Department of Computer Science and Engineering
National Sun Yat-sen University
Design and Analysis of Algorithms - Final Exam., Dec. 26, 2023

1. Explain each of the following terms. (20%)
   (a) NP-hard, NP-complete
   (b) quadratic nonresidue problem
   (c) skew heap
   (d) Euler circuit of a graph
   (e) pairwise independent property of move-to-the-front in sequential search
2. In the 0/1 *knapsack* problem, there are n objects with knapsack capacity M, where the profit of each object i is denoted by p_i and the weight is denoted by w_i, $1 \leq i \leq n$. Please present the *dynamic programming* formula for solving the 0/1 knapsack problem. In the formula, let $f_i(Q)$ be the maximum profit obtained by objects $1,2,3,\ldots,i$ with capacity Q. (10%)
3. Given two sets A and B, each consisting of n integers, design an efficient algorithm to check whether A is equal to B or not. And analyze the time complexity of your algorithm. Note that your algorithm should be in $O(n\log n)$ time. (10%)
4. Explain the searching strategies: *depth-first search*, *breadth-first search* and *best-first search*. What data structures are used in these strategies? (12%)
5. Explain the common properties among the following problems: *convex hull*, *one-center*, *constrained one-center*, *rectilinear m-center*. And give the differences between them. (12%)
6. Present an algorithm for solving the shortest path (from a single source) problem on a graph. Analyze the time complexity of your algorithm. (12%)
7. In the *bottleneck traveling salesperson* problem, the goal is to minimize the longest edge in the solution. Assume that $h(G)$ can determine whether a graph G has a Hamiltonian cycle or not. Please present a greedy method for solving this problem utilizing $h(G)$. Note that there is no need to design $h(G)$; you can directly call $h(G)$. (12%)
8. Prove that the *sum of subset decision* problem polynomially reduces to the *partition* problem. (12%)

Answers

1.

(a)

NP-hard: the class of problems to which every NP problem reduces.

NP-complete (NPC): the class of problems which are NP-hard and belong to NP.

(b)

GCD(x, y) = 1, y is a quadratic residue mod x if z^2 = y mod x for some z, 0 < z < x, GCD(x, z) = 1, and y is a quadratic nonresidue mod x if otherwise.

(c)

Skew heaps may be described with the following recursive definition:

- A heap with only one element is a skew heap.
- The result of *skew merging* two skew heaps sh_1 and sh_2 is also a skew heap.

(d)

A circuit that uses every edge of a graph exactly once.

(e)

For any sequence S and all pairs P and Q, # of interword comparisons of P and Q is exactly # of interword comparisons made for the subsequence of S consisting of only P's and Q's.

2.

$f_i(Q) = \max\{ f_{i-1}(Q), f_{i-1}(Q-W_i)+P_i \}$

$f_0(0) = f_i(0) = f_0(Q) = 0$ for $1 \le i \le n$, $0 < Q \le M$

3.

分別將 set A 和 set B 裡面的整數由小到大做排序，set A = {a1, a2, ..., an} 且 set B = {b1, b2, ..., bn}，接著進行比較，若 a1= b1, a2 = b2, ..., an = bn，則 set A 等於 set B，排序需要 O(nlogn)的時間，而比較每一項是否相等需要 O(n)的時間，Time complexity 為 O(nlogn) + O(n) = O(nlogn)

4.

Depth-first search: DFS is a traversal approach in which the traverse begins at the root and explores as far as possible along each branch before backtracking.. DFS uses Stack data structure.

Breadth-first search: BFS is a traversal approach , which explores all the neighboring nodes at the same level before moving to the next level.. BFS uses Queue data structure.

Best-first search: The idea of Best-first search is to use an evaluation function to decide which adjacent is most promising and then explore. Best-first search uses priority queue (heap) data structure.

5.

Common properties: 以最小的範圍，將全部的點包圍起來。

不同之處：

Convex Hull:

以凸多邊形，包含全部的點。沒有中心點的概念。

One-center :

以一個最小的圓，包含全部的點。圓心是中心點。

Constrain one-center :

以一個最小的圓，包含全部的點。圓心是中心點，但圓心須在所給定的一條直線上。

Rectilinear m-center:

以 m 個正方形(邊為垂直於 x 與 y 軸)，包含全部的點，邊長為最小。正方形正中間為是中心點。

6.

Dijkstra’s Algorithm:

Input: 點集合 V, 起點 S, cost matrix E

Step1: 設計一個一維矩陣 dis[]用來記錄 S 到各個點當前的最短路徑，若無路徑則設無窮大。

Step2: 從 dis[]挑選沒被選過的點中與 S 距離最小的點(i)，找出與該點相連接的點(j)，並進行鬆弛操作更新 dis[] : dis[j] = min{dis[j], dis[i]+E[i][j]}

Step3: 重覆做 step2 直到所有點都走過就結束。

Time Complexity:

在 Step2 進行 dis[]更新時會花 O(n)時間，總共會進行 n 輪更新，因此時間複雜度維 $O(n^2)$, 其中 n=|V|

7.

G = (V, E)，若 G 中的 *Hamiltonian cycle* 有解，則此 cycle 中的 longest edge 可能為 E 中的最長邊(u,v)，因此當 h(G)為 true，從 G 中刪除(u,v)，G’=(V,E’), E’=E-(u,v)，重新判斷 h(G’)是否為 true，重複上述步驟直到 *Hamiltonian cycle* 無解，最後一個有解的 G 的 *Hamiltonian cycle* 的 longest edge 最小

8.

An instance of the Subset Sum Problem (SSP) is given with a set of integers A = { a_1, a_2, ..., a_n } and a target sum C. The SSP problem is to decide whether there exists a subset of A whose sum is exactly equal to C.

Given a set of integers S, the Partition Problem (PP) problem is to decide whether B can be partitioned into two disjoint subsets B_1 and B_2 such that the sum of the elements in B_1 is equal to the sum of the elements in B_2.

Reduce SSP to PP:

1. Given an instance of SSP with a set of integers A = { a_1, a_2, ..., a_n } and a target sum C.
2. For PP, construct an instance of with a new set B = { b_1, b_2, ...,b_n, b_{n+1} b_{n+2} }, where each $b_i = a_i$ for $1 \leq i \leq n$, and b_{n+1} = C+1 and b_{n+2} = (Σ from i=1 to n a_i) + 1 - C.

The sum of all elements in B is (Σ from i=1 to n b_i)+ b_{n+1} + b_{n+2} = (Σ from i=1 to n a_i)*2 + 2.

We want to prove that there exists a solution of SSP (a subset S ⊆ A such that Σ for a_i in S a_i)= C if and only if there exists a partition in PP (B can be partitioned into two subsets whose sums are equal).

Subset Sum Problem (SSP) => Partition Problem (PP)

If there is a subset S ⊆ A whose sum is equal to C, then there exists a partition of B into B_1={ b_i | $a_i \in$ S } ∪ { b_{n+2} } and B_2={ b_i | $a_i \notin$ S } ∪ { b_{n+1} }. The sum of elements in B_1 will be (Σ for a_i in S a_i) + b_{n+2} = C + (Σ from i=1 to n a_i) + 1 - C = (Σ from i=1 to n a_i)+ 1, which is equal to the sum of B_2, (Σ for ai ∉ S a_i) + b_{n+1} = (Σ from i=1 to n a_i) - C + C + 1 = (Σ from i=1 to n a_i)+ 1.

Partition Problem (PP) => Subset Sum Problem (SSP)

Conversely, if there exists a partition of set B into two subsets with equal sums, because b_{n+1} and b_{n+2} cannot be in the same subset (since their sum is greater than the sum of all other elements), one of them must be in subset B_1 and the other in B_2. Without loss of generality, assume $b_{n+1} \in B_2$ and $b_{n+2} \in B_1$. As the sum of all elements in B is (Σ from i=1 to n a_i)*2 + 2, and the sums of B_1 and B_2 are equal, the sum of the elements is B_1 is (Σ from i=1 to n a_i) + 1.

We have b_{n+2} = (Σ from i=1 to n a_i) + 1 - C. So the sum of the other elements from A in B_1 must be C, proving the existence of a solution to the SSP.