Department of Computer Science and Engineering
National Sun Yat-sen University
Design and Analysis of Algorithms - Final Exam., Jan. 11, 2022

1. Explain each of the following terms. (20%)
   (a) NP, NP-complete
   (b) the Eulerian cycle of a graph
   (c) hill climbing in the searching strategy
   (d) the knapsack problem
   (e) Graham scan
2. Explain the *longest common subsequence* (LCS) problem. And, then give an example to illustrate your answer. Note that you should give both explanation and example. (8%)
3. Please derive the recurrence formula $T(n)=2T(n/2)+O(n)$ to get the time complexity for $T(n)$, where n denotes the size of the input data. (12%)
4. Present an algorithm for solving the *shortest path* (from a single source) problem on a graph. Analyze the time complexity of your algorithm. (12%)
5. Please present the *prune-and-search* method for solving the *constrained 1-center* problem (the center should be on the line $y=0$). (12%)
6. The first-fit algorithm is a simple approximation algorithm for solving the *bin packing* problem. The algorithm puts an object into the ith bin as long as it is available and tries the $(i+1)$th bin if otherwise. Show that the number of bins used in the first-fit algorithm is no more than twice the number of bins needed in an optimal solution. (12%)
7. Prove that the *clique decision* problem polynomially reduces to the *node cover decision* problem. (12%)
8. Please explain the *transpose* heuristics, *move-to-front* heuristics, and *count* heuristics for the self-organizing sequential search. (12%)

Answer:

1.

(a).

NP：可以用 non-deterministic polynomial algorithm 解決的 decision problem

NP-complete：同時為 NP 與 NP-hard 的問題

(b).

每條邊都會且只能經過一次的封閉環。

(c).

DFS 的一種變型，在展開每一層節點時，會計算此層節點的預估值，並優先展開此層預估值最高的節點。

(d).

背包問題有兩種，第一種是物品可被分割的，第二種是物品不可被分割的 0/1 問題。會給予每個物品的重量和價值，以及背包的最大負重限制，目標為在不超重的情形，最大化背包能裝的價值。

(e).

2. 用於尋找 convex hull 的演算法，其步驟為：

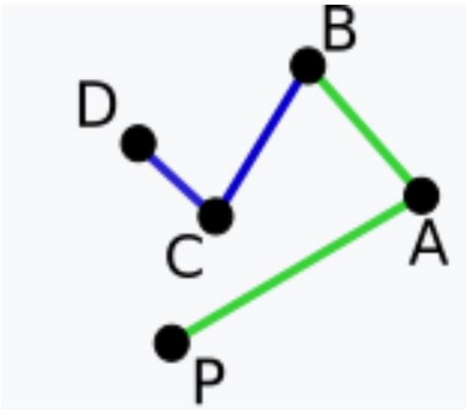
Step 1: 選擇內點（每點 x 和 y 值的平均）。

Step 2: 計算每點和內點的角度並依角度逆時針排序。

Step 3: 每三點可形成三角形，檢查其是否為凹角（內角>180°），如果是凹角，則將中間點移除。

Step 4: 如果回到起點，則將所有點連線即為 convex hull，否則繼續步驟(2), (3)。

如下圖：內角∠DCB > 180°為凹角，則將中間點 C 刪除。



2.

common subsequence: 給兩字串 X 和 Y，分別刪除 X 和 Y 中某些 字元(也可不刪)後，使兩結果完全相同

LCS: 找到最長 common subsequence 之長度

假設 X = abcb , Y = acb

則 common subsequence = ab, ac, cb, acb

LCS = acb

3.

$T(n)$

$= 2T\left(\frac{n}{2}\right) + O(n) \leq 2\left(2T\left(\frac{n}{4}\right) + O\left(\frac{n}{2}\right)\right) + cn$

$= 4T\left(\frac{n}{4}\right) + 2 * \frac{cn}{2} + cn$

$= 4T\left(\frac{n}{4}\right) + 2cn$

$= \cdots$

$= 2^k T\left(\frac{n}{2^k}\right) + kcn\ , n = 2^k\ k = logn$

$= n + cnlogn$

$= O(nlogn)$

4.

使用 Dijkstra's Algorithm:

<u>Input</u>: 點集合 V，起點為 S 和 cost matrix

Step1:從 S 出發計算此點走一步可到的節點之距離，並將此距離寫入 cost matrix 中，若走一步到不了其距離設為無限大。

Step2:選出一個距離最小的節點 k，找出經過這個點的距離，當 cost[i][k]+cost[k][j] < cost[i][j]則代表經過 k 點的距離比原先更短所以將距離更新為 cost[i][k]+cost[k][j]，反之則不更新。

Step3:重覆做 step2 直到所有點都走過就結束。

Time complexity : $O(n^2)$

每次更新要看 O(n)，總共要執行 n 次，時間複雜度為 $n*O(n) = O(n^2)$，其中 n=|V|

5.

Input : n points and a straight line y = 0.

Output: The constrained center on the straight line y = 0.

Step 1: If n is no more than 2, solve this problem by a brute force method.

Step 2: Form disjoint pairs of points$(p_1,p_2),(p_3,p_4)$, …, (p_{n-1}, p_n). If there are odd number of points, just let the final pair be (p_n, p_1).

Step 3: For each pair of points, (p_i, p_{i+1}), find the point $x_{i,i+1}$ on the line y = 0 such that $d(p_i, x_{i,i+1}) = d(p_{i+1},x_{i,i+1})$.

Step 4: Find the median of the $\left\lfloor \frac{n}{2} \right\rfloor$ $x_{i,i+1}$'s. Denote it as x_m.

Step 5: Calculate the distance between p_i and x_m for all i. Let p_j be the point which is farthest from x_m. Let x_j denote the projection of p_j onto y = 0. If x_j is to the left (right) of x_m, then the optimal solution, x^*, must be to the left (right) of x_m.

Step 6: If $x^* < x_m$, for each $x_{i,i+1} > x_m$, prune the point p_i if p_i is closer to x_m than p_{i+1}, otherwise prune the point p_{i+1};If $x^* > x_m$, do similarly.

Step 7: Go to Step 1.

6.

$S(a_i)$: the size of item a_i

OPT: # of bins used in an optimal solution

m: # of bins used in the first-fit algorithm

$C(B_i)$: the sum of the sizes of a_j's packed in bin Bi in the first-fit algorithm

OPT >= $S(a_1)+S(a_2)+\ldots.+S(a_n)$

$C(B_i) + C(B_{i+1}) > 1$

$C(B_1)+C(B_2)+\ldots.C(B_m) > m/2$

$m/2<C(B_1)+C(B_2)+\ldots.C(B_m)=S(a_1)+S(a_2)+\ldots.+S(a_n)<=OPT$

m < 2 OPT

First fit' s bin 的數量 < 2 OPT' s Bin 的數量

7.

instance of clique decision:

G = (V, E), clique Q of size k ($Q \subseteq V$)

instance of node cover decision:

G'=(V, E'), where E'={(u, v) | $u \in V$, $v \in V$ and $(u, v) \notin E$},

node cover N of size n-k, N = V - Q

(=>)

由於 clique Q 的任兩點均有 edges，亦即在 E 中有 edge(u,v) for u,v $\in$ Q。而在 E'中，則不會有 edge (u, v) for u, v $\in$ Q，亦即每個 edge 至少有一個點在 V - Q 中。所以 N=V - Q 為 G'的 node cover，size = |V| - k

(<=)

在 G'中，任一 edge (u, v) 至少滿足 $u \in N$ or $v \in N$。換言之，若 u, v $\in$ Q，則在 G' 沒有存在 edge。因此，若 u, v $\in$ Q，則在 G 必定有 edge，亦即 Q 是一個 complete subgraph。

所以在 G 中會有 clique = V – N，size = |V| - (|V| - k) = k

So, clique decision problem polynomially reduces to node cover decision problem

8.

<u>Transpose heuristics</u>: The node that is accessed is swapped with its predecessor.

<u>move-to-front heuristics</u>: The recently searched item is moved to the front of the list.

Count heuristics: Most frequently accessed node is kept at the head of the list.