Homework 5

ECE 358 Foundations of Computing
Fall Semester, 2023

Due: Saturday, December 9, 5:00pm

- All page numbers are from the 2022 edition of Cormen, Leiserson, Rivest and Stein.
- For each algorithm you asked to design you should give a detailed *description* of the idea, proof of algorithm correctness, termination, analysis of time and space complexity. If not, your answer will be incomplete and you will miss credit. You are allowed to refer to pages in the textbook.
- Do not write C code! When asked to describe an algorithm give analytical pseudocode.
- **Read the handout with the instructions on how to submit your Homework online. Failure to adhere to those instructions may disqualify you and you may receive a mark of zero on your HW.**
- Write *clearly*, if we cannot understand what you write you may not get credit for the question. Be as formal as possible in your answers. Don't forget to include your name(s) and student number(s) on the front page!
- No Junk Clause: For any question, if you don't know the answer, you may write "I DON'T KNOW" in order to receive 20% of the marks.

---

1. [**Approximation Algorithms, 30 Points**]

   Given a connected, weighted, undirected graph $G = (V, E)$ and a subset $R \subseteq V$ (of "required" vertices), a *minimum Steiner tree* of G is a tree of minimum weight that contains all the vertices in R. (It may or may not contain the remaining vertices).

   Finding a minimum Steiner tree is NP-hard in general. Consider a class of approximation algorithms that uses the following heuristic strategy:

   1) Compute the complete distance graph $G_1 = (R, R \times R)$ between vertices in R; each edge (u, v) in G_1 is weighted with the length (*i.e.,* number of edges) of the shortest path from u to v in G.
   2) Compute a minimum spanning tree G_2 of G_1.
   3) Map the graph G_2 back into G by substituting for each edge of G_2 a corresponding shortest path in G. Call the resulting graph G_3.
   4) Compute a minimum spanning tree G_4 of G_3.
   5) Iteratively delete all leaves in G_4 that are not vertices in R.

Of all the minimum Steiner trees for G and R, let T_{opt} be the one with the minimum number of leaves. Let T_{approx} be the Steiner tree obtained using the strategy outlined above. If $w(T)$ denotes the total cost of a tree T (*i.e.,* the sum of all its edge weights), prove that $w(T_{approx}) \leq 2(1 - \frac{1}{\ell})w(T_{opt})$, where ℓ is the number of leaves in T_{opt}.

Hint:

a. Consider the nodes of T_{opt} in the order of the clockwise traversal A, that starts and ends at the same vertex. We split the traversal in parts, where each part will be a simple path from one leaf to another leaf in order of traversal of A. Let's denote these parts as $C_1, C_2, ..., C_l$ (ℓ is the number of parts in A).
How does $\sum_{i=1}^{\ell} w(C_i)$ relate to the total weight of T_{opt}?

b. Consider the largest part (in terms of edge weight) in the set of C_i denoted as C_{max}. How does its weight relate to the average part weight $\frac{\sum_{i=1}^{\ell} w(C_i)}{\ell}$?

c. Consider $P = A \backslash \{C_{max}\}$. Derive a lower bound for the quantity in the problem ($P \leq 2(1 - \frac{1}{\ell})w(T_{opt})$) using answers to the questions a-b.

d. P contains each edge of T_{opt} **at least once**, and **maybe twice**. How do you bound its weight from below?

e. Can you find a bound that is even lower that uses one of the graphs mentioned in the algorithm?

f. Combine answers to the points a-e to prove the statement of the problem.

2. **[NP-completeness, 5+15 points]**

Consider the following problems A and B:

A = CLIQUE: Given a graph G determine if it contains a clique (complete subgraph) of size k.

B = LARGEST-COMMON-SUBGRAPH: Given two graphs G_1 and G_2 and an integer k, determine whether there is a graph G with at least k edges which is a subgraph of both G_1 and G_2.

Show that B is NP-complete by showing that:

(a) $B \in$ NP. That is, show that a given solution can be verified in polynomial time. Remember to give the form of the certificate.

(b) B is NP-hard, by showing that $A \leq_P B$.

3. **[NP-completeness, 5+15 points]**

Consider the following problems A and B:

A = SUBSET SUM: Given a set S of n integers and a target value k, determine whether any subset $C \subseteq S$ has a sum equal to k.

B = KNAPSACK: Given a set S of items numbered 1 to n each with weight w_{i} and value v_{i}, determine whether a collection of items $C \subseteq S$ can be made such that the sum total weight of all items in C is at most W and the sum total value of all items in C is at least V.

Show that B is NP-complete by showing that:

(a) $B \in$ NP. That is, show that a given solution can be verified in polynomial time. Remember to give the form of the certificate.

(b) B is NP-hard, by showing that $A \leq_P B$.