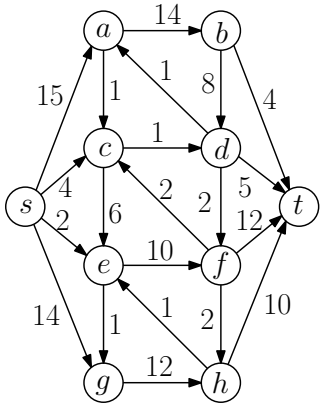
Problem Set 5

CMSC 27230: Honors Theory of Algorithms

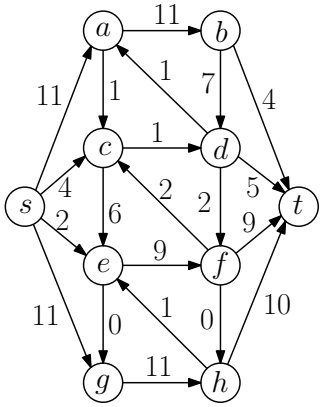
Assigned February 13, Due February 20

Problem 1: Ford-Fulkerson and Maximum Matching on Bipartite Graphs (30 points)

(a) 10 points: Find the residual graph for the given input graph and flow. Is this flow maximal? If so, demonstrate it by finding a cut which has capacity equal to the flow from s to t. If not, run the Ford-Fulkerson algorithm starting from the current flow to find a maximum flow and demonstrate that the flow is maximal by finding a cut which has capacity equal to the flow from s to t.

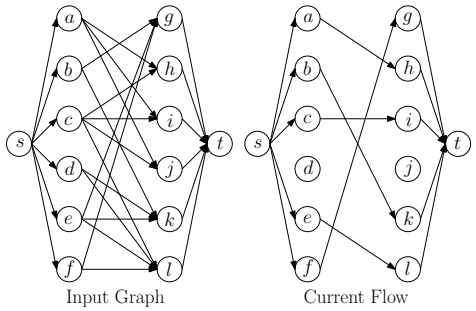


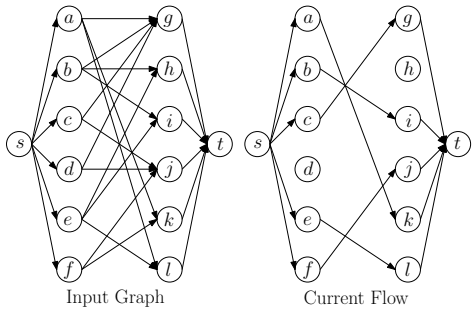
Input Graph



Current Flow

(b) 20 points: Find the residual graph for each of the following two given input graphs and flows (where all edges have capacity 1). If the flow is maximal, demonstrate it by finding a cut which has capacity equal to the flow from s to t. If not, find a maximal flow. How do your answers for these graphs relate to maximum matching, vertex covers, and augmenting paths?





Problem 2: Optimizing a Binary Search Tree (25 points)

Let's say that you are storing n objects $x_1, \ldots, x_n$ in a binary search tree where $x_1 < x_2 < \ldots < x_n$. You know that after you store the objects in the tree, object x_i will be searched for a_i times. You would like to construct the tree so that the total number of queries to the nodes (including the queries used to locate the object in the tree) is minimized. More precisely, you want to minimize $\sum_{i=1}^{n} a_i(d_i + 1)$ where d_i is the depth of x_i in the binary search tree.

Here are some examples:

1. If $n = 3$, $a_1 = 4$, $a_2 = 3$, and $a_3 = 5$ then it is optimal to have x_2 be the root with children x_1 and x_3, which gives $\sum_{i=1}^{n} a_i(d_i + 1) = 4*2+3*1+5*2 = 21$. However, if we instead had $a_1 = 4$, $a_2 = 2$, and $a_3 = 5$ then it would be optimal to have x_3 be the root with left child x_1 and then have x_2 be the right child of x_1. This gives $\sum_{i=1}^{n} a_i(d_i + 1) = 4*2+2*3+5*1 = 19$.

2. If $n = 6$, $a_1 = 2$, $a_2 = 3$, $a_3 = 4$, $a_4 = 5$, $a_5 = 6$, and $a_6 = 7$ then it is optimal to take the root to be x_5 with children x_3 and x_6, have x_2 and x_4 be the children of x_3, have x_1 be the left child of x_2, and have x_1, x_4, and x_6 be leaves. This gives $d_1 = 3$, $d_2 = 2$, $d_3 = 1$, $d_4 = 2$, $d_5 = 0$, and $d_6 = 1$ so we have that

$$\sum_{i=1}^{n} a_i(d_i + 1) = 2 * 4 + 3 * 3 + 4 * 2 + 5 * 3 + 6 * 1 + 7 * 2 = 60$$

Give a polynomial time algorithm to solve this problem, explain why your algorithm is correct, and analyze its runtime.

Problem 3: Ford-Fulkerson With No Backwards Edges (Rephrasing of Problem 11 in Chapter 7 of Kleinberg-Tardos), 20 points)

Your friend wonders, what happens if we don't have backwards edges in the Ford-Fulkerson algorithm? In other words, what happens if, after an iteration of Ford-Fulkerson where we find a path from s to t and route c_{min} flow through it (where c_{min} is the minimum remaining capacity of an edge on this path), we just reduce the capacities of the edges along this route by c_e in the residual graph (and don't increase the capacities of the edges going backwards by c_{min} as we are ignoring them)?

Your friend conjectures that the Ford-Fulkerson algorithm with no backwards edges will still always give a constant factor approximation, i.e. there exists a constant $c > 0$ such that if the maximum flow is F, regardless of how we choose the paths from s to t we will obtain a total flow of at least cF from s to t. Is your friend's conjecture correct? Either prove your friend's conjecture or describe how to construct a counterexample for any $c > 0$. Hint is available.

Challenge question (not for credit, may be very hard): Your friend also conjectures that for any input graph G, there always exists a way for the Ford-Fulkerson algorithm to choose the paths from s to t such that no backwards edges are needed. Is this conjecture true?

Problem 4: Arranging Blind Dates (25 points)

You are in charge of arranging blind dates for n women $x_1, \ldots, x_n$ and n men $y_1, \ldots, y_n$ in a set of m venues $V = \{v_1, \ldots, v_m\}$. Each woman x_i has a list $X_i \subseteq V$ of venues that she would like to go to and each man y_j has a list $Y_j \subseteq V$ of venues that he would like to go to. However, each venue v_k only has capacity for c_k couples. Can you find a polynomial time algorithm to arrange the blind dates so that everyone goes to one of their preferred venues?

More precisely, can you find maps $f : \{x_1, \ldots, x_n\} \to V$ and $g : \{y_1, \ldots, y_n\} \to V$ such that

1. $\forall i \in [n], f(x_i) \in X_i$ (all women get one of their preferred venues)
2. $\forall j \in [n], g(y_j) \in Y_j$ (all men get one of their preferred venues)
3. $\forall k \in [m] : |\{i \in [n] : f(i) = v_k\}| = |\{j \in [n] : g(j) = v_k\}| \leq c_k$ (all venues have an equal number of men and women and are within their capacities)

Hint is available.
Note: 80% credit will be given if you answer this problem when there is unlimited capacity, i.e. $c_k = \infty$ for all k.

Hints

3. What happens if your path from s to t goes back and forth across the minimum cut?

4. Try routing flow from the women to the venues and then from the venues to the men.