Midterm 2

Due: December 15, 2021

Reminder: Your name should not appear anywhere on your handin; each individual page of the homework should include your Banner ID only. For your digital submission, each page should include work for only one problem (i.e., make a new page/new pages for each problem).

Collaboration or any kind or use of any third-party resource is strictly not allowed on this assignment. The TA staff will only answer questions regarding clarifications on the statement of the problems in the assignment. Questions should be asked via private posts on Ed. Please monitor Ed, as we will post clarifications of frequently asked questions there.

You have until **11:59 PM** on December 15 to hand your submission in on Gradescope (an extension from the usual due date time). See the course syllabus for the late policy. Usage of any materials outside of course notes, the course textbook, lecture slides and Ed posts is strictly forbidden.

Lastly, you may apply up to two late days to this assignment. Good luck!

Problem 1 - 10 points

Given a flow network G with source s and sink t let V denote it sets of vertices, let E be its set of directed edges such that $(u,v) \in E \implies (v,u) \notin E$ and $(v,u) \in E \implies (u,v) \notin E$. Let $c : E \rightarrow \mathbb{N}^+$ denote the capacity of the edges, and let f be a flow assignment to the network G. That is, for each edge $(u,v) \in E$, f assigns to it a value $f((u,v)) \in \{0, 1, \ldots, c((u,v))\}$.

The *residual network* R of G, f is a directed graph whose set of vertices corresponds to V and such that for direct each edge $(u,v) \in E$, in R there is an edge directed from u to v (i.e., (u,v)) that is assigned the value (or *label*) $c((u,v)) - f((u,v))$, and an edge directed form v to u (i.e., (v,u)) that is assigned the value (or *label*) $f((u,v))$. Finally, remove from R all of the edges who are assigned value O. By construction, all of the edges in R are assigned a positive integer value.

1. Design an algorithm that uses R to find one of the shortest augmenting paths for G given the flow f. That is, an augmenting path such that the number of edges being traversed in G is minimum among all possible augmenting paths. Your algorithm should run in time $O(|V| + |E|)$. Argue its correctness and analyze its running time.
2. Assume that after finding the shortest path you update the network flow and repeat the process on the updates residual network. Further, assume that after at most $O(|E||V|)$ iterations there there is no more shortest augmenting path. Use this components and this approach to obtain an algorithm that finds the maximum flow of a graph. Argue its correctness and analyze its running time.

Problem 2 - 10 points

Let P be a set of n points in the plane. Assume that no two points of P are coincident (i.e. they have both the same x-coordinate and the same y-coordinate). However, it is possible that two or more points have the same x-coordinate or y-coordinate. Also, it is possible that three or more points are collinear (i.e. there exists a straight line that contains them).

1. Modify the convex hull algorithm presented in class to compute for every point p in P that is not a vertex of the convex hull, either a triangle with vertices in P or a segment with endpoints in P that contains p. Note that the only restriction on the points you use to construct your triangles / line segments is that you cannot use point p itself (however, you may find it easier to use a specific subset of points to make your triangles / line segments as opposed to searching through all points in the set– try drawing a picture!). Justify the correctness, running time, and memory utilization of your algorithm (which should run in $O(n \log n)$ time).

Problem 3 - 10 points

Let P be an array with n integer values.

1. Design an algorithm that finds all of the elements of P that are repeated at least 3 times. Your algorithm should have expected running time $O(n)$, and may use some additional memory (i.e. your algorithm can have a non constant space complexity). Provide analysis of correctness, worst case running time, and memory utilization.
2. Assume that the values in P are non-negative integers which are multiples of 5 and such that any element $x \in P$ is such that $0 \leq x \leq 5(n-1)$. Show that you can improve on the algorithm in part 1 without using any additional memory space (relative to part 1).

Problem 4 -10 points

Recall that a *literal* in a Boolean formula is either a Boolean variable (e.g., x_i) or its negated form (e.g., $\neg x_i$) appearing in the formula. Let ϕ be a 3cnf formula. An $\neq$-assignment to the variables of ϕ is one in where each clause contains two literals with unequal truth values. In other words, a given clause cannot be assigned all true or all false in a $\neq$-assignment.

1. Show that the negation of any $\neq$-assignment to ϕ is also a $\neq$-assignment of ϕ.

2. Let $\neq SAT$ denote the problem of deciding whether a Boolean Formula has a $\neq$-assignment. Show that we can obtain a valid and polynomial time reduction $f()$ from $3SAT$ to $\neq SAT$ as follows: Given a 3cnf formula ϕ' we add to each of its clause an additional $\vee u$ term, where u is a new Boolean variable which did not appear in ϕ'. For example

   $$\phi' = (x_1 \vee x_2 \vee x_3) \wedge (x_4 \vee x_1 \vee x_3)$$

   and

   $$f(\phi') = (x_1 \vee x_2 \vee x_3 \vee u) \wedge (x_4 \vee x_1 \vee x_3 \vee u)$$

   .

3. Conclude that $\neq SAT$ is NP-complete.