Flows & Linear Programs
Problem Set 6 – CS6515 (Spring 2025)

- This problem set is due on **Thursday March 6th**.
- Submission is via Gradescope.
- Your solution must be a typed pdf (e.g., via LaTeX, Markdown, etc. Anything that allows you to type math notation) – no handwritten solutions.
- Please try to make your solutions as concise and readable as possible. Most problems will have solutions that are no more than a page long. Consider using bullet points and adding space to break up large paragraphs into smaller chunks.
- There are 3 problems. Each problem is graded with 20p. **There is +1p bonus per problem** for stating (i) how long it took you to solve that problem, and (ii) how long it took you to type the answer.

16 Fixing Flow After Deleting an Edge

We are given a directed graph $G = (V, E)$ with unit-capacities, meaning every edge has capacity 1. We are also given the maximum flow $f \in \mathbb{R}^E$. Now an attacker deletes some edge e from the graph, so the flow f might no longer be valid (i.e., it might try to send flow through e, but e no longer exists).

1. Design an algorithm that, given G, f, e, returns the new maximum flow. (Add a brief explanation why it's correct. This is to help the TA understand your algorithm, and allow for partial points if the algorithm does not work.)
2. Analyze your algorithm's time complexity.

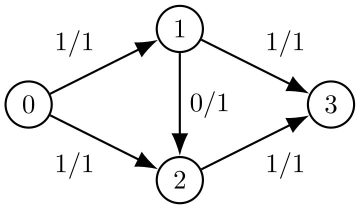
Hint: Let's assume edge (u, v) carries flow and now that edge is deleted. Now the old flow might not be valid anymore because flow arrives at u without leaving it. Likewise, flow leaves v without first having arrived there. I suggest to do 3 steps: (i) get rid of the extra flow to u. (ii) get rid of the extra flow leaving v. (iii) you now have a valid flow, but is it maximum?

16.1 Programming option

Grades: Gradescope will automatically test your submission. You can resubmit as often as you want (until the due date) and you can see on gradescope how many test cases you currently passed. There are no partial points for passing part of the test cases, because even very wrong solutions can pass some test cases. For partial points, submit a theory/pdf solution instead. While your submission must pass all test cases on gradescope, the final pass/fail decision is made by a TA. (This is to make sure your submission is actually solving the problem described above. For example, hardcoding the output is obviously not a valid solution.) We need this rule because all test cases are public (on Piazza) to help you test/debug your program.

Input There are code templates on Piazza that handle reading the input. The input will be of the following format. The first line is n and m, separated by space, representing $n = |V|, m = |E|$. The next m lines contain two numbers, separated by space, specifying directed edges (u, v), where $0 \leq u, v \leq n - 1$. The next line is an m character long binary string. The i-th character represents the flow on the i-th edge. The last line is some integer k, representing that edge number k is to be deleted. (Here we count from 0, so $k = 0$ means the first edge is deleted.) For the flow, we always have $s = 0$ and $t = n - 1$, i.e., the flow goes from vertex 0 to vertex n-1.
Example input:

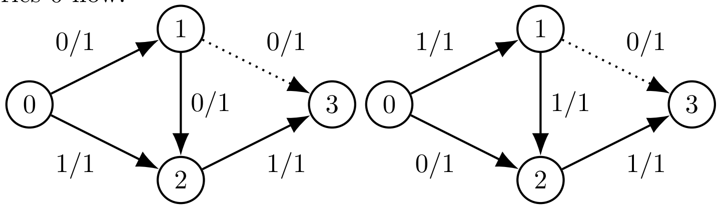
```
4 5
0 1
0 2
1 2
1 3
2 3
11011
3
```



This represents the graph on the right. The numbers inside the nodes are $0, ..., n-1$, representing the node index. E.g., line "`0 1`" is the edge $(0, 1)$ from node 0 to node 1.

The edge at index 3 is to be deleted (here we count from 0, so it's actually the 4th edge), i.e., edge $(1, 3)$.

The new valid max flow would be one of the following, where the dotted edge was deleted so it now carries 0 flow:



The expected output would be

```
01001
```

or

```
10101
```

The autograder will accept either solution.

17 Procurement

Consider the following extension of the bakery problem we discussed in class: we have a certain amount of flour and eggs in storage, and have two different recipes for bread. Each unit of bread requires the following units of flour, egg, and spice, and can be sold for the following price:

Recipe	Flour	Egg	Spices	Sell-price
Bread A	4	10	1	60
Bread B	9	3	1	65

We have some of the ingredients in storage, but we can also buy more from sellers. The sellers do not have an infinite amount, so there is a limit to how much we can buy of each seller. Each seller also offers different prices. The following table provides the different sources for ingredients (own storage and sellers) and the respective prices for buying one unit of the respective ingredients.

Seller/Source	Flour Amount	Flour Price	Egg Amount	Egg price	Spice Amount	Spice Price
(our storage)	4	0 (free)	8	0 (free)	0	0 (free)
Alice	20	2	30	3	40	6
Bob	25	4	30	7	5	5

We want to figure out (i) how much of each type of bread to bake, and (ii) how much of each ingredient to buy of each seller. All units here are in weight, so it's fine if the answers are not whole integers. The aim is to maximize the profit, i.e., sell price of baked goods minus expenses of buying ingredients.

1. Give a linear program that models the above problem, by answering the following questions:
   (a) List the variables, and describe what they mean (e.g., "x_A is the amount of bread A we bake,...")
   (b) Give the linear cost function. (In form like "$\max 10x + 4y - 10z$" etc)

(c) Give the linear constraints. (In form like "$x + 2y - 4z \leq 99$" or $x \geq 0$ etc)

2. (+2p bonus) Solve the linear program (e.g., use a linear programming library for your favorite programming language. Or use an online linear program solver[1]) and give the following information:

Recipe	Amount produced
A	___
B	___

Seller	Flour Bought	Egg Bought	Spices Bought
Alice	___	___	___
Bob	___	___	___

Total profit made:
___

Remark: To verify your results: the expected total profit should be 165.67 (i.e., $\frac{497}{3}$).
If you got a different number then your linear program is incorrect, or you made an error while using the linear program solver.

18 Min-Cost Flow as Linear Program

We are given a directed graph $G = (V, E)$ with edge capacities $c \in \mathbb{R}^E$ (ie, c_e is the capacity of edge e) and edge-costs $w \in \mathbb{R}^E$ (i.e., it costs $w_e \cdot x$ to send x amount of flow through edge e). We are also given a demand-vector $d \in \mathbb{R}^V$, where for $v \in V$, d_v is how much flow should stay at vertex v. E.g., if $d_v = 2$ for some vertex v, then it would be fine if a total amount of flow of 5 arrives at v and 3 amount of flow leave v, as then the net-amount staying at v is 5-3=2. When d_v is negative, it means that $|d_v|$ flow should originate at v. E.g., if $d_v = -3$, then it would be fine if 1 unit of flow arrives at v, and 4 flow leave v, as then the net-amount of flow originating at v is $4 - 1 = 3$.

Problem: We want to find the flow that satisfies the demands, while also minimizing the cost. Model this problem as a linear program. The linear program will have $|E|$ many variables: one variable f_{uv} for each $(u, v) \in E$.

1. Give the linear cost function.

2. What are the linear constraints that model how much flow stays/originates at each vertex?

3. What are the linear constraints that enforce the flow not exceeding edge capacities?

4. What are the constraints that enforce that the flow cannot go backwards along edges?

Remark: Recall that in addition to inequality constraints (e.g., $10x + 3y - 4z \leq 2$ or $2x - y + 3z \geq 2$) we also allowed to use equality constraints (e.g., $-2x + 2y + 3z = 99$).

Another Remark: While we could try to come up with some new augmenting-path-style algorithm to solve minimum cost flow, we can also be lazy and just import a linear programming library and solve the linear program from Q18.

That's what makes linear programs so powerful. They allow us to solve many different problems without having to invent a new algorithm for each problem.
It comes at a cost though. Since linear programs are so general, solving them is often slower than algorithms tailored to specific applications.

[1]For example, https://online-optimizer.appspot.com/?model=builtin:default.mod