Algorithm Design and Analysis

Assignment 3

Deadline: May 7, 2024

1. (25 points) Given an undirected graph $G = (V, E)$, a *perfect matching* is a set of edges that touches each vertex exactly once. Design an $O(n)$ time algorithm that, given a *tree* as input, decides if it has a perfect matching, where n is the number of vertices. Prove the correctness of your algorithm.

   Solution:

   Starting from the root of the tree, perform the following operations: If its left/right nodes are leaf nodes, mark the root(now) as visited.

   If its left/right node is not a leaf node, recursively call this function on its left/right node, and then if its left/right node is not marked as visited, mark the root(now) as visited.

   Throughout the process, when a node marked as visited is repeatedly marked, all recursions terminate and return false.

   After all recursion ends, if the root node is marked, return true; else return false.

   Proof of correctness:

   Leaf nodes have only one edge connected, and to select these leaf nodes, their adjacent edges must be selected. When selecting edges, we cannot let the selected points be selected again, so when marking the same point repeatedly, the answer is false. After all leaf nodes are selected, if there are no conflicts, these leaf nodes and their adjacent points are "deleted", to repeat the recursive process until it can be determined whether the root node is marked.

2. (25 points) A *matroid* is a pair $(U, \mathcal{I})$ where U is a set of finitely many elements and $\mathcal{I}$ is a collection of subsets of U, called *the independent sets*, that satisfies the following conditions:

   1. $\emptyset \in \mathcal{I}$.
   2. If $A \in \mathcal{I}$ and $B \subseteq A$, then $B \in \mathcal{I}$.
   3. If $A, B \in \mathcal{I}$ and $|A| < |B|$, then there exists an element $x \in B \setminus A$ such that $A \cup \{x\} \in \mathcal{I}$.

   For example, the pair $(U, \mathcal{I})$ with $U = \{1, 2, 3\}$ and $\mathcal{I} = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}\}$ is a matroid, and it can be verified that the above three conditions are satisfied. The seven subsets of U in the collection $\mathcal{I}$ are called the independent sets.

   (a) Given a matroid $(U, \mathcal{I})$, an independent set $S \in \mathcal{I}$ is *maximal* if there does not exist $x \in U \setminus S$ such that $S \cup \{x\} \in \mathcal{I}$. Prove that every maximal independent set has the same size.

   (b) Given an undirected connected graph $G = (V, E)$, let $\mathcal{I}$ be the collection of the edge subsets $S \subseteq E$ such that the edges in each $S \in \mathcal{I}$ do not form any cycle. Prove that $(E, \mathcal{I})$ is a matroid and the maximal independent sets are the spanning trees.

   (c) Consider a matroid $(U, \mathcal{I})$ and a weight function $w : U \to \mathbb{Z}$ that assigns a weight to each element of U. Prove that the following algorithm correctly finds a maximal independent set S with minimum total weight $\sum_{i \in S} w(i)$.

      1. Sort the elements of U by the weight-ascending order, and let $(1, \ldots, n)$ be the sorted list.
      2. Initialize $S \leftarrow \emptyset$.
      3. **for** each $i = 1, \ldots, n$:
      4. if $S \cup i \in \mathcal{I}$, update $S \leftarrow S \cup \{i\}$; otherwise, do nothing.
      5. **return** S.

      Remark: If $(U, \mathcal{I})$ is the matroid defined in Part (b), then this algorithm is exactly Kruskal's algorithm.

   Solutions:

   (a) If the sizes of these sets are different, let $|A| < |B|$, and $x \in B$ while $x \notin A$, then according to 3, $C = A \cup \{x\} \in \mathcal{I}$, $|C| > |A|$, which is inconsistent with assumptions.

   (b) Verify that $(E, \mathcal{I})$ is a matroid:

      1. The empty set is a subset of the set of edges and contains no cycles.

      2. If a set of edges does not contain a cycle, then any subset of it also does not contain a cycle.

3. If a set of edges A has fewer edges than another set B, then there must be at least one edge in B not in A, adding which to A does not create a cycle. Therefore, $(E, \mathcal{I})$ is a matroid.

Prove that maximal independent sets are spanning trees. A spanning tree is a maximal independent set because it contains $|V| - 1$ edges, and adding any more edge will create a cycle. On the other hand, a maximal independent set cannot have more than $|V| - 1$ edges, or it would create a cycle. Hence, every spanning tree is a maximal independent set.

(c) Firstly, the algorithm sorts the elements in ascending order of weights and then iterates through each element, attempting to add it to set S only if it maintains independence. Since the algorithm processes elements in ascending order of weights, the element with the minimum weight, which maintains independence when added, will be chosen first. Therefore, the resulting set S is a maximal independent set with the minimum total weight.

3. (25 points) Given an undirected graph $G = (V, E)$, a *vertex cover* is a subset of vertices $S \subseteq V$ such that each edge has at least one endpoint included in S. The *minimum vertex cover problem* takes an undirected graph $G = (V, E)$ as input and outputs a vertex cover S with minimum size $|S|$.

   Prove that the following algorithm is a $(\ln |E|)$-approximation algorithm.

   1. Initialize $S \leftarrow \emptyset$.
   2. **while** S is not a vertex cover
   3. find a vertex x with the largest degree
   4. update $S \leftarrow S \cup \{x\}$
   5. update G by removing vertex x and all the edges incident to x
   6. **endwhile**
   7. **return** S.

   Solution:

   Let's consider each iteration of the algorithm, where we select the vertex with the highest degree each time. Suppose x is the vertex with the highest degree, then it covers at least all neighbors of x. Therefore, every vertex in S covers at least one edge. Thus, S is a vertex cover.

   On the other hand, let OPT be the size of the optimal vertex cover. Since the algorithm selects the vertex with the highest degree to add to S each time, the size of S will not exceed twice the size of OPT, i.e., $|S| \leq 2 \cdot |OPT|$.

   Additionally, for any graph, the maximum degree of a vertex does not exceed the number of edges in the graph. Thus, $|S| \leq 2 \cdot |OPT| \leq 2 \cdot |E|$.

   Therefore, the approximation ratio is $\frac{|S|}{|OPT|} \leq \frac{2 \cdot |E|}{|OPT|}$.

4. (25 points) Consider the minimum vertex cover problem defined in the previous question and suppose G is a connected undirected graph. Consider the following algorithm. Find a DFS-tree T of G and output the set of all internal nodes of T (an internal node of a tree is a node that is not a leaf).

   (a) Prove that this is a 2-approximation algorithm.

   (b) Does the algorithm still work if we use BFS-tree instead?

   Solutions:

   (a) Let S be the set of nodes output by the algorithm, and OPT be the optimal solution to the minimum vertex cover problem. Since the algorithm outputs all internal nodes of the depth-first search tree T, the size of S is $|S| = |V(T)| - 1$, where $|V(T)|$ denotes the number of nodes in T.

   In T, each edge is incident to an internal node. Since G is connected, each node in T has a degree of at least 2. Therefore, $|E(T)| \geq \frac{1}{2} \cdot |V(T)|$, where $|E(T)|$ denotes the number of edges in T.

   Since T is a tree, we have $|E(T)| = |V(T)| - 1$.

   Combining the above inequalities, we get $|S| = |V(T)| - 1 \leq 2 \cdot |E(T)| \leq 2 \cdot |E(G)|$, where $|E(G)|$ denotes the number of edges in G.

   Therefore, $|S| \leq 2 \cdot |E(G)|$, indicating that the output of the algorithm is at most twice the size of the optimal solution.

   Thus, the algorithm is a 2-approximation algorithm.

   (b) If we use a breadth-first search tree instead of a depth-first search tree, the algorithm still works In a breadth-first search tree, each node has at least one child node because the depth of each node is at most one more than the depth of its parent node.

   Since each edge is incident to an internal node, the set of nodes output by the algorithm is still an approximate solution to the vertex cover problem.

   Therefore, the algorithm remains a 2-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.

   I spent about four hours completing this assignment.

   I think the difficulty score is 5. Question 1, 2 is relatively easy, while the following questions are more difficult.

   I used ChatGPT when answering. Although these problems are related to greedy algorithms and graph algorithms, the introduction of many new concepts has made understanding the problems somewhat difficult. If possible, I would like to reduce the difficulty of the homework or use some easy to understand algorithm questions (such as those on Leetcode) as questions.