Habib University
Homework 02
CS 412 (Algorithms: Design and Analysis)

Spring 2024. Total Points: 100

March 26, 2024

1. 10 points In the lecture, we discussed the 0/1 knapsack optimization problem and its dynamic programming solution. Let us define a rooted binary tree where each node denotes a partial solution of a given knapsack problem and is specified as a quadruple as follows:

   - A set of items to be taken in the knapsack.
   - The total value [thus far] of the items to be taken in the knapsack.
   - The list of items for which a decision has not been made.
   - The remaining capacity in the knapsack.

   Construction: The rooted binary tree is a decision tree built top-down, starting with the root.

   At each internal (non-leaf) node, we consider one element at a time from the 'still-to-be-considered' list of items. If there is still room for that item in the knapsack, we create a left child node that shows the consequence of choosing that item. On the other hand, a right child node shows the consequence of not choosing that item. The process is applied recursively until either the knapsack is full, or there are no more items left to be considered, i.e., we arrive at the leaf node. A leaf node with the largest value states the optimal solution.

   Now, imagine a burglar carrying an empty knapsack that can carry a maximum weight of 5 kg. Draw a left-first depth-first decision tree (based on the above definition) for the following list of items in the table below. Also, state the optimal solution along with a list of items that end up in the knapsack in the optimal solution. Label the nodes in the left-first depth-first order as natural numbers.

   The root node is a quadruple of labelled as $\mathbf{0} :< \{\}, [\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}], \mathbf{0}, \mathbf{5} >$

Item	Value	Weight
A	6	3
B	7	3
C	8	2
D	9	5

2. Imagine a city with n buildings labeled as $1, ..., n$, with $n-1$ two-way streets. A street $s_i = (a_i, b_i)$ connects buildings a_i and b_i, where $1 \leq a_i \leq b_i \leq n$, and $1 \leq i \leq n-1$.

   We know that starting from any building, we can reach any other building walking through the city streets. Setting up a lamp in front of a building lights all the streets adjacent to it. We need to find the minimum number of lamps required to light all the city streets.

   (a) 10 points Design an $O(n)$ dynamic programming algorithm to solve this problem.

(b) [10 points] Justify the runtime complexity of your algorithm.

3. [10 points] **CLRS: Longest palindrome subsequence**
A palindrome is a nonempty string over some alphabet that reads the same forward and backward. Examples of palindromes are all strings of length 1, civic, racecar, and aibohphobia (fear of palindromes).

   Give an efficient algorithm to find the longest palindrome that is a subsequence of a given input string. For example, given the input character, your algorithm should return carac. What is the running time of your algorithm?

4. Let $G_1 = (V, E)$ and $G_2 = (V, E)$ be two graphs with the same structure (i.e., same vertices and same edges). Let the costs of edges in G_1 be distinct and non-negative and the costs of edges in G_2 be the squares of costs of their corresponding edges in G_1.

   (a) [10 points] Assume G_1 and G_2 are undirected graphs. Prove or disprove that Prim's algorithm on G_1 and G_2 would result in the same minimum spanning tree if we start with the same vertex.

   (b) [10 points] Now, assume G_1 and G_2 are directed graphs then prove or disprove that the shortest paths in G_1 and G_2 from some vertex s to any other vertex t by Dijkstra's algorithm are the same.

5. [5 points] Is it possible for a string to have a **min-edit** distance of 0 to its reverse? Justify your answer succinctly.

6. [10 points] **From Dasgupta et al.:** A certain string-processing language offers a primitive operation which splits a string into two pieces. Since this operation involves copying the original string, it takes n units of time for a string of length n, regardless of the location of the cut. Suppose, now, that you want to break a string into many pieces. The order in which the breaks are made can affect the total running time. For example, if you want to cut a 20-character string at positions 3 and 10, then making the first cut at position 3 incurs a total cost of $20 + 17 = 37$, while doing position 10 first has a better cost of $20 + 10 = 30$. Give a dynamic programming algorithm that, given the locations of m cuts in a string of length n, finds the minimum cost of breaking the string into $m + 1$ pieces.

7. [10 points] **From Dasgupta et al.:** A long string consists of the four characters A, C, G, T. They appear with frequency $31\%, 20\%, 9\%$, and 40% respectively. What is the Huffman encoding of these four characters?

8. [5 points] Prof. Habib claims that solving the fractional knapsack problem using dynamic programming is an overkill. Do you agree with this statement? Justify your claim, i.e., with a supporting argument or a counterexample.

9. [10 points] **From Dasgupta et al.:** Alice wants to throw a party and is deciding whom to call. She has n people to choose from, and she has made up a list of which pairs of these people know each other. She wants to pick as many people as possible, subject to two constraints: at the party, each person should have at least five other people whom they know and five other people whom they don't know. Give an efficient algorithm that takes as input the list of n people and the list of pairs who know each other and outputs the best choice of party invitees. Give the running time in terms of n.