CS218, Winter 2024, **Due: 11:59pm, Wed 03/13, 2024**
Assignment #5 [1] **Training Programming Due: 11:59pm, Wed 03/06, 2024**
Name: **Student ID:**

Deadline. The homework is due **11:59pm, Wed 03/13, 2024**. You must submit your solutions (in pdf format generated by LaTeX) via GradeScope. The training programming assignment is due earlier, see more details below.

Late Policy. You have up to four grace days (calendar day) for the entire quarter. You don't lose any point by using grace days. If you decide to use grace days, please specify how many grace days you would like to use and the reason at the beginning of your submission.

Collaboration Policy. You can discuss the homework solutions with your classmates. You can get help from the instructor, but only after you have thought about the problems on your own. It is OK to get inspiration (but not solutions) from books or online resources, again after you have carefully thought about the problems on your own. However, you **cannot copy anything from other source**. You **cannot** share your solution/code with anyone else. You **cannot** read other's solution/code. If you use any reference or webpage, or discussed with anyone, you must cite it fully and completely (e.g., *I used case 2 in the examples in the Wikipedia page `https://en.wikipedia.org/wiki/Master_theorem_(analysis_of_algorithms)` about Master theorem for Problem 3*, or *I discussed problem 2 with Alice and Bob.*). *If you use ChatGPT or similar AI tools, you have to attach the full conversation to make it clear what help you obtained from them.* **Otherwise it will be considered cheating.** We reserve the right to deduct points for using such material beyond reason. **You must write up your solution independently, and type your answers word by word on your own: close the book/notes/online resources when you are writing your final answers.**

Write-up. Please use LaTeX to prepare your solutions. For all problems, **please explain how you get the answer** instead of directly giving the final answer, except for some special cases (will be specified in the problem). For all algorithm design problems, please describe using **natural language**. You could present pseudocode if you think that helps to illustrate your idea. Please do not only give some code without explanation. **In grading, we will reward not only correctness but also clarity and simplicity. To avoid losing points for hard-to-understand solutions, you should make sure your solutions are either self-explanatory or contains good explanations.** See some more details here `https://www.cs.ucr.edu/~ygu/teaching/218/W24/assignments/index.html`

Programming Problems. You will need to submit your code on CodeForces (a readme file about submitting code is available on the course webpage). You also need to submit a short report through GradeScope along with your solutions of the written assignments. In the report, you need to **specify your submission id**, describe the algorithm you designed, and show cost analysis if necessary. Note that your code will be automatically saved by CodeForces, so you do not need to submit the code again. For each problem, there will be 10-20 test cases. **For the training programming problems, you need to finish before 11:59pm, Wed 03/06, 2024.** With reasonable implementation, using C++ or Java is guaranteed to be able to pass all tests. You can use other languages, but it's not guaranteed that the implementations can be within the time limit.

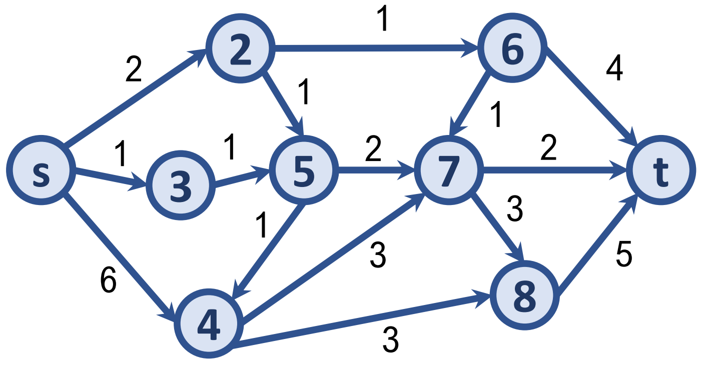
[1] Some of the problems are adapted from existing problems from online sources. Thanks to the original authors.

1 (1.5 pt) Union-Find

Prove that union-by-height (union-by-rank) can also guarantee that the longest length in the data structure is $O(\log n)$. Union-by-height means that, when merging two linked-trees in the linked forest, we always attach the shallower one to the deeper one.

Hint: in class we've already talked bout union-by-size, you can come up with a similar proof about union-by-height.

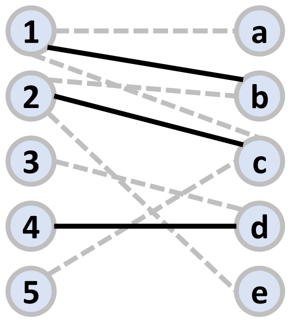
2 (1.5 pt) Network Flow



Let's consider using Ford-Fulkerson Method to find the max flow for this graph.

1. (0.3pts) In this graph, find an augmenting path. What is the flow you can increment based on this path?
2. (0.6pts) Draw the residual graph after you process the augmenting path.
3. (0.6pts) Use Ford-Fulkerson Method to find the max flow of this graph. Show your work.

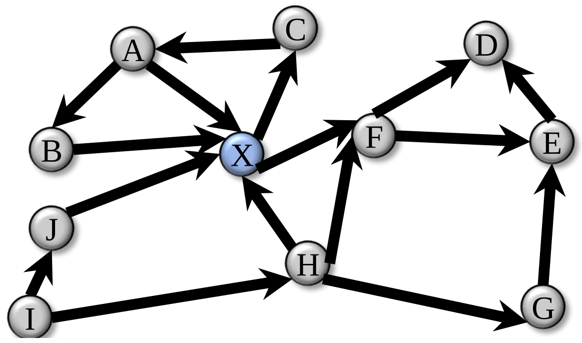
3 (0.8 pt) Bipartite Graph Matching



Let's consider using Hungarian Method to find the maximum matching for this graph. In the graph on the left, we use solid lines to represent matched edges, and dotted lines to represent edges in the graph, but not chosen in the match. For both problems, you only need to show the answer.

1. (0.4pts) Based on the graph and the current matching, find an augmenting path.
2. (0.4pts) Show the match after you process this augmenting path.

4 (1.2 pt) Strongly Connected Components



You only need to show the answer for the following questions.

1. (0.2 pt)What is a strongly connected component (SCC) in a graph?
2. (0.3 pt) Which vertices are in the same SCC of vertex X? **Answer:** ________________________________.
3. (0.5 pt) If we run reachability search on vertex X, and partition the vertices in four parts as in the BGSS algorithm, what is the partition of the graph? **Answer:** ________________________________.
4. (0.2 pt) How many SCCs are there in this graph? **Answer:** ________________________________.

5 Programming Assignments

Training and challenge programming problems are available on codeforces. In this assignment we provided four challenge problems, and you are expected to solve at least one of them to get the full grade of this assignment. You can still complete more as your bonus points.