Midterm 2

Name:

SID:

Rules and Guidelines

- **This checkpoint is out of 101 points and you will have a week to complete it.**
- Answer all questions. Read them carefully first. Not all parts of a problem are weighted equally.
- Be precise and concise.
- The problems may **not** necessarily follow the order of increasing difficulty.
- Any algorithm covered in the lecture can be used as a blackbox unless stated otherwise.
- Please either latex your answers, or provide a neatly handwritten version of the solutions where every subpart is on a new page of your submission and the number is clearly labelled.
- Good luck!

Discussion Section

Which of these do you consider to be your primary discussion section(s)? Feel free to choose multiple, or to select the last option if you do not attend a section.

(a) Fred Zhang Wheeler 202 • Tu 5-6pm

(b) Jialin Li Etcheverry 3105 • Tu 5-6pm

(c) Tiffany Chien Etcheverry 3109 • W 9-10am

(d) Sidhanth Mohanty Wheeler 130 • W 9-10am

(e) Teddy Tran Moffitt Library 106 • W 10-11am

(f) Emaan Hariri Hearst Field Annex B5 • W 12-1pm

(g) Dee Guo Wheeler 224 • W 12-1pm

(h) Arpita Singhal Dwinelle 234 • W 1-2pm

(i) Gillian Chu Barrows 136 • W 1-2pm

(j) Rachel De Jaen Barrows 155 • W 1-2pm

(k) Joshua Turcotti Wheeler 20 • W 2-3pm

(l) Varun Jhunjhunwalla Wheeler 202 • W 2-3pm

(m) Vishnu Iyer Etcheverry 3113 (Advanced Section) • W 2-3pm

(n) Avni Singhal Wheeler 30 • W 3-4pm

(o) Jiazheng Zhao Dwinelle 229 • W 3-4pm

(p) Rishi Veerapaneni Wheeler 224 • W 3-4pm

(q) Jeff Xu Evans 9 • W 3-4pm

(r) Noah Kingdon Wheeler 202 • W 5-6pm

(s) Neha Kunjal Hildebrand B51 • Th 11-12pm

(t) Christina Jin Evans 9 • Th 12-1pm

(u) Noah Krakoff Moffitt Library 103 • Th 1-2pm

(v) Ajay Raj Wheeler 202 • Th 2-3pm

(w) Don't attend section.

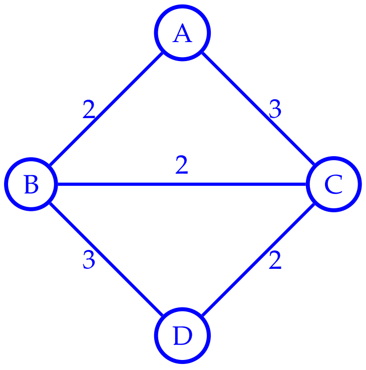
1 True or False? (14 points)

For each part, give a brief proof if you think it's true, or provide a counter example if you believe it is false.

(a) (**2 points**) For any undirected graph with all positive edge weights, there exists a vertex v such that the tree of shortest paths rooted at v is an MST of the graph.

◯ True ◯ False

Solution: False: MST minimizes total weight, not the weights of the shortest paths. Consider the counterexample below. The MST contains the edges AB, BC, and CD with total weight 6. There is no shortest paths tree starting from any vertex that has weight 6.



(b) (**2 points**) Consider a linear program with a minimization objective. If there are infinitely many integral feasible solutions, where all variables take integral values, then the objective value can be made arbitrarily small.

◯ True ◯ False

Solution: False: Consider min x subject to $x \geq 0$. The LP has infinitely many integral feasible solutions, that is, $\{0, 1, 2, \cdots\}$, but the optimal solution is just $x = 0$.

(c) (**2 points**) Let A be a $m \times n$ matrix, b be a m-dimensional vector, and c be a n-dimensional vector. If the linear program

$$\min_x c^\top x \quad \text{such that} \quad Ax \geq b, x \geq 0$$

has a feasible solution with objective value M, then every feasible point of its dual has an objective value at most M.

◯ True ◯ False

Solution: True: this is the definition of weak duality.

(d) (**2 points**) If a graph has a unique lightest edge, then that edge must be part of *every* MST of the graph.

◯ True ◯ False

Solution: True: Suppose the unique lightest edge is $e = (u, v)$. Consider the cut $\{u\}$ and the rest of the graph. Since e is the unique lightest edge among the cut edges in this cut, the cut property guarantees it must be chosen by any MST.

(e) (**2 points**) Let F_1 and F_2 be feasible s-t flows in a directed graph G, satisfying the flow conservation and capacity constraints. Then the flow F defined as $F(e) := 0.6F_1(e) + 0.4F_2(e)$ for all $e \in E$ is also a feasible s-t flow in G.

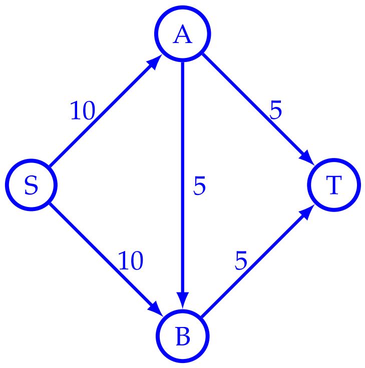
○ True ○ False

Solution: True: the set of feasible flows is a convex set. Any convex combination of feasible flows is feasible.

(f) (**2 points**) We say an edge is *saturated* by a flow if the amount of flow on the edge equals its edge capacity. In a directed graph, let e be an edge saturated by some maximum s-t flow of value F. Let c be its capacity. If we reverse e, then in the new graph any maximum s-t flow must have value at most $F - c$.

○ True ○ False

Solution: False. Consider the counterexample below. Maximum flow of 10 units results from sending 5 units along SABT and 5 units along SAT. Edge AB is used to capacity. After reversing AB, we can achieve the same amount of flow by sending 5 units along SAT and 5 units along SBT.



(g) (**2 points**) Consider maximum s-t flow in a directed graph. Let T be the number of incoming edges of t. If we add a constant $C > 0$ to all edge capacities, then the value of maximum s-t flow will increase by $C \cdot T$.

○ True ○ False

Solution: False: There might be vertices connected to t that s can't reach. For example, consider t has many incoming edges like (u_1, t), (u_2, t) and so on, but u_i doesn't have any incoming edge.

2 Quick Union-Find (6 points)

Consider a disjoint set data structure $\mathcal{S}$ instantiated on the universe $\{1, \ldots, n\}$. Some arbitrary number of Union and Find operations has been performed on $\mathcal{S}$.

(a) (**2 points**) Assume $\mathcal{S}$ only uses union-by-rank, without path compression. Give the tightest possible upper bound on the average time complexity of running 1000 additional Find operations. Give your answer in big-O notation as a function of n.

Solution: $O(\log n)$.

(b) (**2 points**) Assume $\mathcal{S}$ uses union-by-rank **with path compression**. Give the tightest possible upper bound on the average time complexity of running 1000 additional Find operations. Give your answer in big-O notation as a function of n.

Solution: $O(\log n)$. Just the first find takes $\mathcal{O}(\log n)$, this is enough to bring the average up to $\mathcal{O}(\log n)$.

(c) (**2 points**) Assume $\mathcal{S}$ uses union-by-rank **with path compression**. Then the time complexity of each individual Find is at most $\mathcal{O}(\log^* n)$. Mark if this is true or false, and justify.

◯ True ◯ False

Solution: False. As in b), the first find is at worst $\mathcal{O}(\log n)$ individually. It is true that the time complexity of $m > n$ Finds is $\mathcal{O}(m \log^* n)$, but this is only amortized, and not for individual Finds.

3 Space/Time complexity (9 points)

Define a function $S(i, j)$ recursively by:

$$S(i, j) = \max\left\{S(i-2, j-1) + 4, S(i-1, j-1) + 7, S(i-1, j)\right\}$$

when $i \in [0, m]$, $j \in [0, n]$ and $m < n$.

Notice how this function is not fully specified due to a lack of base cases.

(a) (**3 points**) Describe a minimal set of base cases that would make this a well-defined recurrence. We call (i, j) a base case if it is not in $[0, m] \times [0, n]$. Set the value of S for each base case to be 0. Make sure to only include the base cases that are necessary.

Solution: The bases cases are as follows:

$$\forall j \in [0, n-1], \forall i \in [-2, -1], S(i, j) = 0$$

$$\forall i \in [-2, m-1], S(i, -1) = 0$$

$$S(-1, n) = 0$$

(b) (**2 points**) For all possible algorithms, give the tightest asymptotic bound on the **space complexity** of computing $S(m, n)$ using dynamic programming/memoization.

Solution: The optimal space complexity would be $O(m)$ because m is less than n and you only depend on subproblems that are at most $j - 1$. So you only need to keep track of the previous column, plus one column for the column you are currently computing.

(c) (**4 points**) Which of the following recurrence relations would take $O(n^2)$ time to compute if memoized and all subproblems in the specified range have to be computed? You may assume that each of the functions below are equal to 0 if any of their inputs are negative.
Note: there may be multiple answers.

i) $S(1) = 5, S(x) = \min_{1 \le k \le x-1} S(k) + S(k)^2$ for $x \in [2, n]$.

ii) $S(i, j) = \max\left\{S(i-3, j) + 9, S(i, j-1) + 3\right\}$ where $i \in [0, n], j \in [0, n]$

iii) $S(i, j, k) = \max\left\{S(i-3, j, k-1) + 2, S(i, j-1, k-1) + 3\right\}$ where $i \in [0, n], j \in [0, n], k \in [0, n/2]$

iv) $S(i, j, k) = \max\left\{S(i-3, j, k) + 9, S(i, j-1, k-1) + 3\right\}$ where $i \in [0, n], j \in [0, n], k \in [0, 3]$

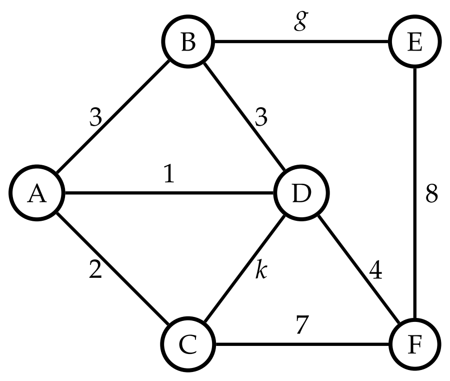
Solution:
i, ii, iv should be selected.
i) n subproblems, which take at most n work to compute, so $O(n^2)$.
However as students pointed out you can actually do this is constant time, as

$$S(k) + S(k)^2$$

is a monotonically increasing function so when k ¿= 2 it is always 30. Thus the answer can be computed in $O(1)$ which is $O(n^2)$. Since this wasn't intended, we gave everyone points for this part.
ii) n^2 subproblems, which take constant time to compute, so $O(n^2)$.
iii) n^3 subproblems, which takes constant time to compute, $O(n^3)$.
iv) n^2 subproblems, which takes constant time to compute, $O(n^2)$.

4 To choose or not to choose (12 points)

We are given the following graph G:



a) (**2 points**) For what range of values of g are you **guaranteed** to include edge BE in the MST?

Solution: Construct a cut where one set is E and the other set is (A, B, C, D, F). In order to guarantee g is in the MST, it must be the lightest edge. $g < 8$.

b) (**2 points**) For what range of values of k are you **guaranteed** to include edge CD in the MST?

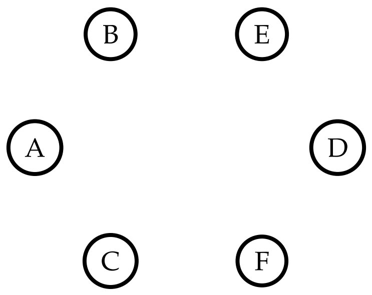
Solution: There is a cycle using edges AD, AC, and CD. The heaviest edge in the cycle will not be used and so in order to make CD not the heaviest edge $k < 2$.

c) (**2 points**) For what range of values of k are you **guaranteed to NOT** include edge CD in the MST?

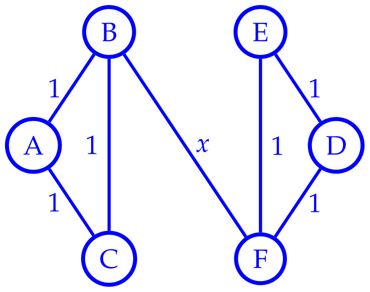
Solution: There is a cycle using edges AD, AC, and CD. The heaviest edge in the cycle will not be used and so in order to make CD the heaviest edge, $k > 2$.

d) (**3 points**) Suppose an adversary can set g and k arbitrarily. What is the maximum cost of a MST that the adversary can force? **Solution:** MST will only choose BE and CD if it is better than or equal to cuts of the other edges, so the maximum cost is $2 + 1 + 3 + 4 + 8 = 18$.

e) (**3 points**) Draw seven edges on the following graph with six vertices, and then mark six edges with 1 and one edge with x such that an adversary can choose x such that the cost of the MST is arbitrarily large.



Solution:



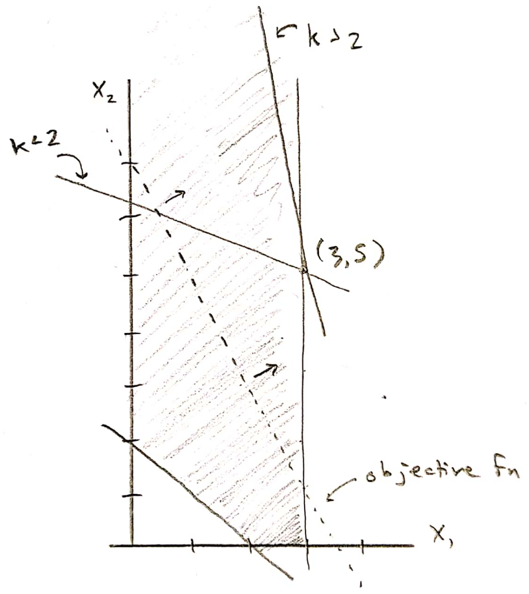
5 Find the unknown coefficient (9 points)

(a) (**4 points**) Consider the following linear program in which the value of k is unknown. Find the range of values of k such that the point $(x_1, x_2) = (3, 5)$ is optimal.

$$\begin{aligned} \max 2x_1 + x_2 \\ x_1 + x_2 \geq 2 \\ x_1 \leq 3 \\ kx_1 + x_2 \leq 3k + 5 \\ x_1, x_2 \geq 0 \end{aligned}$$

Solution: Note that $(3, 5)$ is always feasible, regardless of the value of k. The constraint with k (3rd condition) is always tight at $(3, 5)$. Thus adjusting k simply modifies the slope of the line that runs through $(3, 5)$. The slope of this line is $-k$ (plot the 3rd condition as $x_2 = 3k + 5 - kx_1$).

The slope of the objective function is -2. If the slope of condition 3 is steeper than -2 (i.e. smaller than -2), then our objective function would keep pushing to the right, surpassing the point $(3, 5)$. Thus, for $(3, 5)$ to be optimal the slope of the constraint must be greater than -2: $-k \geq -2 \rightarrow k \leq 2$.



(b) (**3 points**) Construct the dual of the linear program in part a. Leave k in your answer.
Solution: The primal is:

$$\begin{aligned} \max 2x_1 + x_2 \\ -x_1 - x_2 \leq -2 \\ x_1 \leq 3 \\ kx_1 + x_2 \leq 3k + 5 \\ x_1, x_2 \geq 0 \end{aligned}$$

The dual of this is:

$$
\begin{aligned}
\min 3ky_3 + 5y_3 + 3y_2 - 2y_1 \\
-y_1 + y_2 + ky_3 \geq 2 \\
-y_1 + y_3 \geq 1 \\
y_1, y_2, y_3 \geq 0
\end{aligned}
$$

(c) (**2 points**) Convert the linear program **in part a** to be of the form where your objective function is a minimization function, and your constraints are all in the form variable $\geq$ equation (e.g. $x_1 \geq 5$). Leave k in your answer.
Note that this question is not asking you to find the dual
Solution:

$$
\begin{aligned}
\min -2x_1 - x_2 \\
x_1 + x_2 \geq 2 \\
-x_1 \geq -3 \\
-kx_1 - x_2 \geq -3k - 5 \\
x_1, x_2 \geq 0
\end{aligned}
$$

6 Trees of Candles (14 points)

You are an interior decorator, confronted with a dark living room. To lighten the room up, you have n candles and want to build k trees of candles. First, the candles $\{1, \ldots, k\}$ are already put on the floor, to serve as roots of the k trees. You would like to put the remaining $n-k$ candles $\{k+1, \ldots, n\}$ on top of them to form k trees $\{T_1, \cdots, T_k\}$ of candles, where each T_i is rooted at candle i.

You can put an arbitrary number of candles on top of the same candle. The cost to putting candle i on top of candle j is $C(i,j)$. We assume that $C(i,j) \geq 0$ and $C(i,j) = C(j,i))$ for $i \neq j$. No candle can be reused or put on top of itself. You cannot use the candles $\{1, \ldots, k\}$ anymore, as they have already been fixed to the floor.

(a) (**6 points**) Describe an algorithm to find a solution of minimum cost.

Grading Remark: We will only look at your part b) and c) if you got part (a) correct; Any mistake in algorithm design would render an automatic 0 in the other two parts. Any re-grade request won't help.

Solution: Create a complete weighted undirected graph on n vertices, where the edge weights are distances. Call the graph G. Add a dummy node q to G which has k edges, one to each of the candles $1, \ldots, k$. Let the new grpah be G'. Each of these edges has weight -1, for example. We then compute the MST of G', then remove q to get the trees.

(b) (**5 points**) Prove the correctness of your algorithm.

Solution: If you take the cut $\{q, x_1, \ldots, x_{n+k}\} \backslash \{x_{n+i}\}$ versus $\{x_{n+i}\}$, the edge (q, x_{n+i}) is the unique cheapest edge crossing this cut and thus must be in any MST of G'. Thus any MST of G' takes all these -1-weighted edges incident on q, which means the cluster heads will be in distinct trees (the clusters).

For optimality, note that if there is a cheaper way to build the k trees $\{T_1, \cdots, T_k\}$ in G (with each rooted at node $i \in \{1, \cdots, k\}$), then it corresponds to a cheaper spanning tree of G'.

(c) (**3 points**) Give a runtime analysis of your algorithm in the previous part.

Solution: Since the graph has at most $n+$ vertices and $\Theta(n^2)$ edges, the runtime of either Kruskal or Prim is $O(n^2 \log n)$. Forget about heap, naive Prim would be able to give us $O(n^2)$.

7 Fancy Sequence (11 points)

Let $A[1], \dots, A[n]$ be an array. Call a (not necessarily contiguous) subsequence $[A[i_1], \dots, A[i_k]]$ *fancy* if $k \le 2$, or if for all $j \notin \{1, k\}$, we have $17A[i_j] < A[i_{j+1}] + A[i_{j-1}]$.

This question asks you to provide an algorithm to compute the length of the longest fancy subsequence of a given length-n array A. The sequence need not be contiguous. You may assume arithmetic operations (addition, division and multiplication) and comparisons ($<$, $>$, $==$) can be done in constant time.

(a) (**6 points**) For $i < j$, let $f(i, j)$ denote the length of the longest fancy subsequence of $A[i..n]$ whose first two elements are $A[i]$ and $A[j]$. Write a recurrence relation for $f(i, j)$ and specify the base case(s).

(b) (**3 points**) Give a polynomial-time algorithm to compute the length of the longest fancy subsequence of A. You may write pseudocode.

(c) (**2 points**) Give a runtime analysis of your algorithm.

Solution:

(a) The recurrence is given by

$$f(i, j) = 2 + \max\{f(j, k) - 1 \mid j < k \le n \text{ and } A[i] + A[k] > 17A[j]\}.$$

where we define $\max \varnothing = 0$. This also handles the base case.

(b) We fill in the table in reverse row order and from right to left. In the end, we return the maximum entry in the entire table: $\max_{1 \le i < j \le n} f(i, j)$.

Algorithm 1: Fancy Subsequence

```
for $i \leftarrow n-1$ down to $1$ do
    for $j \leftarrow n$ down to $i+1$ do
        $m \leftarrow 0$
        for $k \leftarrow j+1$ to $n$ do
            if $A[i] + A[k] > 17A[j]$ then
                $m \leftarrow \max\{m, f(j,k) - 1\}$
        $f(i,j) \leftarrow 2 + m$
return $\max_{1 \le i < j \le n} f(i,j)$
```

(c) $O(n^3)$

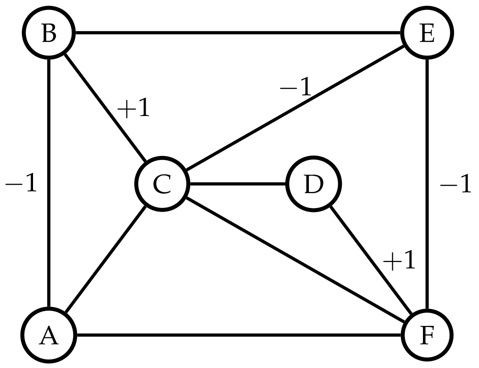
8 Switchings and Signings (14 points)

Let $G = (V, E)$ be a connected, undirected graph on n vertices as input. Every edge e has a sign $\sigma(e) \in \{-1, +1\}$, initially equal to $+1$. We will call an assignment of signs to all the edges in G a *signing*.

We call a signing σ *balanced* if **for every cycle** C in G, the product of signs of edges in the cycle is equal to 1, i.e.,

$$\prod_{e \in C} \sigma(e) = 1.$$

(a) (**2 points**) A partial signing is given for the below graph. Sign the remaining edges so that the resulting signing is balanced.



BE	
CD	
CF	
AC	
AF	

Solution:

BE	-1
CD	+1
CF	+1
AC	-1
AF	-1

You are allowed to perform the following operation known as a *switching* — choose a vertex v and toggle the sign of all edges incident to v, i.e., update the sign on every edge e incident to v from $\sigma(e)$ to $-\sigma(e)$.

(b) (**4 points**) Prove that any signing of G constructed via a sequence of *switching* operations must be balanced. *Recall that initially all the edge signs are* $+1$.

Solution: For vertex v, let $\gamma_v = -1$ if a switching operation is performed on v and $+1$ otherwise. The sign on edge uv is then $\gamma_u\gamma_v$. Let $v_1v_2 \dots v_k$ be any cycle in the graph. The product of signs of edges in the cycle is $(\gamma_{v_1}\gamma_{v_2})(\gamma_{v_2}\gamma_{v_3}) \dots (\gamma_{v_k}\gamma_{v_1}) = \prod_{i=1}^{k} \gamma_{v_k}^2 = 1$.

(c) (**4 points**) Let T be a spanning tree of G, and σ_T be an arbitrary signing of edges in T. Prove that there is a unique balanced signing σ of G such that $\sigma(e) = \sigma_T(e)$ for all $e \in T$.
Hint: first try to show that there is at least one such balanced signing (part (b) might be useful for this), and next show that there is at most one balanced signing that agrees with σ_T.

Solution: To prove the desired statement, we (i) prove that there exists a balanced extension of σ_T, (ii) prove that there is at most one balanced extension of σ_T. To prove (i), observe that σ_T can be induced by a sequence of switching operations obtained via the following algorithm.

> Fix a root r, and perform a switching operation on vertex v if and only if the product of edges along the path from r to v is equal to -1.

Performing these switching operations on G induces a balanced signing which is equal to σ_T on T. To prove (ii), observe that in any balanced extension σ, for any edge $uv \notin T$, σ_{uv} must equal $\prod_{e \in P_{uv}} \sigma_T(e)$.

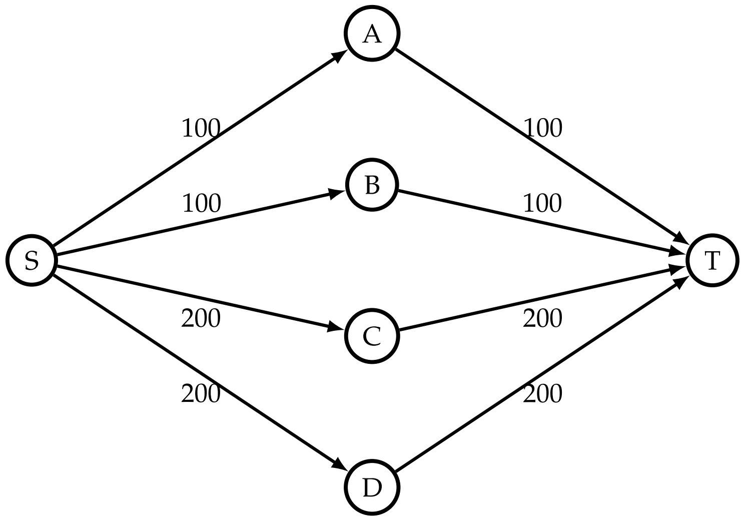
(d) (**4 points**) Let σ be a balanced signing of G. Show that there exists a sequence of *switching* operations that produces σ and give an algorithm that takes in G and σ as input and produces this sequence of *switching* operations. Prove the correctness and analyze the running time of your algorithm.
Hint: consider using the statement in part (c).

Solution: Let T be an arbitrary spanning tree of G, and let σ_T be the restriction of σ to edges of T. From part (c), σ is the unique balanced extension of σ_T. From the solution to part (c) there is a sequence of switching operations S to induce σ_T on T, and since the signing induced on G by S is balanced, it must equal σ. The algorithm is then to (1) find a spanning tree of G, (2) perform the algorithm from part (c) to find a relevant sequence of switching operations and perform them.

9 Flowww (12 points)

Recall that in each iteration of the Ford-Fulkerson method, we push as much flow as possible along an augmenting path from S to T. For this problem, consider all possible executions of Ford-Fulkerson, which can use any augmenting path, and Edmonds-Karp, an implementation of Ford-Fulkerson that uses BFS to select the augmenting path in each iteration. All edge capacities in the network must be non-negative integers.

a) (**6 points**) Add one directed edge and its capacity to the graph below that **maximizes** the **largest possible** number of iterations of Ford-Fulkerson. If no edge can be added to strictly increase the maximum number of iterations, leave the graph blank and fill in the "No such edge" box.



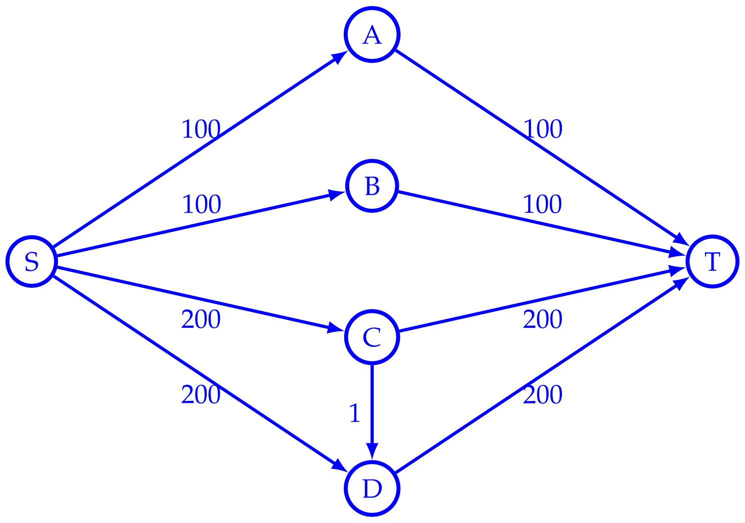
☐ No such edge

i) What is the new maximum number of iterations? ☐

ii) What are the paths of the flow, as well as the units pushed across each path, found by the algorithm once it is run to completion? You may not need all the rows in the table.

Path	Units

Solution:



A clearly drawn or specified edge, *with direction and capacity*, was necessary in order to receive credit. We gave partial credit if you increased the largest possible number of iterations but did not find the maximum, as long as all parts were correct with respect to the edge added.

Note: Adding an edge DC with capacity 1 is equivalent.

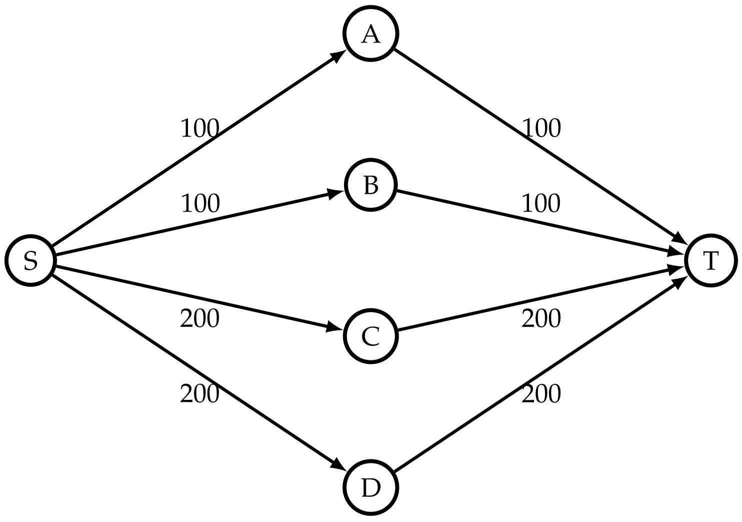
i) Number of iterations: 402
Repeatedly push 1 unit flow back and forth along edge CD or DC using paths SCDT and SDCT. This occurs a total of 200 + 200 times. Then push 100 units of flow along SAT and SBT. Total number of iterations is 200 + 200 + 2 = 402.

ii)

Path	Units
SAT	100
SBT	100
SCT	200
SDT	200

Note that we accepted augmenting paths due to ambiguity in the problem wording, but were looking for the final paths which achieve the maximum flow.

b) (**6 points**) Add one directed edge and its capacity to the graph below that **maximizes** the **largest possible** number of iterations of Edmonds-Karp. If no edge can be added to strictly increase the maximum number of iterations, leave the graph blank and fill in the ”No such edge” box.



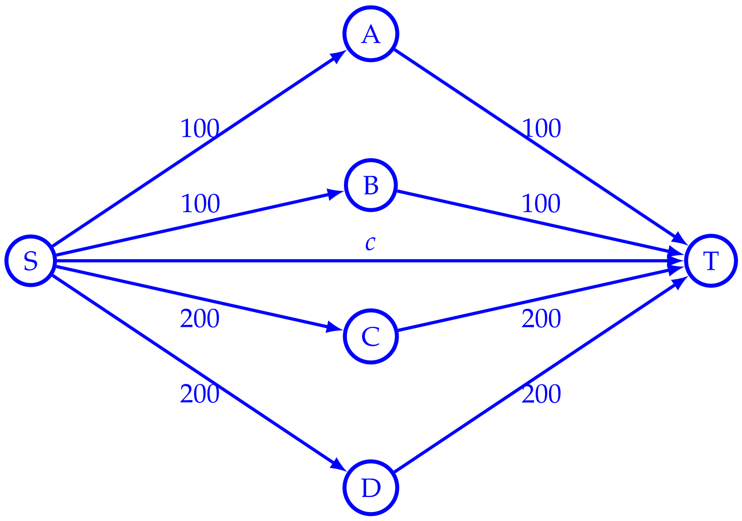
☐ No such edge

i) What is the new maximum number of iterations? $\boxed{}$

ii) What are the paths of the flow, as well as the units pushed across each path, found by the algorithm once it is run to completion? You may not need all the rows in the table.

Path	Units

Solution:



A clearly drawn or specified edge, *with direction and capacity*, was necessary in order to receive credit. We gave partial credit if you increased the largest possible number of iterations but did not find the maximum, as long as all parts were correct with respect to the edge added.

An edge from S to T of any capacity c would work.

i) Number of iterations: 5

ii)

Path	Units
SAT	100
SBT	100
SCT	200
SDT	200
ST	c