TA: Bekhzod Soliev (as always)
Email: bsoliev@eng.ucsd.edu
Office hours: Thursday 3-5pm B270A
Discussion sections: odd weeks on Wednesdays and Thursdays

Resume.
Facebook. Let me know if you are interested in ACM ICPC.
Problem 1 (DPV 5.6)

**Statement.** Let $G = (V, E)$ be an undirected graph. Prove that if all its edge weights are distinct, then it has a unique minimum spanning tree.
Observations.

- The uniqueness of some object is usually proved by assuming the contrary.
- For this question, you may assume that there are two distinct minimum spanning trees in $G$: $T_1$ and $T_2$.
- There is at least one edge $e$ in $T_1$, which is not in $T_2$.
- Think about which edge $e$ to choose.
- Remember that all edge weights are distinct, thus, there is no other edge in $T_1$ or $T_2$ with the same weight as the weight of edge $e$.
- Applying cut property or cycle property may be useful (see Lecture 9).
Problem 2

Statement. We are given two arrays, $A$ and $B$, containing $n$ positive integers each. You can permute each of the two arrays in any fashion such that after the permutation, we have two permuted arrays $A'$ and $B'$ corresponding to arrays $A$ and $B$ respectively. Give an efficient algorithm to compute the maximum possible value of $\prod_{i=1}^{n} (A'[i] + B'[i])$. 
Problem 2

Observations.

▶ Try solving this problem with a greedy algorithm. Think, which ordering of numbers would lead you to the largest possible product.

▶ To prove your solution you may use exchange method.

▶ Let $(A'[1], B'[1]), (A'[2], B'[2]), \ldots, (A'[n], B'[n])$ be the answer produced by your algorithm and let $(A_{opt}[1], B_{opt}[1]), (A_{opt}[2], B_{opt}[2]), \ldots, (A_{opt}[n], B_{opt}[n])$ be the optimal answer.

▶ Your goal is to get your answer from the optimal answer by exchanging elements or numbers in pairs in the optimal answer without changing its optimality.
Problem 2

Observations.

▶ You may permute the pairs in the optimal answer to get $A'[1], A'[2], ..., A'[n]$ as their first coordinates without making the optimal answer worse (explain/prove why). The result will be $(A'[1], B_{opt}^*[1]), (A'[2], B_{opt}^*[2]), ..., (A'[n], B_{opt}^*[n])$

▶ Compare the obtained sequence with the sequence produced by your algorithm from left to right. Find the first pair of pairs $(A'[i], B'[i]), (A'[i], B_{opt}^*[i])$, such that $B'[i] \neq B_{opt}^*[i]$.

▶ Find pair in the optimal answer $(A'[k], B_{opt}^*[k])$, such that $B'[i] = B_{opt}^*[k]$

▶ If you can show that $k \geq i$ and that exchanging $B_{opt}^*[k]$ with $B_{opt}^*[i]$ won't make the optimal answer worse, then you may use this to prove the optimality of your answer (produced by the greedy algorithm) (explain/prove why)

▶ Complete the proof using the information obtained from the previous steps (just proving the previous steps isn’t enough!:)
Problem 3

**Statement.** You are given $2n$ numbers. Your task is to partition them into two sets of $n$ numbers each, so as to maximize the quantity: (sum of all numbers in the first set) minus (sum of all numbers in the second set). Devise an efficient algorithm to accomplish your task.
Problem 3

Observations.

- Try solving this problem with a greedy algorithm.
- To prove your solution you may use exchange method.
- Let $A$ and $B$ be the sets of $n$ numbers produced by your algorithm and let $s(A)$ and $s(B)$ be their respective sums of the elements in them.
- Let $A_{opt}$ and $B_{opt}$ be the optimal answer and let $s(A_{opt})$ and $s(B_{opt})$ be their respective sums of the elements in them.
- **Sets $A$ and $B$ may have duplicate elements.**
- Your goal is to exchange elements between $A_{opt}$ and $B_{opt}$ **without making the optimal answer worse**, thus, without decreasing $s(A_{opt}) - s(B_{opt})$. 
Problem 3

Observations.

▸ (Let’s denote our array as \((a_1, a_2, \ldots, a_{2n-1}, a_{2n})\))

▸ Prove that if your solution is different than the optimal one, then there exists a pair of numbers \((a_i, a_j)\), such that \(a_i \in A\), \(a_i \in B_{opt}\) and \(a_j \in B\), \(a_j \in A_{opt}\).

▸ If you can show that exchanging \((a_i, a_j)\) in the optimal answer won’t make the optimal answer worse, than you may use this to prove the optimality of your answer (produced by the greedy algorithm) (explain/prove why)

▸ Complete the proof using the information obtained from the previous steps (just proving the previous steps isn’t enough!:)
Problem 4

**Statement.** Let’s consider $n$ cities $c_1, c_2, c_3, \ldots, c_n$. One can travel from a city to another city through a one-way bridge. Furthermore, assume that there are no cycles generated by the bridges (even in between two cities). Thus, we have a directed acyclic graph, where vertices represent cities and directed edges represent one-way bridges. Give an efficient algorithm to find the longest sequence of cities that one can travel to. Note that you are not given a city to start with; the algorithm should rather return a longest sequence of cities that is possible to travel to.
Problem 4

Observations.

- Represent cities as nodes and one-way bridges as directed edges. You will get a DAG.
- Define a function $f[v]$ as the length of the longest path starting at node $v$.
- Compute function $f[v]$ for all nodes in an efficient way (think about the order of the nodes and the recursion).
- Can you find the the length of the longest path in the DAG using function $f[v]$?
- Can you find the longest path itself (a sequence of nodes) using values for function $f[v]$?
- The overall solution should have a linear time complexity, i.e. $O(|V| + |E|)$. 
Problem 5

**Statement.** *(Extra credit, worth 0.5 problems on this assignment = 0.5 total points in the class.)* Jersey Ike is aiming to enter the Guinness Book of World Records with the fastest and most massive delivery of sandwiches to customers in a single location. The company’s strategy is to set up a VERY long service counter, with $K$ Jersey Ike employees standing behind it. Before the timer starts, $N$ customers take their places side by side at the counter, and place their sandwich orders. For each sandwich, it is known how much time is required to make the sandwich (all of the Jersey Ike employees are identically fast on all sandwich types). Jersey Ike must assign its employees to contiguous sections of the counter, so as to minimize the time required to make all of the customers’ sandwiches. *(In other words, a single employee cannot make the sandwiches of, say, customers 13, 22 and 57. But, a single employee could be assigned to make the sandwiches of customers 13, 14 and 15.)* Also, the sandwich-making must respect a core value of the Jersey Ike company, namely, that any given sandwich must be made by exactly one employee.
Problem 5

**Statement.** Devise an efficient algorithm that determines how Jersey Ike should optimally assign sections of the counter (i.e., contiguous intervals of customers) to its $K$ employees.
Problem 5

Observations.

- Be patient and read the statement carefully.
- You are trying to find the optimal assignment by minimizing time.
- Can you implement an algorithm which will determine whether there is a valid assignment for some time $t$?
- Can you now find the optimal value for $t$ in an efficient way? (prove why this works)
- Week 5’s discussion section slides should be useful.