Note: Your TA may not get to all the problems. This is totally fine, the discussion worksheets are not designed to be finished in an hour. The discussion worksheet is also a resource you can use to practice, reinforce, and build upon concepts discussed in lecture, readings, and the homework.

1 (MT2 Practice) MST True/False

For each of the following, state if it's true or false. If true, justify your answer, if false give a counterexample.

In each problem, assume G is a weighted undirected graph with non-negative edge weights. You may use the following fact in your justifications without proof: For any MST T of a graph, there is some sorted ordering of the edges such that Kruskal's will output T if it considers adding edges in this order.

(a) G has a unique minimum spanning tree if and only if all edges in G have distinct edge weights.

(b) Suppose all edges in G have weight at most W, and let G' have the same vertices and edges as G, but edge e has weight $W - w_e$ in G' instead of w_e. Then the maximum weight spanning tree of G is the minimum spanning tree of G'.

(c) Let G' have the same vertices and edges as G, but edge e has weight w_e^2 in G' instead of w_e. Then any minimum spanning tree in G' is also a minimum spanning tree in G.

(d) If the smallest edge weight is 1, every MST has the same number of edges with weight 1.

Solution:

(a) False. Consider a triangle graph with weights $1, 1, 2$. The MST of this graph is unique.

(b) True. This follows because any spanning tree has $|V| - 1$ edges. The weight of a spanning tree in G' is then $(|V| - 1)W - \sum_{e \in T} w_e$, i.e. maximizing the weight of this spanning tree is equivalent to minimizing the corresponding spanning tree's weight in G.

(c) True. This follows because squaring the weights maintains the sorted order of edges, so Kruskal's choices are unaffected by squaring weights, and so the same set of MSTs can be generated by Kruskal's.

(d) True. One way to see this: If we change all weight 1 edges to weight 0, then by the given fact the set of MSTs remains the same, which means the cost of every MST must have decreased by the same amount, which means they all must have the same number of weight 1 edges.

2 (MT2 Practice) Huffman and LP

Consider the following Huffman code for characters a, b, c, d: $a = 0, b = 10, c = 110, d = 111$.

Let f_a, f_b, f_c, f_d denote the fraction of characters in a file (only containing these characters) that are a, b, c, d respectively. Write a linear program with variables f_a, f_b, f_c, f_d to solve the following problem: What values of f_a, f_b, f_c, f_d that can generate this Huffman code result in the Huffman code using the most bits per character?

Solution:

Our objective is to maximize the bits per character used:

$$\max f_a + 2f_b + 3f_c + 3f_d$$

We know the fractions must add to 1 and be non-negative:

$$f_a + f_b + f_c + f_d = 1, f_a, f_b, f_c, f_d \geq 0$$

We know the frequencies of the characters must satisfy $f_a \geq f_b \geq f_c, f_d$. We also know that $f_c + f_d \leq f_a$, since we chose to merge (c, d) with b instead of merging a. So we get the following constraints:

$$f_c \leq f_b, f_d \leq f_b, f_b \leq f_a, f_c + f_d \leq f_a$$

3 (MT2 Practice) Discrete Golf

Professor Vazirani loves to play golf, specifically a version called discrete golf. The goal of discrete golf is to hit a golf ball from checkpoint 1 to checkpoint n on a golf course in as few strokes as possible. Based on the terrain of the course, he knows that from checkpoint i, he can hit the ball up to $d(i) \geq 1$ checkpoints away in one stroke. That is, if the ball is at checkpoint i, he can hit the ball to any of checkpoints $i + 1, i + 2 \ldots i + d(i)$ in one stroke.

(a) Suppose he hits the ball as far as possible every stroke, i.e. if he is at checkpoint i, he hits the ball to checkpoint $i + d(i)$. Give a counterexample where this greedy algorithm does not achieve the minimum number of strokes needed to reach checkpoint n from checkpoint 1.

(b) Suppose that all $d(i)$ are at most D. Give a $O(nD)$ time algorithm for computing the minimum number of strokes Professor Vazirani needs to reach point n. How much memory does this algorithm need?

Solution:

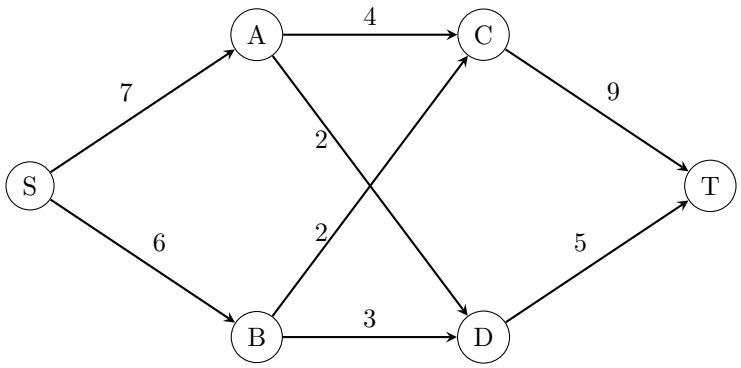
(a) Consider when $n = 5, d(1) = 2, d(2) = 3, d(3) = 1, d(4) = 1$. The greedy algorithm will visit checkpoints $1, 3, 4, 5$, the optimal route is to visit checkpoints $1, 2, 5$.

(b) Let $s(i)$ be the minimum number of strokes needed to reach checkpoint n from checkpoint i. Then the base case is $s(n) = 0$, and for $i < n$, $s(i) = 1 + \min_{i+1 \leq j \leq i+d(i)} s(j)$. The DP table has n entries, and each takes $O(D)$ time to update, so the runtime is $O(nD)$. Since each entry of the DP table only relies on the next D entries, it suffices to only store D entries, i.e. only $O(D)$ space is needed.

Equivalently, let $s(i)$ be the minimum number of strokes needed to reach checkpoint i from checkpoint 1. Then the base case is $s(1) = 0$, and for $i > 1$, $s(i) = 1 + \min_{i-D \leq j \leq i-1 : i \leq j+d(j)} s(j)$. The DP table has n entries, and each takes $O(D)$ time to update, so the runtime is $O(nD)$. Since each entry of the DP table only relies on the previous D entries, it suffices to only store D entries, i.e. only $O(D)$ space is needed.

4 (MT2 Practice) Bottleneck Edges

Consider the following network (the numbers are edge capacities):



(a) Find the following:

- A maximum flow f, specified as a list of $s-t$ paths and the amount of flow being pushed through each.
- A minimum cut (the set of edges with the smallest total capacity, whose removal disconnects S and T), specified as a list of edges that are part of the cut.

(b) Draw the residual graph G_f (along with its edge capacities). In this residual network, mark the vertices reachable from S and the vertices from which T is reachable.

(c) An edge of a network is called a *bottleneck edge* if increasing its capacity results in an increase in the maximum flow. List all bottleneck edges in the above network.

(d) Give a very simple example (containing at most four nodes) of a network which has no bottleneck edges.

(e) Give an efficient algorithm to identify all bottleneck edges in a network. (Hint: Start by running the usual network flow algorithm, and then examine the residual graph.)

Solution:

(a) The maximum flow is given by the following sequence of updates:

- Route 4 units of flow along $S-A-C-T$
- Route 2 units along $S-A-D-T$
- Route 2 units along $S-B-C-T$
- Route 3 units along $S-B-D-T$

The resulting flow is feasible and has value 11. It produces the mincut $(\{S,A,B\},\{C,D,T\})$ (the edges across the cut are (A,C), (A,D), (B,C), (B,D)) of capacity 11, certifying the optimality of the flow.

Notice that $(\{S,A,B,D\},\{C,T\})$ (edges (A,C), (B,C), (D,T)) is also a mincut.

(b) The residual graph is shown in the figure. Vertices S,A and B are reachable from S. T can be reached from vertices C and T.

-.5.5

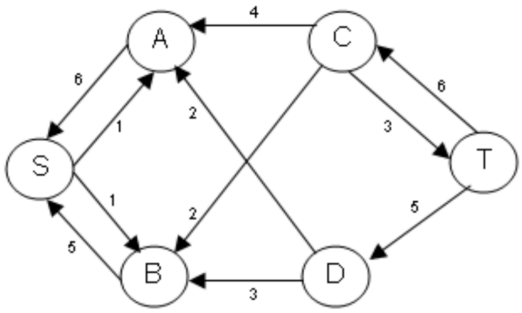


Figure 1: The residual graph

(c) (A,C) and (B,C) are bottleneck edges. The other edges belonging to a mincut are not bottleneck, as increasing their capacity does not increase the capacity of the minimum (s,t)-cut.

(d) The following figure shows an example with no bottleneck edges. The optimal flow saturates all edges, but augmenting the capacity of any of them does not increase the capacity of the minimum cut, i.e. does not open up any new path for flow to run from source to sink.

-.5.5

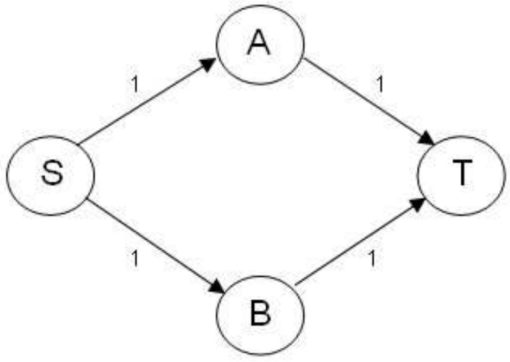


Figure 2: An example with no bottleneck edges

(e) Run the usual network flow algorithm and consider the final residual graph. Let S be the set of vertices reachable from s and T the set of vertices from which t is reachable in this graph. By the optimality of the flow S and T must be disjoint, as they are separated by a saturated cut. Suppose now that $e = (u, v)$ is a bottleneck edge. As we increase e's capacity we are able to route flow from s to u, through e and from v to t in the residual graph. This implies that $u \in S$ and $v \in T$. Moreover, if an edge $e = (u, v)$ has $u \in S$ and $v \in T$ (notice that e must then be in a minimum cut), increasing its capacity allows us to route flow from s to u (as $u \in S$), through the new capacity of e and to t (as $v \in T$). Hence, the set of bottleneck edges is the set $E(S, T)$, i.e. the set of edges which originate in S and end in T.

The running time is dominated by computing the max flow. Using the Ford-Fulkerson algorithm from the lecture and textbook, this is $O(|E| \cdot f)$, where f is the size of the max flow.

5 (MT2 Practice) Permutation Games

A permutation game is a special form of zero-sum game. In a permutation game, the payoff matrix is n-by-n, and has the following property: Every row and column contains exactly the entries $p_1, p_2, \ldots p_n$ in some order. For example, the payoff matrix might look like:

$$P = \begin{bmatrix} p_1 & p_2 & p_3 \\ p_2 & p_3 & p_1 \\ p_3 & p_1 & p_2 \end{bmatrix}$$

Given an arbitrary permutation game, describe the row and column players' optimal strategies, justify why these are the optimal strategies, and state the row player's expected payoff (that is, the expected value of the entry chosen by the row and column player).

Solution: Both players' optimal strategy is to choose a row/column uniformly at random. The expected payoff of this strategy is $\sum_k p_k/n$.

To show these are optimal, note that if the column player picks this strategy, any strategy the row player chooses has expected payoff $\sum_k p_k/n$. By symmetry, this also implies that if the row

player picks this strategy, any strategy the column player picks achieves the same expected payoff. By duality, this must be the optimal pair of mixed equilibrium strategies.