Algorithms, 2023 Fall, Homework 4
(Due: Oct 22)

October 9, 2023

Required Readings:

1. KT book, section 6.5, 6.10

You can discuss homeworks with your classmates. But you will have to write down the solution on your own. You need to acknowledge other students who help you in the homework. If you read some other source that is helpful, you should list all of them.

Problems marked with (@) can be skipped by those who have extensive experience in OI (you need to provide a certificate to TA).

A footnote: Dynamic programming is a fairly standard algorithmic technique. But if you never see it before, it takes some effort to master it. Typically, the tricky part is to correctly set up the subproblems and this sometimes requires good intuition and some experience. If you never learn dynamic programming before, I highly suggest you to do more exercises to train yourself, especially you have a hard time finishing the above problems. (Both KT and CLRS books are very good sources of such exercises.)

Problem 1 : (@) KT book pp.318 problem 7

Problem 2 : (@) KT book pp.323 problem 11

Problem 3 : (@) Implement the weighted interval scheduling problem in $O(n \log n)$ time. You algorithm should return both the optimal value and the optimal solution (the subset of jobs).

Problem 4 : (@) (Traveling salesman path for the set of vertices of a convex polygon) We are given a convex polygon with n vertices $\{1, 2, \ldots, n\}$ in the 2-d plane. The coordinates of these vertices are given as input in clockwise order. Design an polynomial time algorithm to find a shortest traveling salesman path starting at vertex 1. (A traveling salesman path is a path that traverses each vertex exactly once.)

Problem 5 : KT book pp.329 problem 19

Problem 6 : There is a pile of n sticks. S is a set of positive integers. Two players A and B take turns removing s sticks for some $s \in S$. A starts first. The players who removes the last stick wins. Design a pseudo-polynomial time algorithm (i.e., polynomial in n) that decides which player has a winning strategy (i.e., no matter how the opponent plays, the player can take certain moves to win the game.) (Research problem: can you do better than pseudo-polynomial?)

Problem 7 : (1) Show that the treewidth of a tree is 1.

(2) Show that the treewidth of (a tree plus one edge) is at most 2.

(you need to build a tree decomposition for a graph)

Problem 8 : In this problem, we will prove the separator lemma I mentioned in the class. Suppose $G(V, E)$ is an undirected graph and T is a tree decomposition of G. Suppose (b, b') is an edge in T, where b and b' are two bags. Show $b \cap b'$ is a separator of G: Suppose we remove edge (b, b') from T, T becomes disconnected. Let T_1, T_2 are the two resulting components. Now consider two different vertices $v, u \in V$ ($u, v \notin b \cap b'$). Suppose $v \in b_1 \in T_1$ and $u \in b_2 \in T_2$ where b_1 is a bag in T_1 and b_2 is a bag in T_2.

Prove that if we remove all vertices in $b \cap b'$ from G, u is disconnected from v.

Problem 9 : In this problem, we introduce a very useful and important concept in combinatorics, *matroid*. Matroids have very close connections with greedy algorithms and submodular functions (another important concept we will explore in later homeworks). The theory of matroids is a deep branch of combinatorics.

A finite matroid M is a pair $(E, \mathcal{I})$, where E is a finite set (called the "ground set") and $\mathcal{I}$ is a collection of subsets of E. Each element in $\mathcal{I}$ (which is a subset of E) is called an *independent set* (this notion has nothing to do with the notion of independent set in graph theory). Moreover, $(E, \mathcal{I})$ has to satisfy the following properties:

1. $\emptyset \in I$.
2. Every subset of an independent set is independent, i.e., if $A \subset B, B \in \mathcal{I}$, then $A \in \mathcal{I}$.
3. If A and B are two independent sets of $\mathcal{I}$ and A has more elements than B, then there exists an element $e \in A - B$ such that $B \cup \{e\}$ is an independent set.

Try to answer the following questions.

1. Suppose E is a set of real vector of the same length and
$$\mathcal{I} = \{S \subseteq E \mid \text{All vectors in } S \text{ are linearly independent}\}.$$
Prove that $(E, \mathcal{I})$ is a matroid.

2. Suppose E is the set of edges of a given graph G and
$$\mathcal{I} = \{S \subseteq E \mid \text{All edges in } S \text{ constitute a forest of } G\}.$$
(A forest is just a collection of trees. Note that a singleton node is also a tree) Prove that $(E, \mathcal{I})$ is a matroid.

3. Suppose each element $e \in E$ is associated with a weight $c_e \geq 0$. Design a polynomial time algorithm to find a max-weight independent set. In other words, the algorithm should output an independent set $A \in \mathcal{I}$ such that $\sum_{e \in A} c_e$ is maximized. You need to prove the correctness of your algorithm. (Hint: Kruskal's algorithm.)

(Sidenote: There are several important problems which also fits to the framework of matroid. We will cover some of them in the later part of this course. Matroid also has a deep connection with linear programming and other set systems (like polymatroid and greedoid). You will learn some of them later.)

Bonus problems: Bonus problems are problems that may be (may be not) more challenging than other problems. You do not lose points if you do not answer bonus problems. But you will get extra points if you get correct answers for such problems. **To make your life easier, we give you two weeks for the bonus problems**).

You need to first review the matrix tree theorem for the following problem.

Problem 1 : (bonus, 1pt) In this problem, you will see how to count the number of spanning trees of a specific weight in pseudo-polynomial time, using the matrix technique. In this problem, each edge e has a weight which is a positive integer $w_e \in [M]$. Now, I provide you a very high level idea and you are asked to fill in all the details. Essentially, we make the following changes to the matrix A: The value of an entry of A now is x^{w_e} (with appropriate sign), where x is a variable (instead of just a number). Then $\det(A)$ is in fact a polynomial in x (think about it! and figure out how to compute it in poly-time) and the coefficient of x^n would count the number of spanning trees with weight exactly n. (Such a polynomial is called a *generating function*). Formulate the theorem you want to prove (either the directed version or undirected version is fine) and prove it. (We say an algorithm runs in pseudo-polynomial time if the running time is polynomial in the input size and the largest integer, M, in the input instance. On the other hand, a truly polynomial time algorithm should run in time polynomial in the input size and $\log M$ (because an integer M can be encoded in $\log M$ bits).)