# Fall 2019 B503 Homework 1 Solutions

Due date: TBD    Lecturer: Qin Zhang

**Problem 1 (10 points).** Please prove the following claim: When applying the algorithm we learned form the class to solve stable matching problem, if men propose, every woman gets her worst valid partner (according to her own preference).

**Solution:** Let's proof it by contradiction. Let $(m, w)$ be a man-woman pair in the solution returned by the algorithm. Assume there is another solution where $w$ is matched with man $m'$ and $rank[w][m'] < rank[w][m]$. And in this solution $m$ is matched with another woman $w'$. There are 2 possible cases:

- $rank[m][w'] > rank[m][w]$. As we proved in class, this is impossible, since each man gets its best possible partner.
- $rank[m][w'] < rank[m][w]$. It means that $m$ prefers $w$ to $w'$. Since $w$ prefers $m$ to $m'$, solution with matching $(m', w)$ is unstable.

Thus, both cases are impossible to happen, we proof our claim.

**Problem 2 (10 points).** A polygon is convex if all of its internal angles are less than 180 degree (and none of the edges cross each other). Figure 1 shows an example.

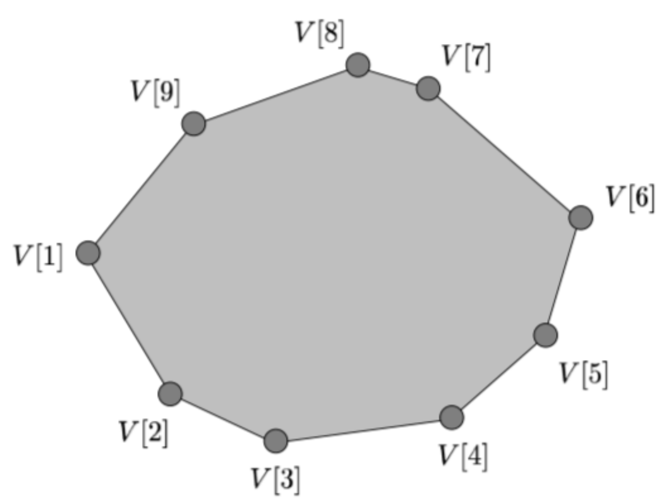


**Figure 1**: An example of a convex polygon represented by the array $V[1..9]$. $V[1]$ is the vertex with the minimum $x$-coordinate, and $V[1..9]$ are ordered counterclockwise.

We represent a convex polygon as an array $V[1,\ldots,n]$ where each element of the array represents a vertex of the polygon in the form of a coordinate pair $(x, y)$. We are told vertex $V[1]$ is the vertex with the minimum $x$ coordinate and that the vertices $V[1...n]$ are ordered counterclockwise, as in the figure. You may also assume that the $x$ coordinates of the vertices are all distinct, as are the $y$ coordinates of the vertices.

Give an algorithm to find the vertex with the maximum $y$ coordinate in $O(\log n)$ time.

**Solution:** An array $A$ is unimodal if it consists of an increasing sequence followed by a decreasing sequence. To find a the maximum in such an array, we could use the following algorithm.

Look into $A[n/2-1], A[n/2], A[n/2+1]$:

1. if $A[n/2]$ is maximum ($A[n/2-1] < A[n/2] > A[n/2+1]$), return;
2. if $A[n/2-1] > A[n/2] > A[n/2+1]$, look for maximum in $A[n/2+1,\ldots,n]$;
3. if $A[n/2-1] < A[n/2] < A[n/2+1]$, look for maximum in $V[1,\ldots,n/2-1]$.

Repeat the process until find the maximum. $T(n) = T(n/2) + O(1)$ leads to $O(\log n)$ algorithm.

To find the maximum $y$ coordinate, we need two steps. First is to find the maximum $x$ coordinates. As $x$ coordinates forms a unimodal array, we could find the maximum in $O(\log n)$ time. Let $max$ represents the index of the elements with maximum $x$ coordinate. The second steps is to find the maximum $y$ which is among $V[max], V[max+1],\ldots,V[n]$, which is another unimodal array.

So the original problem can be solved in $O(\log n)$ time.

**Problem 3 (10 points).** You are given a binary tree $T = (V, E)$, along with a designated root node $r \in V$. You wish to preprocess the tree so that queries of the form "is $u$ an ancestor of $v$?" can be answered in $O(1)$ time. The preprocessing itself should take $O(n)$ (linear in terms of the number of vertices $n$) time. Please describe and analyze your preprocessing and query algorithm.

**Solution:** This problem can be solved in one round of DFS in $O(n)$ time.

During the DFS, record two time stamps for each node $v$: $v_{first}$ and $v_{last}$, which represent the time that a node is first and last visited.

If $u$ is indeed a child of $v$, we will have the following property: $v_{first} < u_{first} < u_{last} < v_{last}$, which only takes constant time to check.

**Problem 4 (10 points).** Judge Jill are dealing with tons of complains, each complaint containing exactly two names: that of the person who filled it and that of the person he or she is complaining about. Jill would like an automated approach to deal with such large amount of complaints.

She decides to try to label each person involved in the complaint as either good or evil. She only needs the labeling scheme to be consistent, not necessarily correct. A labeling is consistent if every complaint labels one person as good and the other person as evil, and no person gets labeled both as good and evil in different complaints.

(a) Please model this problem as a graph model and propose an efficient algorithm to consistently label all the names as good or evil, or to decide that no such labeling exists. Your algorithm should run in $O(m+n)$ time (linear in terms of the number of vertices $n$ and edges $m$).

(b) Later, Judge Jill wants more than a consistent labeling. She will interview some people to figure out his or her true label, as she can always determine whether a person is good or evil by interviewing him or her. Assuming that there exits a correct labeling such that one person in every complaint is either good or evil, what is the minimum number of people she needs to interview to correctly classify all the people named in the complaints?

**Solution:**

(a) We could model this problem using an undirected graph in which each edge represents a complaint and each vertex represents a person.

This problem corresponds to determine if a graph is bipartite, which can be solve using DFS in $O(m+n)$ time. Begin with an arbitrary node and mark it as good or evil, then every time we visit a new node, we mark it with a different label than its parent. Each time we see a visited node, we check for consistency. If we find any inconsistency, the graph cannot be bipartite, then no consistent classification exits in the original problem

(b) She only needs to interview one person in each connected component of the graph. If the person she interviews has the right label, then everyone in the connected component will also have the right label. If the person she interviews has the wrong label, then everyone in the connected component will also have the wrong label.

**Problem 5 (10 points).** Given an undirected graph, please give an $O(n)$ time (linear in terms of the number of vertices $n$) algorithm to detect whether the given graph contains a cycle.

**Solution:** A connected undirected graph $G$ that has no cycles is a tree with $n-1$ edges. First count the number of edges, if it has more than $n-1$ edges, this graph contains a cycle. If the graph has at most $n-1$ edges, we could conduct a DFS. If an unexplored edge leads to a visited node, then the graph contains a cycle. Otherwise this graph has no cycle. So the algorithm takes $O(n)$ time.

**Problem 6 (10 points).** Suppose that an $n$-node undirected graph $G=(V,E)$ contains two nodes $s$ and $t$ such that the distance between $s$ and $t$ is strictly greater than $n/2$.

(a) Show that there must exist some node $v$, not equal to either $s$ or $t$, such that deleting $v$ from $G$ destroys all $s$-$t$ paths. (In other words, the graph obtained from $G$ by deleting $v$ contains no path from $s$ to $t$.)

(b) Give an algorithm to find such a node $v$ with $O(m + n)$ running time (linear in terms of the number of vertices $n$ and edges $m$).

**Solution:** If deleting a node results two unconnected component a graph, the graph must have at least one cut of size one. We could prove by contradiction: assuming the graph has cut size of at least two. In this case, when we perform BFS, we could reach at least two new nodes in each iteration.

Start from s and perform BFS, if we could reach at least two new node in each iteration, we could finish BFS in $n/2$ iterations. Then every node in $G$ has a path to $s$ at most $n/2$ which violates the fact that there exits a $s - t$ path with length strictly more than $n/2$.

Applying BFS until we reach the round in which only one new node is exposed. The node is the one we are looking for. Such BFS runs in $O(m + n)$ time.

**Problem 7 (10 points).** The police department in the city of the Wonder Land has made every street in Wonder Land one-way. Despite widespread complaints from confused motorists, the mayor claims that it is possible to legally drive from any intersection to any other intersection.

(a) The city needs to either verify or refute the mayor's claim. Formalize this problem in terms of graphs, and then describe and analyze an algorithm to solve it.

(b) After running your algorithm from part (a), the mayor reluctantly admits that she was misinformed. Call an intersection $x$ good if, for any intersection $y$ that one can legally reach from $x$, it is possible to legally drive from $y$ back to $x$. Now the mayor claims that over 95% of the intersections in Wonder Land are good. Describe and analyze an efficient algorithm to verify or refute her claim.

For full credit, both algorithms should run in $O(m + n)$ time (linear in terms of the number of vertices $n$ and edges $m$).

**Solution:**

(a) We could model this problem using an directed graph in which each vertex represents an intersection and each edge represents a one-way connection.

If the mayor's claim is true, then the graph can only have one strongly connected component. Strongly connected component can be solve in two DFS (Kosaraju's algorithm).

(b) First apply Kosaraju's algorithm to find all strongly connected components and give each component an unique component index. For each component, randomly pick a node and perform BFS or DFS. If we encounter a node that belongs to another component, we know that nodes in this component are not good. The last thing we need to do is to count the number of nodes that are good.

For a component $c$, we use $n_c$ to represent the number of nodes in $c$ and $m_c$ to represent the number of edges in $c$. For the component $c$ with good nodes, BFS or DFS takes time $O(n_c + m_c)$. For the component $c$ with bad nodes, BFS or DFS takes time $O(n_c + m_c)$ because it stops whenever we encounter a node from another component. So the total running time is $O(m + n)$.