Cs445 — Homework #6
Midterm's leftover, Dynamic Programming and Network Flow
Due April 18

1. Let A be a sorted array of n keys, and x is a key in the array.

   (a) Explain how to perform the operation $find(x)$ in time $O(\log k)$, where k is the index of the cell containing x (that is $A[k] = x$).

   Obviously, if $k = \Theta(n)$ then a standard binary search would do, but think what to do if k is much smaller than n. A first good step would be to narrow the search to a portion of the array that contains x. Do not use any other data structures.

   (b) Repeat the previous question, but now a $find(x)$ operation should take $O(\log \min\{k, n-k\})$

2. The algorithm discussed in class for finding the edit distance between two strings only find the *cost* of the edit distance. Explain how to change the algorithm that it would actually report the sequence of insert/delete/find operations corresponding to the edit distance.

3. You are given two strings A and B, each with n characters, and you are also given a fixed integer k. We want to compute $ed(A, B)$, (the *edit distance* between A and B) but only if $ed(A, B) \leq k$;

   Show that only $O(nk)$ cells in the array used in the dynamic programming algorithm for computing $ed(A, B)$ need to be checked. Explain why this can be used to design an $O(nk)$-time algorithm in this case.

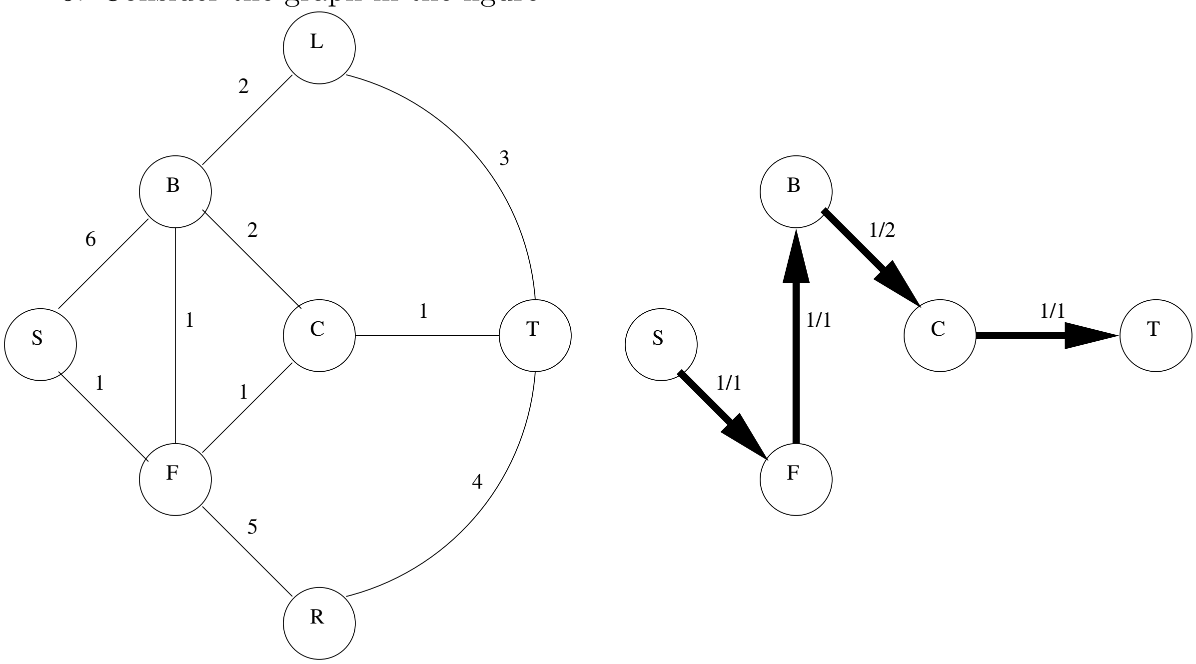
4. **(bonus)** You are given a convex polygon P with n vertices. Assume that its vertices are $\{v_1 \ldots v_n\}$. The segment connecting v_i to v_j, denote $\overline{v_i v_j}$ is called a *chord*. You are also given a array $Cost[1..n, 1..n]$ such that, $Cost[i, j]$ is the cost of the chord $\overline{v_i v_j}$. All costs are positive numbers. Suggest an algorithm that runs in time $O(n^3)$, and finds a subset X of the chords, so that no two chords

in X cross each other, and the sum of their costs is as high as possible. Assume that the chords $\overline{v_1 v_{n/2}}$ must be in X.

Note — two chords of X cannot cross each other, but they can share an end-point.

Hint: The solution for this question is similar to the matrix-chain multiplication algorithm discussed in class. Identify what are subproblems you would like to solved, are how to use the idea that the global optimum consists of optimal solutions to locale subproblems.

5. Consider the graph in the figure



(a) Graph with capacities (same in each direction)

(b) Initial flow for performing Ford–Fulkerson

Show the residual network of this graph, and explain how you compute the residual capacities. Then perform 2 iterations of Ford-Fulkerson algorithm, when the first augmenting path you find is along the vertices S, B, F, R, T.

6. play with the Ford-Fulkerson animation in the class webpage (under web resources).

7. 26.1-6 from CLRS

8. Let $G(V, E)$ be a flow problem, and let f be a flow in G. Show that $|E| \leq |E_f| \leq 2|E|$.

9. Let $G(V, E)$ be a flow problem, and let f be a flow in G. Let π be a path from s to t in G_f. Prove that there is some value $w > 0$ such that if for every edge $(u, v) \in \pi$ we set

$$f'(u, v) = f(u, v) + w \quad ; \quad f'(v, u) = f(v, u) - w$$

and set $f'(u, v) = f(u, v)$ if neither (u, v) nor (v, u) are in π, then f' is also a flow, and $|f'| = |f| + w$.

10. 26.1-9 from CLRS

11. 26.3-3 from CLRS