O f(w) 10100 1 f(w) g(v)

WB WB WB WB

w_1 w_2 w_3

Final Examination on Algorithms

Teacher: Biing-Feng Wang **Jan. 12, 2022**

Remark. For each problem, you must justify your answer. All time complexities are in worst-case, unless otherwise specified. It is suggested that your algorithms are described in words and examples (instead of in pseudo codes), unless pseudo codes are asked to be provided.

Problem 1: (11%, Problems selected from midterm examination-Part I)

(1) (4%) Let $f(n)$, $g(n)$, and $h(n)$ be asymptotically positive functions such that $f(n) + 50 = O(g(n))$ and $0.05 \times g(n) = O(h(n))$. Using the definition of O-notation, prove that $f(n) = O(h(n))$.

(2) (7%) Find an upper bound on the recurrence $T(n) = 4T(\lfloor n/4 \rfloor) + n$ by appealing to the substitution method. (You may assume that $T(1) = T(2) = T(3) = 1$.)

Problem 2: (14%, Problems selected from midterm examination-Part II)

(1) (7%) Consider the problem of finding the smallest set S of unit-length closed intervals that contain a set X of n given points. Prove that this problem exhibits the following two properties: greedy-choice property and optimal substructure.

(2) (7%) Let $A_1, A_2, \ldots, A_k$ be k non-empty sorted lists that contain a total number of n elements, where $k = 20 \times (\lg n)^{1/2}$ and each A_i may have a different length. Give an efficient algorithm to merge all lists into one. What's the time complexity, in terms of n, of your algorithm? Explanation is necessary.

Problem 3: (10%, Disjoint sets) Suppose that linked-lists are used to represent disjoint sets. Prove that using the weighted-union heuristic, a sequence of m Make-Set, Union, and Find-Set operations takes $O(m + n \lg n)$ time, where n is the number of Make-Set operations.

Problem 4: (12%, Depth-first search) Let $G = (V, E)$ be a directed graph. According to the depth-first forest G_π, we can classify the edges of G into four types.

(1) (4%) List the names of these edge types. (No definitions are required.)

(2) (8%) Modify the depth-first search algorithm to classify the edges as it encounters them. No explanation is necessary.

Problem 5: (10%, Minimum spanning trees)

(1) (6%) Describe Kruskal's minimum spanning tree algorithm. (No proof for the correctness.)

(2) (4%) Assume that the length of each edge is a positive integer less than n. What is the running time of Kruskal's algorithm? Explanation is necessary.

Problem 6: (10%, Single source shortest paths)

(1) (5%) Describe Bellman-Ford's algorithm for the single source shortest paths problem.

(2) (5%) Assume that the input graph does not contain negative cycles. Prove the correctness of Bellman-Ford's algorithm.

Problem 7: (10%, Number-theoretic algorithms)

(1) (6%) Let x and a be two positive integers. Give an efficient algorithm that computes x^a.

(2) (4%) What is the time complexity of your algorithm? Is it polynomial or pseudo polynomial? Explanation is necessary.

Problem 8: (13%, Approximation algorithms)

(1) (5%) Describe an approximation algorithm with ratio bound 2 for the Euclidean TSP problem.

(2) (3%) What is the time complexity? Justify your answer.

(3) (5%) Prove that your algorithm in (1) has a ratio bound 2.

Problem 9: (10%, NP-completeness) Define the following terms and draw a Venn diagram to describe their relationships: P, NP, NP-complete, NP-hard. (Assume that NP $\neq$ P.)

Problem 10: (10%, homework) This problem is to verify whether or not you did homework by yourself. Please answer either of the following. (If you answer both, only the one getting less score will be counted.)

(a) Suppose we wish not only to increment a counter but also to reset it to zero (i.e., make all bits in it 0). Counting the time to examine or modify a bit as $\Theta(1)$,show how to implement a counter as an array of bits so that any sequence of n INCREMENT and RESET operations takes time $O(n)$ on an initially zero counter. Justify your answer. Explanation is necessary.

(b) Let $G = (V, E)$ be a directed graph in which each edge has a real weight. We define the *mean weight* of a cycle $c = (e_1, e_2, \ldots, e_k)$ to be

$$\mu(c) = (\Sigma_{1 \le i \le k}\, w(e_i)) / k,$$

where $w(e)$ denotes the weight of an edge $e \in E$. Let $\mu^* = \min_c \mu(c)$, where c ranges over all directed cycles in G. Assume that every vertex is reachable from a source vertex s. Let $\delta(s, v)$ be the weight of a shortest path from s to a vertex v, and let $\delta_k(s, v)$ be the weight of a shortest path from s to v consisting of exactly k edges. Show that if $\mu^* = 0$, then

$$\max_{0 \le k \le n-1} (\delta_n(s, v) - \delta_k(s, v)) / (n - k) \ge 0$$

for all vertices $v \in V$.

1. (1) 已知. $\exists\ c_0, n_0 > 0 \ni f(n)+50 \le c_0 \cdot g(n)\ \forall n \ge n_0 > 0$ 且

$\exists\ c_1, n_1 > 0 \ni 0.05 \cdot g(n) \le c_1 \cdot h(n)\ \forall n \ge n_1 > 0$ ，令 $n_2 = \max\{n_0, n_1\}$

$\rightarrow f(n) \le c_0 \cdot g(n) - 50$ 且 $g(n) \le \frac{c_1}{0.05} \cdot h(n)\ \forall n \ge n_2 > 0$

$\rightarrow f(n) \le \frac{c_0 \cdot c_1}{0.05} h(n) - 50 \le \frac{c_0 \cdot c_1}{0.05} \cdot h(n)\ \forall n \ge n_2 > 0$

$\rightarrow f(n) = O(h(n))$

(2) guess: $T(n) = O(n \lg n)$

basis: $n_0 = 2$, $T(2) \le c \cdot 2 \cdot \lg 2$? ok for $c \ge \frac{T(2)}{2\lg 2}$

$T(3) \le c \cdot 3 \cdot \lg 3$? ok for $c \ge \frac{T(3)}{3\lg 3}$

$T(4) \le c \cdot 4 \lg 4$? ok for $c \ge \frac{T(4)}{4\lg 4}$

$T(5) \le c \cdot 5 \lg 5$? ok for $c \ge \frac{T(5)}{5\lg 5}$

$T(6) \le c\, 6 \lg 6$? ok for $c \ge \frac{T(6)}{6\lg 6}$

$T(7) \le c\, 7 \lg 7$? ok for $c \ge \frac{T(7)}{7\lg 7}$

$\rightarrow$ 取 $n_0 = 2, 3, 4, \ldots, 7$, $c \ge \frac{T(n_0)}{n_0 \lg n_0}$ 可使 basis holds. —— ①

Induction: 設 $\forall k$ s.t. $n_0 \le k < n$ $T(k) \le c \cdot k \lg k$

, where $n \ge 8$

當 $k = n$, $T(n) = 4T(\lfloor \frac{n}{4} \rfloor) + n$

require $n \ge 8$

$\le 4 \cdot c \lfloor \frac{n}{4} \rfloor \lg \lfloor \frac{n}{4} \rfloor + n$

$\le 4 \cdot c \frac{n}{4} \lg \frac{n}{4} + n$

$= cn \cdot \lg n - cn \lg 4 + n$

-1

$\le cn \cdot \lg n$

when $-cn \lg 4 + n \le 0$

$\rightarrow -\lg 4 \cdot c + 1 \le 0$

$\rightarrow c \ge \frac{1}{\lg 4}$ —— ②

根據①、②，取 ~~$n_0 = 2, 3, \ldots, 7$~~, $c =$ ~~$\max\{\frac{1}{\lg 4},$~~

取 $n_0 = 2$

$\max\left\{\frac{1}{\lg 4}, \max_{2 \le k \le 7}\left\{\frac{T(k)}{k \lg k}\right\}\right\}$

可使 basis & induction step holds, 得證 #

2. (1) greedy-choice property:

每次選取 n points 中最小的点 y，並加入 $[y, y+1]$。

proof. 設 Y 為最佳解且 $[y, y+1] \notin Y$，由於 y 為最小，因此實線上左邊無其他点，因此可用 $[y, y+1]$ 代替 Y 中用來 cover y 的 interval (e.g. $[z, z+1]$

$X = Y - [z, z+1] \cup [y, y+1]$ 由於 interval 總數不變且也 cover 每個点，($z \le y$)。

因此 X 也是最佳解 #

optimal substructure: 設 S 為 n points 的最佳解，

選了 $[y, y+1]$，並移除被 $[y, y+1]$ cover 的 points，剩下的 point set 為 P

利用反證法，設對 P 的解 S' 不為最佳，則可取 P 的一個最佳解 S^*

使得 $|S'| > |S^*|$，用 S^* 取代 S' 就得到 S'' 且 $|S''| < |S|$，

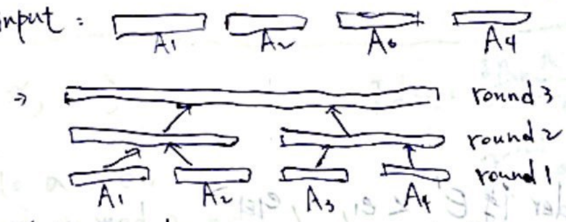
矛盾了 S 為 optimal 的假設，因此具 optimal substructure #

(2) 設 A_i 為由小到大排序，$i = 1, \dots, k$，

algo:

將 A_i 兩兩合併，直到剩一個 list。

e.g. input:



時間: 每個 round 共 n 個 elements 合併 → $O(n)$

兩兩合併，所以共 ~~lg k~~ $O(\lg k)$ round

→ 總共 $O(\lg k) \cdot O(n) = O(n \cdot \lg k) = O(n \cdot (\lg 20 + \frac{1}{2} \lg\lg n)) = O(n \lg\lg n)$ #

3. m 個 operations 後，

(1) Union 的時間為所有 element 的被降次數加總 (union 中被接到另一個 list 後)，設 $t(x)$ 為 x 的被降次數，

L_x 為最後包含 x 的 list，則觀察可得 $2^{t(x)} \le |L_x| \rightarrow t(x) \le \lg |L_x|$

因 L_x 最長為 $n \rightarrow t(x) \le \lg n$

總被降 $= \sum_{\text{所有}x} t(x) \le n \cdot \lg n$，得 m 個 operations 的 union 時間為 $O(n \lg n)$

(2) Make-Set 和 Find-Set 皆為 $O(1)$，共 m 個 operations → $O(m)$

根據(1).(2)，總共為 $O(m + n \lg n)$ #

4. (1) tree edge、back edge、forward edge、cross edge。

(2)

```
main:
    for each u ∈ G.V:
        u.color = white
        u.π = Nil
    t = 0
    for each u ∈ G.V:
        if u.color == white:
            DFS(u)

def DFS(u):
    t = t+1
    u.d = t
    u.color = gray
    for each v ∈ u.adj():
        if v.color == white:
            v.π = u
            print((u,v),"is tree edge")        ——→ tree
            DFS(v)
        elif v.color == grey:
            print((u,v),"is back edge")        ——→ back
        else:
            if u.d < v.d:
                print((u,v),"is forward edge") ——→ forward
            else:
                print((u,v),"is cross edge")   ——→ cross
    t = t+1
    u.f = t
    u.color = black
```

9

5. (1)

```
Kruskal(G):
    S = ∅
1   for each u ∈ G.V:
2       MAKE-Set(u)
3   排序 G.E in non-decreasing order 得 E' = <e1, ..., e|E|>
4   for each (u,v) in E' (會按順序取(u,v)):
5       if Find-set(u) ≠ Find-set(v):
6           Union(u, v)
7           S = S ∪ {(u,v)}
8   return S
```

(2) 將 edge length 視為 n 進位數，則每個 edge 為 1 位 n 進位數

對 G.E 做一輪 counting sort 所需時間為 $O(n+n) = O(n) = O(V)$ ~ L_3 的時間

$|E|$

$L_1 \sim L_2 : O(V)$

$L_4 \sim L_7$: for 最多做 $O(E)$ 輪，每輪若用 fibonaci-heap implement disjoint-set，則 FIND-Set、Union 為 $O(\alpha(V)) \rightarrow O(E \cdot \alpha(V))$

→ 總共為 $O(V + E \cdot \alpha(V))$ #

6. (1) Bellman-Ford (G, w, s):

u.d 表示 s 到 u 的 distance
u.π 表示 u 的 predecessor in shortest path tree starting from s.
(下稱 ST)

10

```
for each u ∈ G.V:
    u.d = ∞
    u.π = Nil
s.d = 0
for i = 1 to V-1:                          ① Relax 每ㄍ edges 共 V-1 輪
    for each edge (u, v) ∈ G.E:
        if u.d + w(u, v) < v.d:
            v.d = u.d + w(u, v)
            v.π = u
for each edge (u, v) ∈ G.E:                檢查 negative cycle
    if u.d + w(u, v) < v.d:
        return False
return True
```

~~不應使用 spanning tree 證~~ 模糊

(2) 設 T 為 G 的 ST starting from s, 則 T 中每 ~~edge~~ vertex 到 s 的邊數 (shortest path)

可分為 $U_1, U_2, \ldots, U_{M-1}$, (因沒 negative cycle, 所以最長為 $M-1$),

一開始 i=1 時, U_1 被 s relax, 因此已完成。設 U_i s.t. $1 \le i < k$. 都完成了

~~每次做①的一輪,~~ 當①的 for $i=k$ 時, U_k 的 vertex 就會被 relax, 並且 U_{k-1} 在 $i=k-1$ 時已為完成計算, 所以 U_k 也就正確計算了。

→ 做 $M-1$ 輪, $U_1, \ldots, U_{M-1}$ 就全算完了。

e.g. s, U_1, U_2

~~初始條件沒有證明~~

7. (1) Fast (x, a):

10

```
s = 1
while a > 0:
    if a mod 2 == 1:
        s = s * x
    x = x·x
    a = a div 2
return s
```

Idea: 設 $a = 101_2$, 則 ~~$x^a = x^{\ldots} = x^{\ldots}$~~

$x^{101} = x^{100} \cdot x^{000} \cdot x^{001}$

$= x^{2^2} \cdot x^{2^0}$

從 a 的低位開始掃描每ㄍ bit, 掃到 k th bit 時若為 1 表示 x^{2^k} 可乘進解 → $s = s * x$。

(2) while-loop 以 a > 0 為條件, 且 a 每輪都会除以 2, 因此共 $\lg_2 a$ 輪。

每輪 $O(1)$ → $O(\lg a)$

因 a 為整數, 共要 $\lg a$ bits → $O(\lg 2^{\lg a}) = O(\lg a)$ → polynomial #

(用 bit 數表示)

13

8. (1) Algo:

1. 任取一点 r (當 root), 並做 Minimum Spanning Tree by Prim, 得 MST: T
2. 对 T 從 r 開始的 preorder traversal $\langle r, v_1, \dots, v_n \rangle$

return $\langle r, v_1, \dots, v_n, r \rangle$

(2) 第一步 用 array 當 disjoint set, 則 Prim 要 $O(V^2)$

第二步 走訪 T 要 $O(V)$

→ 共 $O(V^2)$ #

(3) 定義 $w(\cdot)$ 為一~~path~~走訪序列的 cost. e.g.

則 $w(u \xrightarrow{3} v \xrightarrow{3} u) = 6$

$w(u \xrightarrow{3} v) = 3$

① 对 T 做 full walk 得 F, 則 $w(F) = 2\cdot(\text{T.E 的和})$. e.g.

T的edges weigt sum　　full walk

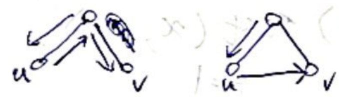
② 設 C^* 為 ETSP 的最佳解, 則自 C^* 任取一 edge 移除, 会形成一 spanning tree T'

→ T'.E 的和 ≥ T.E 的和 → $w(C^*) \geq$ T.E 的和

③ 根據 ETSP 的特性知, 走直線比繞路快 → $w(F) \geq w(\text{T 的 preorder list})$

根據 ①、②、③, $w(\text{T的 }\sout{\text{predict}}\text{ preorder list}) \leq w(F) \leq 2\cdot(\text{T.E的和}) \leq 2\cdot w(C^*)$, 得證 #

e.g.



從 u 到 v, 走直線較快

9. P: 可在 $O(n^k)$ (k 為 constant) 時間, 用 deterministic algo 解決的 problem set.

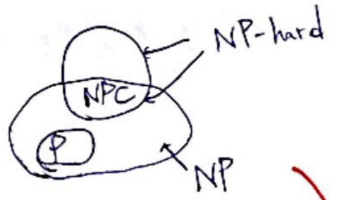
NP: 可在 $O(n^k)$ 用 non-deterministic algo 解的 problem set.

1. 先 guess 一組答案
2. 驗證

NP-complete: 所有在 NP 的問題皆可 reduce to s in $O(n^k)$

且 $s \in NP$, 則 $s \in$ NP-complete. (反向也是. i.e. 若 $s \in NPC$, 則 ⋯)

NP-hard: 所有在 NP 的問題皆可 reduce to s, 則 $s \in$ NP-hard (反向也是)



10. (b) 因 $n-k>0$，所以相當於~~要~~證 $\delta_n(s,v) - \min\limits_{0\le k\le n-1} \delta_k(s,v) \ge 0$

設 $\mu^*=0$，表示 no negative cycle $\rightarrow$ $\delta(s,v)$ 最多用 $n-1$ 條 edges，多走圈不會使距離成本更小

$\rightarrow \delta(s,v) = \min\limits_{0\le k\le n-1} \delta_k(s,v)$

$\rightarrow \delta_n(s,v) \ge \min\limits_{0\le k\le n-1} \delta_k(s,v) = \delta(s,v)$

$\rightarrow \delta_n(s,v) - \min\limits_{0\le k\le n-1} \delta_k(s,v) \ge 0$，得證 #