcholech rozdělují intervaly v podstromech; každý list ve stejné hloubce; pro $t \geq 2$ platí vrchol má t – 2t synů, jen kořen 2 – 2t
Výška stromu $h \leq \log_t[(n+1)/2]$?!?!?!
Varianty: t – 2t klíčů; všechna data v listech; provázané stromy – pointery na sousedy

Hašování

Přímo adresovatelné tabulky; pokud je velký rozptyl klíčů, je zbytečné mít tak velkou tabulku
Adresu budeme z klíče počítat pomocí nějaké funkce, problém, pokud pro dva klíče vychází stejná hodnota
Faktor naplnění – $\alpha = n/m$ počet prvků/ velikost tabulky; pevná cena operací O(1)

Volba hašovací funkce

Dobrá hašovací funkce splňuje předpoklady uniformního hašování – rovnoměrné rozházení do pole
Dělení – $h(k)=k \bmod m$; nevhodné pro $m=x^p$, vhodné pro prvočísla vzdálená od mocnin 2
Násobení – $h(k)=\lfloor m(kA \bmod 1) \rfloor$; A je z (0,1), R\Q; m je mocnina dvojky
Univerzální hašování – univerzum fcí, fci zvolíme náhodně a nezávisle na vstupních datech – randomizace
množina je univerzální, pokud pro všechna $x \neq y$ ke počet fcí, pro které $h(x)=h(y)$ je $|F|/m$
(# kolizí, kterých se účastní náhodně vybraný konkrétní klíč) < 1; dk: P(kolize xy) je $1/m$, $\sum=(n-1)/m$
konstrukce: zvolíme prvočíslo m; $\forall$ klíč rozdělíme na r+1 částí; $x_i<m$; zvolme $a_0, \ldots a_n \in \{0, \ldots, m-1\}$
$h_a(x)=(\sum a_i x_i) \bmod m$; dk: zvolme x, y, pro konkrétní a_i mimo j $\exists!$ 1 řešení $a_j(x_j-y_j) \bmod m = 0$ ...

Řešení kolizí

Zřetězením – V tabulce nejsou přímo prvky, ale pointery na spojové seznamy; vyhledávání průměrně $\Theta(1+\alpha)$
Otevřené adresování - Všechny prvky v tabulce; při kolizi počítáme náhradní adresu h:U×{0…m-1}→{0…m-1}
Lineární – $h(x,i)=[h(x)+i] \bmod m$; vznikají obsazené pásy (pravděpodobnost (i+1)/2); snadné *delete*
Kvadratické – $h(x,i)=[h(x)+c_1 i+c_2 i^2] \bmod m$; konstanty musí být vhodně zvoleny, jen m posloupností
Dvojité h. – $h(x,i)=[h_1(x)+i\ h_2(x)] \bmod m$; h_2 nesoudělné s m; $m=2^p$, h_2 liché nebo m prvočíslo, $h_2<m$
při neúspěšném hledání nejvíce $1/(1-\alpha)$ testování !!! naučit se důkaz !!! str. 17

Hašování do rostoucí tabulky

Řeší omezenou velikost tabulky a nedokonalé delete; periodická reorganizace tabulky – dosáhneme-li α=mez amortizovaně se neprojeví

Haldy

Operace vytvořit, vložit, minimum, odstranit minimum, snížit klíč, odstranit, sjednotit

Binární halda

Zcela zaplněný binární strom, kde otec je vždy menší než oba jeho synové

Binomiální halda

Spojový seznam binomických stromů, které jsou haldově uspořádány, různých velikostí, uspořádaných dle řádu
Binomický strom B_k – 2^k vrcholů, výška k, v hloubce i má k nad i vrcholů; kořen má syny zprava $B_0, B_1, \ldots, B_{k-1}$

Fibonacciho halda

Odstranit (minimum) $\Theta(\log n)$, ostatní $\Theta(1)$

Grafové algoritmy

(Ne)orientovaný; hranově ohodnocený;

Reprezentace grafu

Seznam hran a separovaných vrcholů – pro řídké grafy
Matice sousednosti – n^2 paměť; okamžitě, zda je hrana + její cena
Matice incidence – mxn; d_{ik}=-1 – i je začátek hrany e_k, 1 – i je konec hrany e_k, 0 – jinak
Seznam následníků

Prohledávání grafu

Do hloubky (DFS)– Pro každý v pořadí na 0, vyberu v, z něj rekurzí jdu do synů, v kterých jsem ještě nebyl, čísluji, kdy jsem vkročil a kdy jsem odešel; Nemohu-li táhnout, vyberu nový vrchol, kde jsem ještě nebyl
Klasifikace hran – stromová, zpětná, dopředná, příčná
Graf průchodů do hloubky – DFS-les
Do šířky (BFS) – Z vrcholu kde začínáme dáme do fronty všechny vrcholy, kam z tohoto vede hrana...
Časová složitost $O(|V|+|E|)$

Topologické uspořádání – Takové uspořádání vrcholů, že pro všechny hrany (v_i, v_j) platí $i<j$
G lze topologicky uspořádat <=> G je acyklický <=> DFS nenajde zpětnou hranu
Silně souvislé graf – pro všechna u, v existuje cesta z u do v a zároveň z v do u
Silně souvislá komponenta – maximální podgraf, který je silně souvislý
DFS vezmu časy uzavření v -> pořadí, DFS na opačný G (otočím orientaci hran), stromy ~ kom. souv.
!!! naučit se důkaz !!! str. 28

Problém nejkratší cesty

Dělení dle toho, zda je výchozí nebo cílový vrchol pevný, zda mohou mít i záporné ohodnocení...

Dijkstrův algoritmus

Hledání nejkratší cesty z jednoho vrcholu do ostatních pro nezáporné ohodnocení hran;
Množina vrcholů, jejichž ohodnocení je rovno minimální ceně, jak jsem schopen se k němu dostat;
v každém kroku vyberu z množiny ten s nejmenším ohodnocením, vyřadím ho a ostatní přepočítám...
Složitost – každý v odstraním jedenkrát, každou e jednou použiji pro přepočítání – pole $O(n^2)$, halda $O(m \log n)$
Zpětné vyhledání cesty – od ohodnocení v odečtu ohodnocení e, která do něj vede, a pokud v naproti...
Dk: pro každý přidaný vrchol je cesta nejmenší...

Bellman-Fordův algoritmus

Hledání nejkratší cesty z jednoho vrcholu do ostatních pro reálné ohodnocení hran; test záporných cyklů
V každém kroku přepočítám všechny v, pokud by se ohodnocení změnilo i při $|V|$-tém kroku, je záporný cyklus
Složitost – pro každý vrchol přepočítám všechny hrany $O(|V|\,|E|)$
Dk: indukcí...

Floyd-Warshallův algoritmus
Nalézt nejkratší cestu z každého vrcholu do každého
Pro každý vrchol přepočítá, zda cesta z jednoho v do jiného není kratší přes tento v, pamatuji si předchůdce
Složitost – pro každý vrchol procházím celou matici $O(n^3)$; paměťová složitost $O(n^2)$

Algoritmy násobení matic
Indukcí podle počtu hran na cestě; d_{ij}^{k} minimální cena cesty mezi i, j s nejvýše k hranami
Hodnoty v matici $D^{(k)}$; $D^{(k+1)}=D^{(1)}\Phi D^{(k)}$ místo * sčítání, místo + minimum; nejsou-li záporné cykly, stačí $D^{(n-1)}$
Složitost – využijeme-li asociativitu Φ pak $\Theta(n^3 \log n)$

Tranzitivní uzávěr
Algoritmy lze aplikovat na tranzitivní uzávěr grafu, pokud použijeme booleovské hodnoty & operace

Extremální cesta v acyklickém grafu
Vytvořím topologické uspořádání...
Časová složitost – topologické uspořádání $\Theta(|V|+|E|)$... > O $(|V|+|E|)$

PERT-kritické cesty
Je dána množina úloh, někter na sobě závisí, nejkratší čas, kdy je možné je dokončit
Hrany grafu ~ úlohy, ohodnocení ~ čas trvání, návaznosti ~ v; cesta ~ úlohy, které je třeba vykonat v pořadí
Kritická cesta ~ nejdelší cesta v orientovaném acyklickém grafu – znegování hran/ změna ∞ na $-\infty$ a > na <

Minimální kostra
Vstupem souvislý graf s reálným hranovým ohodnocením; výstupem kostra, která má minimální ohodnocení
lehká hrana – ze všech hran na řezu má nejmenší ohodnocení
řez respektuje množinu hran pokud žádná hrana z množiny nekříží řez
bezpečná hrana – pokud je A podmnožinou minimální kostry, pak i $A \cup e$ je podmnožinou dané kostry
Pokud je hrana e lehká pro řez $(S, V \setminus S)$, tak je bezpečná pro A; důkaz sporem

Kruskalův algoritmus
Zeřadím hrany podle ohodnocení, postupně je vybírám a pokud přidání nevytvoří kružnici, přidám jí
Složitost – třídění hran $\Theta(|E|\log|E|)$, zpracování hrany může mít O(log|E|) (v ve stromové struktuře, pointery)

Jarníkův-Primův algoritmus
Vezmu nějaký vrchol, přepočítám ohodnocení vrcholů mimo, vyberu minimum...
Složitost – binární halda – postavení haldy $O(|V|)$, odstranit min. $|V|O(\log|V|)$, snížení klíčů $|E|O(\log|V|)$
fibonacciho halda – snížení klíče má amortizovanou složitost O(1) => celkově $\Theta(|E|+|V|\log|V|)$

reprezentace v poli – hledání postupným procházením pole $\Theta(|V|^2)$

Metoda rozděl a panuj
Metoda pro návrh rekurzivních programů; úlohu rzdělíme do podúloh stejného typu ale menších; řešení spojíme
Analýza složitosti – $T(n) = a\,T(n/c) + D(n)$ pro $n \geq c$; pro $n<c$ je $T(n)=\Theta(1)$; $T(n)$ je čas trvání úlohy, $D(n)$ čas rozdělení na a podúloh velikosti n/c a opětovné sloučení
Zjednodušení - pro $n<c$ je $T(n)=\Theta(1)$; zanedbáváme celočíselnost; řešení pouze asymptoticky

Substituční metoda
Řešení uhádneme a dokážeme (většinou indukcí) že je to správné Θ i Ω odhad; asym. konst. musí být stejná!!!
Př: Mergesort $T(n)=2T(n/2)+O(n)$; $T(n/2)>c_1(n/2)\log(n/2)$, $T(n)>2c_1(n/2)\log(n/2)+bn=c_1n\log n+(b-c_1)n>c_1n\log n$...

Master theorem
$a \geq 1$, $c>1$, $d \geq 0$ reálná a $T: \mathbb{N} \to \mathbb{N}$ taková, že pro všechna n tvaru c^k platí $T(n)=aT(n/c)+O(n^d)$
$a<c^d$, tj. $\log_c a<d$, potom $T(n)=O(n^d)$
$a=c^d$, tj. $\log_c a=d$, potom $T(n)=O(n^d \log_c n)$
$a>c^d$, tj. $x=\log_c a>d$, potom $T(n)=O(n^x)$
Dk: !!! naučit se důkaz !!! str. 42

$0<a_i<1$, $d\geq 0$ reálná a $T: \mathbb{N} \to \mathbb{N}$ taková, že pro všechna n platí $T(n)=\sum T(a_i n)+O(n^d)$; a l je řešením $\sum a_i^x=1$

$\sum a_i^d<1$, potom $T(n)=O(n^d)$

$\sum a_i^d=1$, potom $T(n)=O(n^d \log_c n)$

$\sum a_i^d>1$, potom $T(n)=O(n^l)$

Př: Násobení dlouhých čísel $T(n)=4*T(n/2)+O(n)$

Rychlé násobení: $u=(x_1+x_2)(y_1+y_2)$, $v=x_1+y_1$, $w=x_2+y_2$; $xy=v*2^n+(u-v-w)*2^m+w$; $T(n)=3T(n/2)+O(n)$?!?!?!

Násobení matic $T(n)=8*T(n/2)+O(n^2)$

Strassenův algoritmus násobení matic: v praxi pro $n>45$; $T(n)=7*T(n/2)+O(n^2)$?!?!?!

$M_1=(A_{12}\ominus A_{22})\otimes(B_{21}\oplus B_{22})$ $M_2=(A_{11}\oplus A_{22})\otimes(B_{11}\oplus B_{22})$ $M_3=(A_{11}\ominus A_{21})\otimes(B_{11}\oplus B_{12})$

$M_4=(A_{11}\oplus A_{12})\otimes B_{22}$ $M_5=A_{11}\otimes(B_{12}\ominus B_{22})$ $M_6=A_{22}\otimes(B_{21}\ominus B_{11})$ $M_7=(A_{21}\oplus A_{22})\otimes B_{11}$

$C_{11}=M_1\oplus M_2\ominus M_4\oplus M_6$ $C_{12}=M_4\oplus M_5$ $C_{12}=M_6\oplus M_7$ $C_{22}=M_2\ominus M_3\oplus M_5\ominus M_7$

Třídění

Quicksort

V ideálním případě $n\log n$, v nejhoším n^2, předpokládá pravděpodobnostní rozložení => existují špatné vstupy

Dk složitosti v průměrném případě: !!! naučit se důkaz !!! str. 46

Randomizovaný Quicksort

Odstraňuje problém možné větší pravděpodobnosti výskytu špatných posloupností; volba pivota náhodně

složitost je průměrem volby pivotů při všech dělících bodech... neexistuje špatný vstup, jen volba pivota

Dolní odhad třídění založeného na porovnávání

Musíme rozlišit $n!$ posloupností – rozhodovací strom musí mít alespoň $n!$ Listů => minimální hloubka $2^h>n!$

$h>\log n!=\log n(n-1)...1=\log [(n+1)/2]^{(n/2)}>(n/2)\log(n/2) => \Omega(n \log n)$

=> Mergesort a Heapsort jsou optimální třídící algoritmy

Radixsort

Klíč rozdělíme na d dílů; třídíme do p přihrádek (od méně významných cifer), v nichž zachováme pořadí vstupu

Složitost $O(dn+dp)$, d předpokládáme konstantní

Countingsort

Při prvním průchodu uložíme počet vstupních čísel & rozmístění výsledků; v druhém ukládáme na místa

Složitost $O(k+n)$ – počet možných klíčů + počet prvků