Asymptotická složitost

1. Algoritmus A projde celým polem délky N a prvek s indexem *k* zpracuje za *c*·*k* milisekund. Konstanta *c* je stále stejná. Určete asymptotickou složitost zpracování celého pole.

index	doba zpracování ms
0	$c \cdot 0$
1	$c \cdot 1$
2	$c \cdot 2$
3	$c \cdot 3$
4	$c \cdot 4$
...	...
k	$c \cdot k$
...	...
N–1	$c \cdot (N-1)$

---------- aplikuj součet aritmetické posloupnosti:

Celkem $c \cdot 0 + c \cdot 1 + c \cdot 2 + ... + c \cdot (N-1) = (c \cdot 0 + c \cdot (N-1)) \cdot N / 2 = c \cdot (N-1) \cdot N / 2$

$c \cdot (N-1) \cdot N / 2 \in \Theta(N^2)$,

2. Algoritmus P projde celým dvourozměrným polem velikosti N×N a prvek na pozici (k, m) zpracuje za
A) $c\cdot(k+m)$, B) $c\cdot k\cdot m$ milisekund. Konstanta c je stále stejná, $0 \le k \le N-1$, $0 \le m \le N-1$. Určete asymptotickou složitost zpracování celého pole v případě A) a B).

A)
Určíme dobu zpracování řádku s indexem k, index m se v řádku mění a určuje sloupce.

Suma{ $c\cdot(k+m)$, pro $m = 0, 1, ..., N-1$ }
Suma{ $c\cdot k+c\cdot m$, pro $m = 0, 1, ..., N-1$ }
Suma{ $c\cdot k$, pro $m = 0, 1, ..., N-1$ } + Suma{ $c\cdot m$, pro $m = 0, 1, ..., N-1$ }
$c\cdot k \cdot$ Suma{ 1, pro $m = 0, 1, ..., N-1$ } + $c \cdot$ Suma{ m, pro $m = 0, 1, ..., N-1$ }
$c\cdot k\cdot N + c \cdot (N-1) \cdot N / 2$

Doby zpracování všech řádků sečteme podle indexu řádků k.
Suma{ $c\cdot k\cdot N + c\cdot(N-1)\cdot N/2$, pro $k = 0, 1, ..., N-1$ }
Suma{ $c\cdot k\cdot N$, pro $k = 0, 1, ..., N-1$ } + Suma{ $c\cdot(N-1)\cdot N/2$, pro $k = 0, 1, ..., N-1$ }
$c\cdot N \cdot$ Suma{ k, pro $k = 0, 1, ..., N-1$ } + $c\cdot(N-1)\cdot N/2 \cdot$ Suma{ 1, pro $k = 0, 1, ..., N-1$ }
$c\cdot N\cdot (N-1)\cdot N/2 + c\cdot (N-1)\cdot N/2\cdot N$
$c\cdot N\cdot N\cdot (N-1)$

$c\cdot N\cdot N\cdot (N-1) \in \Theta(N^3)$,

2. Algoritmus P projde celým dvourozměrným polem velikosti N×N a prvek na pozici (k, m) zpracuje za
A) $c\cdot(k+m)$, B) $k\cdot m$ milisekund. Konstanta c je stále stejná, $1 \leq k \leq$ N, $1 \leq m \leq$ N.
Určete asymptotickou složitost zpracování celého pole v případě A) a B).

B)
V úvahu připadají složitost $\Theta(N^3)$ a $\Theta(N^4)$.
Načrtněte si pole P velikosti N × N, pro N = 7, a do každého jeho prvku napište konkrétní dobu zpracování prvku na odpovídající pozici v původním poli.

B1.
Uvažte prvek $P[N-1][N-1] = c\cdot (N-1) \cdot (N-1)$, jehož hodnota je největší v celém poli, a zdůvodněte, alespoň neformálně, že asymptotická složitost celého procesu je $O(N^4)$, tedy buď úměrná N^4 nebo řádově menší.

B2.
Všimněte si diagonály P, obsahuje druhé mocniny čísel 0, 1, 2, 3, ..., $N-1$.
Platí vztah
$1 + 4 + 9 + 16 + 25 + ... + N^2 = N \cdot (N+1) \cdot (2N+1) / 6$
S využitím tohoto vztahu a hodnot na diagonále zdůvodněte, že zpracování původního pole má složitost alespoň kubickou, tj složitost $\Omega(N^3)$.

B3.
Sečtěte mechanicky s pomocí vzorce pro součet aritmetické posloupnosti všechny hodnoty v P pro obecné N (nejen 7). Nejprve sečtěte hodnoty v každém řádku a součet napište na konec řádku. Potom sečtěte součty na konci všech řádků.

3. Symbolem lg značíme logaritmus o základu 2. Uspořádejte podle řádu růstu uvedené funkce poměnné *n*. Zdůvodněte pořadí každých dvou sousedních funkcí v tomto uspořádání.

$\lg(n!)$ $\left(\sqrt{2}\right)^{\lg(n)}$ $2^{\lg(\lg(n))}$ $4^{\lg(n)}$ $\sqrt{\lg(n)}$ $n \cdot \lg(n^2)$ $n \cdot \lg(n)$ $\lg(n)^2$

A. Pro druhý výraz zleva platí:
(sqrt(2))^lg(*n*) = (2^(1/2))^lg(*n*) = 2 ^ (1/2 * lg(*n*)) = 2 ^ (lg(*n*) * 1/2) = (2 ^ lg(*n*)) ^ 1/2) = *n* ^ 1/2 = sqrt(*n*)

B.
Zjednodušte, pokud možno druhý, třetí a čtvrtý výraz a vyjádřete jej jako mocninu se základem *n*, podobně jako v předchoyím odstavci A.

C. U prvního výrazu využijte fakt, že "logaritmus součinu je součet logaritmů". Vzniklý součet zkuste omezit shora
n-násobkem největšího sčítance.

D. Co nejvíce (optimálně všech osm) výrazů zkuste uspořádat podle řádu růstu s využitím znalostí získaných v A. B. a C.

4. Ověřte, že platí tvrzení

$(\forall n \in \mathbf{N})\ (\ n > 4 \ \Rightarrow\ n^2 - 2n\ >\ 0.5n^2\ )$.

Podle tohoto tvrzení tedy graf funkce

$f(n) = 0.5n^2$

pro $n > 4$ leží vždy pod grafem funkce

$g(n) = n^2 - 2n$

a navíc rozdíl $g(n) - f(n)$ roste do nekonečna s rostoucím n.

Zdůvodněte pomocí definice množiny $O(f(n))$, že i přesto platí $g(n) \in O(f(n))$.

4A. Načrtněte $f(n)$ a $g(n)$.

4B. Spočtěte a porovnejte navzájem $f(5)$ a $g(5)$, $f(10)$ a $g(10)$, $f(50)$ a $g(50)$, $f(100)$ a $g(100)$.

4C. Napište definici množiny $O(f(n))$.

4D. Napište důvod toho, že $g(n) = n^2 - 2n \ \in O(f(n))$.

Zkuste celé samostatně.

Kostry a minimální kostry

* 5.
(A) V grafu k ceně každé hrany přičteme stejnou nenulovou konstantou c.
(B) V grafu vynásobíme ceny všech hran stejnou nenulovou konstantou c.
V jakém vztahu budou minimální kostry původního a upraveného grafu v případech (A) a (B)? Předpokládejte, že původní minimální kostra je určena jednoznačně.

Hint: Zvolte konkrétní algoritmus, možná je nejlepší Kruskal, a porovnejte jeho činnost v původním a v upraveném grafu. Pokud se budou přidávat do kostry stejné hrany v původním i v upraveném grafu, vznikne stejná minimalní kostra. Jejich celkove ceny (= součet cen všech hran kostry) pak lze snadno porovnat.

* 6. Máme najít kostru grafu, nikoli nutně minimální, s tím, že je předepsáno, že cena každé hrany hledané kostry musí ležet v daném intervalu <c1, c2>. Je nutno použít algoritmus pro hledání minimální kostry nebo stačí nějaký jednodušší postup?

Dejme tomu, že při hledání této specifické kostry budeme ignorovat hrany, jejichž cena je mimo interval <c1, c2>.
a) Rozhodněte, jestli lze za této podmínky použít DFS.
b) Rozhodněte, jestli lze za této podmínky použít BFS.
c) Napadá vás nějaký ještě rychlejší postup nebo rychlejší není možný?

* 7. Uveďte asymptotickou složitost algoritmu hledání minimální kostry jednak Primova a jednak Kruskalova.
Který z těchto algoritmů je asymptoticky rychlejší, za předpokladu, že počet hran grafu je čtyřnásobkem počtu uzlů?

Počet hran grafu je čtyřnásobkem počtu uzlů. Jinými slovy, když se zvětšuje počet uzlů, zvětšuje se asymptoticky stejně rychle i počet hran. Jinými slovy, $|E| = \Theta(|V|)$ a $|V| = \Theta(|E|)$. Asymptoticky jsou oba počty "stejné".
Dosaďte V na místo E všude ve výrazech pro asymptotickou složitost obou algoritmů. a upravené výrazy porovnejte.

A. Zvolte vhodné výrazy pro asymptotickou složitost obou algoritmů (bývají v přednášce).
B. Výrazy porovnejte a napište závěr.

* 8. Předpokládejme, že vážený neorientovaný graf G je reprezentován svou váhovou maticí C. Určete, jaká bude asymptotická složitost Kruskalova algoritmu hledání minimální kostry za předpokladu, že doba přístupu ke každému prvku matice C je konstantní, ale zato doba každé jednotlivé operace Union i Find je vždy úměrná počtu uzlů v grafu G.

Spočtěte dobu řazení:

U obou daných akcí určete jejich složitost:

8A. Průchod maticí a extrakce hran do seznamu (pole).

8B. Řazení seznamu (pole) hran.

8C. Která složitost je větší?

Spočtěte dobu vkládání hran do kostry:

8D. Jakou dobu potřebujeme na zpracování jedné hrany? (to je v textu úlohy)

8E. Kolik hran maximálně budeme muset prohlédnout (a zpracovat)?

8F. Doba vkládání do kostry je součinem předchozích dvou hodnot.

* 9. Na svém počítači máte najít minimální kostru úplného váženého grafu. Odhadněte, kolik řádově nejvýše uzlů může takový graf mít, abyste úlohu vyřešili přes noc do druhého dne. Zdůvodněte svůj odhad.

K dispozici je cca 8×3600 sec tedy něco kolem 30 000 sec.
Jedna jednoduchá akce s jednou hranou nebo s jedním uzlem zabere cca 10^{-8} sec.
Kolik můžeme provést celkem za noc jednoduchých akcí s hranou nebo uzlem? (*)
V úplném grafu je přibližně $|E| = |V|^2/2$.
Do výrazu popisujícího složitost hledání MST dosaďte $|V|^2/2$ na místo $|E|$ a položte tento výraz roven celkovému počtu jednoduchých akcí spočtenému v (*).
Co získáme řešením takové rovnice?

10. V souvislém ohodnoceném grafu G s n uzly a m = 5n hranami máme nalézt minimální kostru T. Poté máme z T vyrobit jinou kostru T1, ne už minimální, tak, že nejlacinější hranu T odstraníme z T a poté do vzniklého nesouvislého grafu přidáme nejdražší možnou hranu v G tak, aby výsledný graf byl opět stromem, tedy i kostrou. Navrhněte algoritmus řešení této úlohy a určete jeho asymptotickou složitost.

11. Velitelství Graphlandu hledá co nejlacinější kostru grafu s tím omezením, že u některých hran grafu je předepsáno, že v kostře musí být bez ohledu na jejich cenu. Lze tedy čekat, že výsledná kostra nebude nejlacinější možná. Niceméně i v tomto případě lze použít jeden z algoritmů hledání minimální kostry pro splnění úlohy. Naštěstí žádná podmnožina předepsaných hran netvoří kružnici. Napište algoritmus vašeho řešení a určete jeho asymptotickou složitost.

12. V předchozí úloze se ukáže, že některé předepsané hrany tvoří dohromady kružnici v grafu. Velitelství tedy pozmění úlohu tak, že již nežádá kostru ale nejlacinější propojení všech uzlů grafu, a dále trvá na tom, že všechy předepsané hrany musí být použity. Ukažte, že vůči předchozí úloze nemusí být v řešení prakticky žádný rozdíl.

13. Předpokládejme, že graf je zadán maticí vah jednotlivých hran. Význačná hodnota v této matici (např. nekonečno, minimální/maximální hodnota číselného typu, NaN apod) indikuje, že mezi příslušnými vrcholy hrana neexistuje. Modifikujte Jarníkův-Primův algoritmus tak, aby nezávisel na počtu hran v grafu a měl složitost □(n2), kde n je počet uzlů grafu.

14. Nalezli jsme a odevzdali zákazníkovi minimální kostru jím dodaného grafu s mnoha miliony vrcholů. Odpoledne zákazník volá, že v zadání je chyba a že hrana mezi vrcholy 2075154 a 11439446 je ve skutečnosti o 17% levnější, než v jak bylo uvedeno v původní specifikaci. Veškerá data grafu i kostry jsou dosud na našem disku. Máme určit v lineárním čase vzhledem k počtu vrcholů, zda tato změna ovlivní tvar a cenu minimální kostry, a pokud ano, vydat co možná nejkratší opravu, která se dá tlumočit zpět telefonem.

15. Zopakujte analýzu předchozí úlohy pro případ, že původně chybně zadaná hrana se nakonec ukazuje být ve skutečnosti dražší.

16. Stačí fakt, že všechny hrany mají různou cenu, k tomu, aby minimální kostra byla určena jednoznačně? Zdůvodněte svou odpověď, můžete si pomoci průběhem Kruskalova algoritmu.

17. Připomeňte si, kolik nejvíce hran může mít rovinný graf s n uzly a pak odpovězte na otázku: Jaký algoritmus nejlépe použijete pro hledání minimální kostry tohoto grafu a jaká bude jeho složitost v závislosti pouze na n?

18. Nenasytný loupežník odebírá z grafu co možná nejdražší hrany (uzly ponechává na místě). Nesmí ale graf rozpojit na dvě nebo více komponent, protože by byl odhalen a dopaden. Když odebere maximum všeho, co se dá, zbyde z grafu jeho minimální kostra?

19. Jak se hledá maximální kostra grafu pomocí algoritmu pro hledání minimální kostry?

20. Grafy G1 a G2 mají identickou strukturu (jsou izomorfní), ale v G1 mají skoro všechny hrany různou cenu, zatímco v G2 mají skoro všechny hrany stejnou cenu. Jaký algoritmus hledání minimální kostry navrhujete použít pro G1 a jaký pro G2 a proč?

21. Zpětnovazební množina hran kladně ohodnoceného grafu je každá taková množina hran, která obsahuje alespoň jednu hranu každé kružnice v daném grafu. Z toho plyne, že když zpětnovazební množinu z grafu odstraníme, zbude graf, který je zcela bez kružnic. Jak máme určit co nejlacinější zpětnovazební množinu? (Cena množiny hran je součtem cen všech hran v této množině.)

22. Vysvětlete, proč Borůvkův algoritmus hledání minimální kostry obsahuje podmínku vyžadující, aby váhy všech hran byly navzájem různé a uveďte příklad grafu, na němž by algoritmus selhal, kdyby algoritmus tuto podmínku neobsahoval.

23. Předpokládejme, že vážený neorientovaný graf G je reprezentován svou váhovou maticí C. Určete, jaká bude asymptotická složitost Primova algoritmu hledání minimální kostry za předpokladu, že doba přístupu ke každému prvku matice C je přímo úměrná počtu uzlů G.