CSE 102: Spring 2021

Homework Assignment # 4

Greedy Algorithms

(18 points)

1. *Dijkstra's Single-Source Shortest Path Problem* : We are given a directed graph $G = (V, E)$ with n nodes and m edges. Each edge has a non-negative length $e_l \geq 0$. Length of a path between two nodes is the sum of the lengths of all edges in the path. Single-source shortest path problem is to find the path and the length of the shortest path between a given source (node) to every other node in the graph. If a path does not exist, then the length is inf. This problem is known as single-source shortest path problem.

   Under the assumption that each edge length is non-negative, that is, $e_l \geq 0$, Dijkstra designed a simple greedy algorithm to solve this problem. The time complexity of this algorithm varies depending upon the data structure implementation – array, binary heap, d-ary heap, or Fibonacci heap.

   This is a practical problem arising in many applications where the length represents cost or time.

   (a) (1 point): Clearly state the greedy strategy. Explain briefly.

   (b) (2 points): Exercise 4.1 of DPV. This exercise essentially asks you to demonstrate how Dijkstra's algorithm works on a specific graph.

2. *Minimal Spanning Tree: Kruskal's Algorithm* We are given an undirected connected (there is a path between every pair of nodes) graph $G = (V, E)$ with n nodes and m edges. Each edge has a positive cost $c_e > 0$. The problem is to find a subset of edges so that the resulting graph is connected and the total cost of all the edges combined is the smallest possible. It is easy to observe that the resulting graph must be a tree (a tree is an undirected graph that is connected and acyclic,

because if not then remove one of the edges of the cycle and the remaining graph is still connected and has a lower cost). This problem is referred to as Minimal Spanning Tree problem.

Kruskal designed a greedy algorithm that runs in $O(m \log n)$ time.

(a) (1 point): Clearly state the greedy strategy. Explain briefly.

(b) (2 points): Exercise 5.1 of DPV. This exercise essentially asks you to demonstrate how Kruskal's algorithm works on a specific graph. You do *not* need to specify the cut.

3. *Minimal Spanning Tree: Prim's Algorithm* Prim designed a different greedy algorithm that also runs in $O(m \log n)$ time.

   (a) (1 point): Clearly state the greedy strategy. Explain briefly.

   (b) (2 points): Exercise 5.2 (a) of DPV. This exercise essentially asks you to demonstrate how Prim's algorithm works on a specific graph.

4. Consider the following *office hours* problem faced by a professor. n students (numbered 1 to n) show up at the start of office hours with different questions. Each student i has one question that will take time t_i to answer, and will leave as soon as that question is answered. Given the n values of the t_i's, find an order in which to answer the students' questions such that the sum of the times spent by each student in the office is minimized.

   (a) (1 point) Describe a greedy algorithm that correctly finds optimal solutions. Explain briefly.

   (b) (3 points) Prove that your algorithm always returns an optimal solution. [Hint: Use exchange argument. Discussed in Brassard and Bratley's Book.]

5. *Rod Cutting Problem* described below. (1 point) Provide a Counter-Example.

6. *Coin Changing Problem* described below. (1+2+1) points

CSE 102
Homework Assignment 4

1. (Read the **Rod-Cutting Problem** in section 15.1 pp. 360-369 of CLRS 3rd edition. This is problem 15.1-2 on p. 370.) Show, by means of a counterexample, that the following "greedy" strategy does not always determine an optimal way to cut rods. Define the *density* of a rod of length i to be p_i/i, that is, its value per inch. The greedy strategy for a rod of length n cuts off a first piece of length i, where $1 \leq i \leq n$, having maximum density. It then continues by applying the greedy strategy to the remaining piece of length $n - i$.

2. Recall the coin changing problem: Given denominations $d = (d_1, d_2, \ldots, d_n)$ and an amount N to be paid, determine the number of coins in each denomination necessary to disburse N units using the fewest possible coins. Assume that there is an unlimited supply of coins in each denomination.

   a. Write pseudo-code for a greedy algorithm that attempts to solve this problem. (Recall that the greedy strategy doesn't necessarily produce an optimal solution to this problem. Whether it does or not depends on the denomination set d.) Your algorithm will take the array d as input and return an array G as output, where $G[i]$ is the number of coins of type i to be disbursed. Assume the denominations are arranged by increasing value $d_1 < d_2 < \cdots < d_n$, so your algorithm will step through array d in reverse order. Also assume that $d_1 = 1$ so any amount can be paid.

   b. Let $d_i = b^{i-1}$ for some integer $b > 1$, and $1 \leq i \leq n$, i.e. $d = (1, b, b^2, \ldots, b^{n-1})$. Does the greedy strategy always produce an optimal solution in this case? Either prove that it does, or give a counter-example.

   c. Let $d_1 = 1$ and $2d_i \leq d_{i+1}$ for $1 \leq i \leq n - 1$. Does the greedy strategy always produce an optimal solution in this case? Either prove that it does, or give a counter- example.