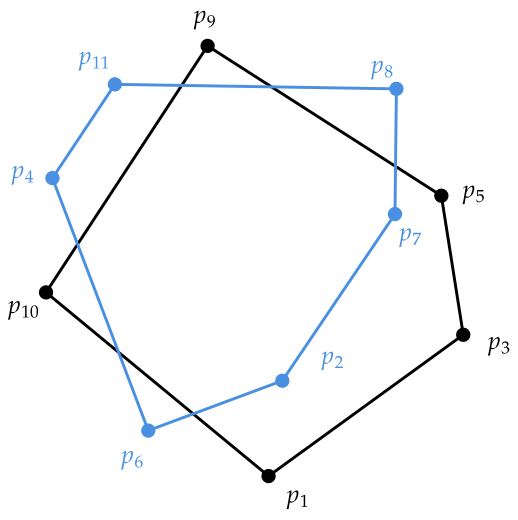
INSTRUCTIONS:

1. You are suggested to use the provided latex template to write your solutions.
2. Submit your solution to Gradescope. Make sure only one of your group members submits. After submitting, make sure to add your group members.
3. You may directly use the algorithms introduced in lectures.

Problem 1 (15 points).

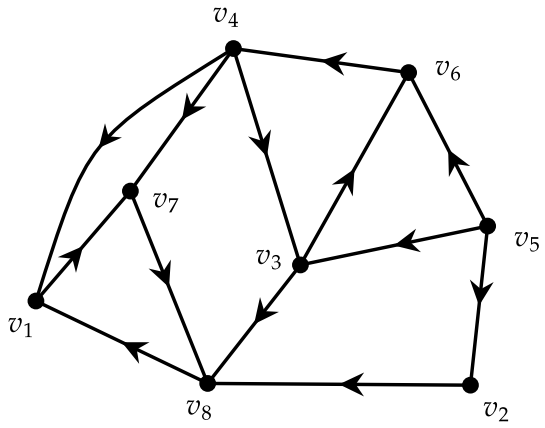
Run the combine step of the divide-and-conquer algorithm for convex hull on the instance given below. You are given $(C_1 = p_5, p_9, p_{10}, p_1, p_3)$ and $C_2 = (p_{11}, p_4, p_6, p_2, p_7, p_8)$.

1. Find the lowest point p^* in $C_1 \cup C_2$.
2. Transform C_1 into C_1' so that points in C_1' is sorted in increasing angle w.r.t. p^*.
3. Partition C_2 into two lists C_{2a} and C_{2b} so that each list is sorted in increasing angle w.r.t. p^*.
4. Give list C_2' by merging C_{2a} and C_{2b} so that each points in C_2' is sorted in increasing angle w.r.t. p^*.
5. Give list C' by merging C_1' and C_2' so that each points in C' is sorted in increasing angle w.r.t. p^*.
6. Run Graham-Scan-Core algorithm to find convex hull of C'. Show stack operations at each step (to deal with each point). For example, you need to write like "For A: push A; pop B", which indicates when you process point A, push A into stack and also pop B out.



Problem 2 (12 points).

Run the DFS-based algorithms on the following graph. Note: whenever you have a choice of vertices to explore, always pick the one that is alphabetically first (i.e., following the order $v_1, v_2, \cdots, v_9$).



1. Run DFS (with timing) on this graph: give the *pre* and *post* numbers all vertices; give the *postlist*.
2. Draw the meta-graph of this graph.
3. Give a linearization of the meta-graph.
4. How many different linearization does the meta-graph have?

Problem 3 (10 points).

Let $G = (V,E)$ be a DAG. Design an $O(|V|+|E|)$ time algorithm to decide if there is only one possible linearization for G. Prove that your algorithm is correct.

Problem 4 (10 points).

1. Given an undirected graph $G = (V,E)$ and an edge $e = (u,v) \in E$, design a $O(|V|+|E|)$ time algorithm to determine whether there exists a cycle in G that contains e.
2. Given a directed graph $G = (V,E)$ and an edge $e = (u,v) \in E$, design a $O(|V|+|E|)$ time algorithm to determine whether there exists a cycle in G that contains e.

Problem 5 (10 points).

A businessman owns n warehouses ($n \geq 3$). In order to protect his products stored in these warehouses, he wants to build a wall surrounding all these warehouses. The shape of the wall will be a convex polygon. You are given n points $P = \{p_1, p_2, \cdots, p_n\}$ on 2D plane, represented as their coordinates. Each point represents a warehouse. Design an algorithm to find the minimal perimeter of such wall. Your algorithm should run in $O(n \log n)$ time.

Problem 6 (10 points).

There are n trains $(X_1, X_2, \cdots, X_n)$ moving in the same direction on parallel tracks. Train X_k moves at constant speed v_k, and at time $t = 0$, is at position s_k. Train X_k will be *awarded*, if there exists a time period (t_1, t_2) of any length, $0 \leq t_1 < t_2$, such that during this entire peroid X_k is in front of all other trains (it is fine if X_k is behind some other train prior to t_1, or X_k is surpassed by some other train after t_2). Given v_k and s_k, $1 \leq k \leq n$, design an $O(n \log n)$ algorithm to list all trains that will be awarded.

Problem 7 (10 points).

Assume that there are n different tasks to be done $T_1, T_2, \ldots T_n$. And there are m different requirements, each requirement is in the form that that one task T_i must be done before another task T_j. Design an algorithm to decide if there exists an order of tasks that satisfies all given requirements, and if it exists, find such an order. Your algorithm should run in $O(n+m)$ time.

Problem 8 (16 points).

You are given a directed graph $G = (V, E)$, where $V = \{v_1, v_2, \cdots, v_n\}$. You can assume that each vertex is reachable from itself.

1. For every vertex v_i, compute the value $f(v_i)$ defined as follows: $f(v_i)$ is the smallest j such that v_i is reachable from v_j. Design an algorithm to calculate values $f(v_1), \cdots, f(v_n)$. Your algorithm should run in $O(|V| + |E|)$ time.

2. For every vertex v_i, compute the value $g(v_i)$ defined as follows: $g(v_i)$ is the smallest j such that v_j is reachable from v_i. Design an algorithm to calculate values $g(v_1), \cdots, g(v_n)$. Your algorithm should run in $O(|V| + |E|)$ time.