Algorithm Design and Analysis

Assignment 3

Deadline: Apr 18, 2022

1. (25 points) Suppose you are a driver, and you plan to drive from A to B through a highway with distance D. Since your car's tank capacity C is limited, you need to refuel your car at the gas station on the way. We are given n gas stations on the highway with surplus supply. Let $d_i \in (0, D)$ be the distance between the starting point A and the i-th gas station. Let p_i be the price for each unit of gas at the i-th gas station. Suppose each unit of gas exactly supports one unit of distance. The car's tank is empty at the beginning, and the 1-st gas station is at A. Design efficient algorithms for the following tasks.

   (a) (5 points) Determine whether it is possible to reach B from A.

   (b) (20 points) Minimized the gas cost for reaching B.

   Please prove the correctness of your algorithms and analyze their running times.

2. (25 points) Given a constant $k \in \mathbb{Z}^+$, we say that a vertex u in an undirected graph *covers* a vertex v if the distance between u and v is at most k. In particular, a vertex u covers all those vertices that are within distance k from u, including u itself. Given an undirected *tree* $G = (V, E)$ and the parameter k, consider the problem of finding a minimum-size subset of vertices that covers all the vertices in G.

   (a) (10 points) Design an efficient algorithm for the problem above with $k = 1$.

   (b) (15 points) Design an efficient algorithm for the problem above with general k.

   Please prove the correctness of your algorithms and analyze their running times.

3. (30 points) In the class, we learned Kruskal's algorithm to find a minimum spanning tree (MST). The strategy is simple and intuitive: pick the best legal edge in each step. The philosophy here is that local optimal choices will yield a global optimal. In this problem, we will try to understand to what extent this simple strategy works. To this end, we study a more abstract algorithmic problem of which MST is a special case.

   Consider a pair $M = (U, \mathcal{I})$ where U is a finite set and $\mathcal{I} \subseteq \{0,1\}^U$ is a collection of subsets of U. We say M is a *matroid* if it satisfies

   - (**hereditary property**) $\mathcal{I}$ is nonempty and for every $A \in \mathcal{I}$ and $B \subseteq A$, it holds that $B \in \mathcal{I}$.
   - (**exchange property**) For any $A, B \in \mathcal{I}$ with $|A| < |B|$, there exists some $x \in B \setminus A$ such that $A \cup \{x\} \in \mathcal{I}$.

   Each set $A \in \mathcal{I}$ is called an *independent set*.

   (a) (6 points) Let $M = (U, \mathcal{I})$ be a matroid. Prove that maximal independent sets are of the same size. (A set $A \in \mathcal{I}$ is called *maximal* if there is *no* $B \in \mathcal{I}$ such that $A \subsetneq B$.)

   (b) (6 points) Let $G = (V, E)$ be a simple undirected graph. Let $M = (E, \mathcal{S})$ where $\mathcal{S} = \{F \subseteq E \mid F \text{ does not contain a cycle}\}$. Prove that M is a matroid. What are the maximal sets of this matroid?

   (c) (6 points) Let $M = (U, \mathcal{I})$ be a matroid. We associate each element $x \in U$ with a nonnegative weight $w(x)$. For every set of elements $S \subseteq U$, the weight of S is defined as $w(S) = \sum_{x \in S} w(x)$. Now we want to find a maximal independent set with maximum weight. Consider the following greedy algorithm.

Algorithm 1 Find a maximal independent set with maximum weight

Input: A matroid $M = (U, \mathcal{I})$ and a weight function $w : U \to \mathbb{R}_{\geq 0}$.

Output: A maximal independent set $S \in \mathcal{I}$ with maximum $w(S)$.

```
S ← ∅
Sort U into decreasing order by weight w
for x ∈ U in decreasing order of w:
    if S ∪ {x} ∈ I:
        S ← S ∪ {x}
    endif
endfor
return S
```

   Now we consider the first element x the algorithm added to S. Prove that there must be a maximal independent set $S' \in \mathcal{I}$ with maximum weight containing x.

(d) (6 points) Prove that the greedy algorithm returns a maximal independent set with maximum weight. (Hint: Can you see that Algorithm 1 is just a generalization of Kruskal's algorithm?)

(e) (6 points) Let $U \subseteq \mathbb{R}^n$ be a finite collection of n-dimensional vectors. Assume $m = |U|$ and we associate each vector $\mathbf{x} \in U$ a positive weight $w(\mathbf{x})$. For any set of vectors $S \subseteq U$, the weight of S is defined as $w(S) = \sum_{\mathbf{x} \in S} w(\mathbf{x})$. Design an efficient algorithm to find a set of vectors $S \subseteq U$ with maximum weight and all vectors in S are linearly independent.

4. (25 points) Recall the makespan minimization problem in the class, where we need to schedule n jobs on m machines to minimize the makespan. Let us consider another variant. Given n jobs and a fixed makespan C, can we minimize the number of machines to complete every job before C? Design a Greedy based algorithm and analyze its performance w.r.t. the optimal solution.

   (a) (20 points) Design a Greedy based algorithm. For any input, let ALG be the solution of your algorithm, OPT be the optimal solution, prove $\mathsf{ALG} \leq 2 \cdot \mathsf{OPT} - 1$. Hint: use the idea of the Greedy algorithm in Makespan Minimization.

   (b) (5 bonus points) Design a smarter algorithm, so that we can prove $\mathsf{ALG} \leq \Gamma \cdot \mathsf{OPT} + O(1)$ with some $\Gamma < 2$. Write down your idea and try to prove. You can make some assumptions to help your proof if you can not prove it in general.

5. How long does it take you to finish the assignment (including thinking and discussion)?

   Give a rating (1,2,3,4,5) to the difficulty (the higher the more difficult) for each problem.

   Do you have any collaborators? Please write down their names here.