1 Instructions

This is due at the start of the class on November 25, 2013. Please write clearly and answer questions precisely. As a thumb rule, the answer should be limited to \( \leq 2 \) written pages, with ample spacing between lines and in margins, per question. Always start a new question on a new page, starting with Question 1, followed by Question 2, ..., Question \( n \). Please cite all the references (including web-sites, names of friends, etc.) which you have used/consulted as the source of information for each of the questions. BTW, when a question asks you to design an algorithm - it requires you to (1) Clearly spell out the steps of your algorithm in pseudocode (2) Prove that your algorithm is correct and (3) Analyze the running time. By default a graph \( G = (V, E) \) is simple, undirected and connected.

2 Problems

1. Suppose you are given \( n \)-points in the plane. We can define a complete graph \( G \) on these points, where the weight of an edge \( e = (u, v) \), is Euclidean distance between \( u \) and \( v \). We need to partition these points into \( k \) non-empty clusters, for some \( n > k > 0 \). The property that this clustering should satisfy is that the minimum distance between any two clusters is maximized. (The distance between two clusters \( A \) and \( B \) is defined to be the minimum among the distances between pair of points, where one point is from cluster \( A \) and the other from cluster \( B \).) Show that the connected components obtained after running Kruskal’s algorithm till it finds all but the last \( k - 1 \) (most expensive) edges of MST of \( G \) produces an optimal clustering. You need to recall how Kruskal’s algorithm work - but do not worry much about its complexity analysis.

2. Consider the following version of the (weighted) planar-separator theorem. Let \( G = (V, E) \) be an embedded undirected triangulated planar graph, where \( n = |V| \). Each vertex \( v \in V \) has a positive weight \( w(v) \geq 0 \) and \( \sum_{v \in V} w(v) = 1 \). There exists a partition of \( V \) into disjoint sets \( A, B, \) and \( S \), such that

- (a) \( w(A), w(B) \leq \frac{2}{3} \), where \( w(A) \) is the sum total of weights of all the vertices in set \( A \)
- (b) \( |S| \leq 4\sqrt{n} \)
- (c) There is no edge in \( E \) that joins a vertex in \( A \) with a vertex in \( B \).
- (d) Such a set \( S \) can be found in linear time.
Show what changes you need to make in the proof of (unweighted) Planar Separator Theorem to prove the above theorem.

3. Prove the following: Let \( G = (V, E) \) be a connected planar graph and \( G^* \) be its dual. For any \( E' \subseteq E \), the subgraph \( (V, E') \) has a cycle if and only if the subgraph \( (V^*, E - E') \) of \( G^* \) is disconnected.

4. Prove the following theorem on Geometric Separators. In 2-dimensions assume that you have \( n \) squares of arbitrary sizes. Squares are axis aligned. Moreover none of the points in the plane is inside more than \( k \)-squares. Prove that there exists either a vertical or a horizontal line which partitions the set of squares in such a way that at least \( \left\lfloor \frac{n+1-k}{4} \right\rfloor \) of squares interiors lie to each side of the line. How fast you can find such a line?

5. Suppose we are given a flow network \( G \), where edges have positive integer capacities, and \( C = \sum_{u,v \in V} c(u,v) \), where \( c(u,v) \) is the capacity of the edge \( e = (u,v) \in E \). Show the following:
   (a) The value of the max flow is an integer.
   (b) There is an assignment of non-negative integer flow values on each edge of \( G \), satisfying all the flow conservation conditions, so that \( G \) achieves max flow.
   (c) Show that the number of iterations required in the Ford-Fulkerson’s algorithm (Residual network, find an augmenting path, augment the flow, repeat) is \( O(C) \).
   (d) Show that in the worst case, Ford-Fulkerson’s algorithm, as stated in the Question 5c runs in exponential time.
   (e) Construct an example, where one can realize the worst case as stated in the Question 5d.

6. Let \( G = (V, E) \) be the flow network. Let \( C = \max_{(u,v) \in E} c(u,v) \) be the maximum capacity. Show the following:
   (a) Minimum cut of \( G \) has a capacity of at most \( C|E| \).
   (b) For a given number \( K > 0 \), show how to find an augmenting path of capacity at least \( K \) in \( O(|E|) \) time, provided that such a path exists.
   (c) Execute the following algorithm:
      i. Initialize the flow \( f = 0 \);
      ii. \( K = 2^{\lceil \log C \rceil} \);
      iii. While \( K \geq 1 \) do
         A. While there exists an augmenting path \( p \) of capacity at least \( K \) then augment flow \( f \) along \( p \).
         B. \( K := K/2 \);
      iv. Return \( f \).
      Show that the above algorithm computes Max-Flow.
   (d) Show that the loop in Step iii.A is executed at most \( O(|E|) \) times for each value of \( K \).
   (e) Show that the algorithm runs in \( O(|E|^2 \log C) \) time.
7. Let \( G = (V, E) \) be a flow network. Recall that \( G \) is a complete graph, where some of the edges may have a capacity of zero. Suppose your task in the max flow problem is to increase the flow of a network as much as possible, but you are only allowed to increase the capacity of only one edge, whose capacity is strictly larger than zero. First show that there are networks where such an edge may not exist, i.e. increasing the capacity of a single edge (> 0 capacity) will not alter the value of the max-flow. Show that there are networks, where such an edge may exist. Try to design an algorithm which can detect whether flow can be increased. (Thanks to Alexis - to make the question relevant/interesting, we should have the condition of non-zero capacity :).)

8. A simple undirected graph \( G = (V, E) \) is called \( k \)-edge connected if removal of any set of \( k \)-edges keeps \( G \) still connected. (e.g. cycles are 1-edge connected.) Show how to compute edge connectivity of \( G \) by invoking at most \( |V| \) network flow computations.

9. Prove that in the LCA algorithm of Bender and Farach-Colton, why does the reduction from the LCA problem to the range-minima query works, i.e., show that in place of finding the LCA of nodes \( u \) and \( v \) in the binary tree, why does it suffice to compute the smallest level number in the level array in an interval defined by the first occurrence of the node \( u \) an \( v \) in the level array.

10. This problem is to show that an arbitrary range minima query (RMQ) problem can be solved within the same complexity as the one with the \( \pm 1 \) RMQ problem. Recall that the \( \pm 1 \) RMQ problem for an array of size \( n \) required \( O(n) \) time to preprocess and then the queries were answered in \( O(1) \) time. The idea is to reduce an arbitrary RMQ problem to the LCA problem. This reduction uses Cartesian Tree. Let \( A \) be an array consisting of \( n \) numbers (need not satisfy the \( \pm 1 \) property). The Cartesian Tree \( C \) for \( A \) is defined as follows: The root of \( C \) is the minimum element of \( A \), and it stores the position of this element in the array. Removing the root element splits the array into left and right subarrays. The left and right children of the root are recursively constructed Cartesian trees of the left and right subarrays, respectively. Prove the following:

   (a) Cartesian tree \( C \) of an array \( A \) of size \( n \) can be computed in \( O(n) \) time (use incremental construction).

   (b) Show that \( RMQ_A(i, j) = LCA_C(i, j) \) (Recall that in \( C \) we store the indices \( i \) and \( j \).)