Algorithm Design and Analysis (Fall 2023)

Assignment 3

Deadline: Dec 12, 2023

1. (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. Design a polynomial time algorithm for this problem. Prove the correctness of your algorithm and analyze its running time. You will receive 15 points if you can solve the problem for $k = 1$.

Algorithm

1.Sort all vertices by its level from low to high and store them in an array $A = \{v_1, v_2, ..., v_n\}$.

2.Define a bool array $Marked[n]$ and set all its values to false.

3.Define an array $result \leftarrow \emptyset$ to store the vertices in the minimum-size subset.

4.For each vertex v_i in A:

if($marked[i] == true$) continue.

if(its $k-th$ ancestor exists):

add its $k-th$ ancestor to $result$.

$marked[v_i's\ k-th\ ancestor\ and\ all\ of\ its\ children] \leftarrow true$.

else:

add the root of the tree to $result$.

Correctness

Let R^* to be an optimal result and R^n to be the answer of the algorithm containing n vertices. We aim to prove $R^n \subseteq R^*$. Below I prove it by induction:

Base step: v_1 means the $k-th$ ancestor of the lowest vertex or the root. If $v_1 \nsubseteq R^*$ and one of v_1's child $v_1' \subseteq R^*$, we can always swap v_1' with v_1 in R^*. In this way, the overall coverage remains or increases, thus making R^* to another optimal result $R^{*\prime}$.

Inductive step: suppose $R^{i-1} \subseteq R^*$.

If v_i is the root of the tree. It means that i vertices covers all vertices. Since $R^{i-1} \subseteq R^*$, R^i is an optimal result, thus $R^i \subseteq R^*$. If v_i is the $k-th$ ancestor of the lowest vertex v_i'. Since in R^*, we must choose one vertex from v_i to all of its children in $i-th$ round to cover all those vertices. If $v_i \in R^*$, then $R^i \subseteq R^*$. Else if $v_i \notin R^*$, instead, one of its child $v_i' \in R^*$. According to the base step, we can always swap v_i with v_i' to form another optimal result $R^{*\prime}$ satisfying $R^i \subseteq R^{*\prime}$.

Therefore, $R^n \subseteq R^*$.

Time Complexity

Step 1: Knowing the level of each vertex takes $O(|V|)$ and sorting them takes $O(|V|log|V|)$.
Step 2&3: $O(|V|)$.
step 4: $|V|$ rounds. Each round takes $O(|V|)$.
Therefore, the overall time complexity is $O(|V|^2)$.

2. (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.

   (a) for each d_i from $i = 1, ..., n$, check if $d_{i+1} - d_i \leq C$, $i = 1, 2, ...n - 1$. Then check if $D - d_n \leq C$. If so, it is possible to reach B from A.

   (b) **Algorithm**

   Repeat at every gas station: if there is a gas station whose price is cheaper than now within reach(distance $\leq C$), then refuel the car until the car can just drive to that gas station. If there is no gas station whose price is cheaper than now within reach, then refuel the car to C.

   Correctness

   Let $R^* = \{r'_0, r'_1, r'_2, ..., r'_n\}$ be the optimal solution and r_i be the amount of gas fueled in $i - th$ gas station(0 means A). Let $R^i = \{r_0, r_1, ..., r_i\}$ be the solution by applying the algorithm. We aim to show $R^i \subseteq R^*$ for every $i = 0, 1, 2, ..., n$. Below I prove it by induction:

   Base step: If the distance between A and the next cheaper station $\leq C$:

   for $i = 0$, if $r_0 \leq r'_0$, the cost of the algorithm is less than or equal to the optimal solution while making sure we can add gas in the next cheaper station to let the amount of gas is equal to the optimal solution at that station, thus making R^* to another optimal solution. If $r_0 > r'_0$. In this case, the optimal solution should add gas in gas station other than A with higher price to just reach the next cheaper station for A. So we can transform the gas added in expensive station to gas added in cheaper station A, thus decreasing the cost to reach the next cheaper station. So $R^1 \subseteq R^{*\prime}$, $R^{*\prime}$ is another optimal solution.

   If the distance between A and the next cheaper station $> C$:

   In this case, $r_0 \geq r'_0$. Similarly, we can increase r'_0 so that we can reach the next cheaper station with less cost while we can add the gas there until the amount of gas = optimal solution, thus forming another optimal solution $R^{*\prime}$ satisfying $R^0 \subseteq R^{*\prime}$.

Inductive step: suppose $R^i \subseteq R^*$. We need to show $R^{i+1} \subseteq R^*$.
In this way, we can discuss this the same as the base step. Suppose the car is at the $i+1-th$ station and the amount of gas is the same as the optimal solution. If the next cheaper station is within reach(distance$\leq C$):
If the distance between $i+1-th$ station and the next cheaper station $\leq C$:
if $r_{i+1} \leq r'_{i+1}$: It is similar to the base step, we can making another optimal solution and $R_{i+1} \subseteq R^{*\prime}$.
if $r_{i+1} > r'_{i+1}$: Similarly, we can decrease the gas added in expensive station and add it in $i+1-th$ station, thus making another optimal solution.
If the distance between $i+1-th$ station and the next cheaper station $> C$:
Similarly, $r_{i+1} \geq r'_{i+1}$. Increase r'_{i+1} so that we can reach the next cheaper station with less cost, thus forming another optimal solution satisfying $R^{i+1} \subseteq R^{*\prime}$.
Therefore, the algorithm is correct.

Time Complexity

In every station, we need to know the first cheaper station within reach. It takes $O(n)$. And there is n rounds, so the overall time complexity is $O(n^2)$.

3. (25 points) Given a ground set $U = \{1, \ldots, n\}$, a *set function on* U is a function $f : \{0,1\}^U \to \mathbb{R}$ that maps a subset of U to a real value. A set function f is *submodular* if

$$f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T)$$

holds for any $S, T \subseteq U$ with $S \subseteq T$ and any $v \in U \setminus T$. We make the following assumptions on a submodular set function f:

- Nonnegative: $f(S) \geq 0$ for any $S \subseteq U$; you can assume $f(S)$ is always a rational number.
- Monotone: $f(S) \leq f(T)$ for any $S, T \subseteq U$ with $S \subseteq T$.
- $f(S)$ can be computed in a polynomial time with respect to $n = |U|$.

Given a positive integer $k > 0$ as an input, the goal is to find $S \subseteq U$ that maximizes $f(S)$ subject to the cardinality constraint $|S| \leq k$. Design a polynomial time $(1 - 1/e)$-approximation algorithm for this maximization problem. Prove that the algorithm you design runs in a polynomial time and provides a $(1 - 1/e)$-approximation.

Algorithm

1.Define a set S.

2.For k loops:

find $v \in U \setminus S$ that maximizes $f(S \cup \{v\}) - f(S)$.

$S \leftarrow S \cup \{v\}$.

Time Complexity

The algorithm runs at most n rounds and every rounds takes $O(n)$. So the overall time complexity is $O(n^2)$. Therefore the algorithm runs in a polynomial time.

Approximation

Let $S^* = \{v_1, v_2, ..., v_k\}$ be any collection of k elements in U. Let $S = \{v'_1, v'_2, ..., v'_l\}$ be the output of greedy after l iterations.

Lemma: $f(S) \geq (1 - (1 - \frac{1}{k})^l) f(S^*)$.

If the Lemma is true. Then for optimal S^*, we have $f(S) \geq (1 - (1 - \frac{1}{k})^k) f(S^*) \geq (1 - \frac{1}{e}) f(S^*)$, which indicates $(1 - \frac{1}{e})$ approximation. Below is the proof of the lemma.

Prove lemma by induction:

Base step $l = 1$: By greedy nature, $f(S_1 = \{v'_1\}) \geq f(\{v_i\})$ for all v_i. Thus, $f(S_1) \geq \frac{1}{k} \sum_{i=1}^{k} f(\{v_i\}) \geq \frac{1}{k} f(S^*) = (1 - (1 - \frac{1}{k})^1) f(S^*)$.

Inductive step: Suppose $f(S) \geq (1 - (1 - \frac{1}{k})^l) f(S^*)$ for $l = 1, 2, ...i$. We aim to show $f(S) \geq (1 - (1 - \frac{1}{k})^l) f(S^*)$ for $l = i + 1$. First, we have $f(S \cup \{v'_{i+1}\}) - f(S) \geq f(S \cup \{v_{i+1}\}) - f(S) \geq \frac{1}{k} \sum_{i=1}^{k} f(S \cup \{v_i\}) - f(S) \geq \frac{1}{k}(f(S \cup S^*) - f(S)) \geq \frac{1}{k}(f(S^*) - f(S))$ (monotonicity of f). So, we have $f(S_{i+1}) \geq \frac{1}{k} f(S^*) + (1 - \frac{1}{k}) f(S)$ (rearranging inequal-

ity) $\geq \frac{1}{k}f(S^*)+(1-\frac{1}{k})(1-(1-\frac{1}{k})^i)f(S^*)$ (induction hypothesis)$= (1-(1-\frac{1}{k})^{i+1})f(S^*)$. Therefore, the lemma is true, thus proving the approximation.

4. (25 points) Given an undirected graph $G = (V, E)$, a *matching* M is a subset of edges such that no two edges in M share an endpoint. The *maximum matching problem* takes the graph $G = (V, E)$ as the input and outputs a matching M with the maximum size $|M|$. Consider the following greedy algorithm.

   - initialize $M \leftarrow \emptyset$
   - while there exists $e \in E$ such that $M \cup \{e\}$ is a valid matching:
     - update $M \leftarrow M \cup \{e\}$
   - endwhile
   - return S

   (a) (20 points) Prove that this is a 0.5-approximation algorithm.

   (b) (5 points) Provide a tight example showing that this is not a $(0.5+\varepsilon)$-approximation algorithm for any $\varepsilon > 0$.

   (a)Let M^* be the maximum matching and M be the matching by greedy algorithm. We aim to prove $|M| \geq 0.5|M^*|$. For each edge in M, there is only two case:
   Case 1: the edge is in M^*, then it does not infect $|M|$.
   Case 2: the edge is not in M^*. Suppose the edge is (a, b). Then, there could be at most two edges in $|M^*|$ adjacent to (a, b), such as (a, c) and (b, d). In this case, it will infect $|M|$. Meanwhile, $|M^*|$ can't include an edge which has no endpoint the same as an edge in $|M|$ because the greedy algorithm will choose it since it is a valid edge if the edge has no endpoint the same as an edge in $|M|$.
   If every edge in M satisfies the case 2, then $|M^*|$ could be at most $2|M|$. Therefore, $|M| \geq 0.5|M^*|$. It is a 0.5-approximation algorithm.
   (b)Consider the graph below:

   1——2——3——4

   The optimal matching $M^* = \{(1, 2), (3, 4)\}$. However, if we apply the greedy algorithm, suppose we first choose $(2, 3)$, then there is no other valid edge. In this case, $|M| = 1$ while $|M^*| = 2$. It is a 0.5-approximation algorithm.

5. How long does it take you to finish the assignment (including thinking and discussion)? Give a score (1,2,3,4,5) to the difficulty. Do you have any collaborators? Please write down their names here.

   Saturday's afternoon, Sunday's morning, afternoon and night.

   1.5

   2.4.8

   3.5

   4.4.5

   No.