

PROBLEM SET 5

DATA STRUCTURES AND ALGORITHMS 2 - FALL 2023

cfr5spw

DUE DECEMBER 5, 2023 AT 11:59 PM

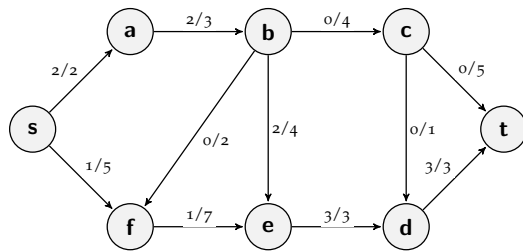
Collaboration Policy: You are encouraged to collaborate with up to 4 other students, but all work submitted must be your own *independently* written solution. List the computing ids of all of your collaborators in the `collabs` command at the top of the tex file. Do not share written notes, documents (including Google docs, Overleaf docs, discussion notes, PDFs), or code. Do not seek published or online solutions for any assignments. If you use any published or online resources (which may not include solutions) when completing this assignment, be sure to cite them. Do not submit a solution that you are unable to explain orally to a member of the course staff. Any solutions that share similar text/code will be considered in breach of this policy. Please refer to the syllabus for a complete description of the collaboration policy.

Collaborators: n/a

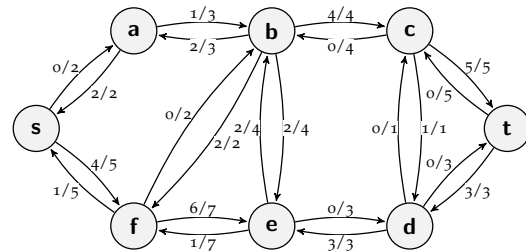
Sources: <https://www.geeksforgeeks.org/maximum-bipartite-matching/#>, Intro to Algorithms, Cormen 4th ed (section 23)

PROBLEM 1 Max Flow

Given the following Flow Network G and Residual Graph G' :



Flow Network G



Residual Graph G'

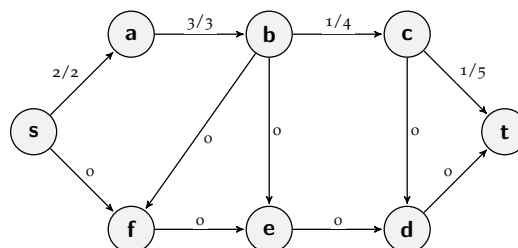
- Find an augmenting path in the graph G' . List the nodes in the path you found in order (e.g., $s \rightarrow a \rightarrow b \rightarrow c \rightarrow d \rightarrow t$).

Solution:

$s \rightarrow a \rightarrow b \rightarrow c \rightarrow t$

- Update the Flow Network G above. You **must** edit the graph below (do not upload a picture).

Solution:



- Find the min cut of the graph. List the nodes on each side of the cut.

Solution:

Set S: s, a, b, c

Set T: f, e, d, t

PROBLEM 2 *Element Uniqueness Reduction*

Reduce Element Uniqueness to Mode in $O(n)$ time. Element Uniqueness is defined as: given a list of numbers, return true if no number appears more than once (i.e., every number is distinct). Mode is defined as: given a list of numbers, return one of the numbers which appears most frequently in the list; i.e. if everything was unique it will return an arbitrary element.

Solution:

we can start by creating a hash table and so we will Initialize an empty hash table, H , that will store each element of the input list as keys and their frequencies as values.

then we'll process the List i.e. Iterate through the list of numbers.

For each number:

Check if the number is already a key in H .

If it is not, insert the number into H with a frequency of 1.

If it is, increment the frequency count of that number.

In order to check for Uniqueness we will process the list and while processing, if we ever increment the frequency count above 1, we immediately know that the Element Uniqueness condition fails (i.e., a number appears more than once).

If we complete the iteration without finding a duplicate then we have found the mode since, we can return any element from the list as the mode since all elements are unique and so all have the same frequency, 1.

This reduction works in $O(n)$ time because hash table ops (insert and check) are $O(1)$ on average and we iterate through the list only once, performing a constant number of operations per element.

PROBLEM 3 *Reading and Evaluating Proofs*

Generative AI systems are exciting – *and scary*. They can answer many questions, but how much can we trust the results?

For this problem you will choose an generative AI system (e.g., ChatGPT, Bing (inside of Edge)) and ask it to do a proof for an algorithm we've studied in this unit—specifically, the proof that the reduction of Bi-Partite Matching to Max-Flow is correct. You'll then carefully read the proof it gives you and compare it to the version of that proof in our textbook, noting any issues or significant differences.

Here's a suggested prompt to give the system. You may use this unchanged, or alter it to try to get a better result.

Answer this question as if you were a computer scientist. Formally prove that the Bi-Partite Matching algorithm using a max-flow algorithm is correct (i.e, it always find the optimal matching between nodes in the bi-partite graph).

In your solution, provide the following;

1. Give the name and version number of the generative AI system you've used.

Solution:

chatgpt 3.5

2. In a sentence, describe the proof strategy used by the AI.

Solution:

The AI uses a broad overview approach to the problem, highlighting how the algorithm should show a valid matching and to prove that the matching has the max size possible.

3. Study the textbook's proof of this algorithm, Lemma 24.9 in Section 24.3 of the 4th edition of the textbook. In no more than 5 or 6 sentences, describe any issues or problems you see in the AI's result or how it differs from the textbook's proof. *Your answer might address the following questions: Do you think it successfully proves correctness? Are there gaps or odd logical jumps in the proof it provides? How different is it from the proof in the textbook? (If there are no issues to report, just say that.)*

Solution:

4. Copy the prompt you gave the AI below.

Solution:

Answer this question as if you were a computer scientist. Formally prove that the Bi-Partite Matching algorithm using a max-flow algorithm is correct (i.e, it always find the optimal matching between nodes in the bi-partite graph).

5. Copy the AI's response (the proof) below.

Solution:

To formally prove the correctness of the Bipartite Matching algorithm using a max-flow algorithm, we need to demonstrate two key properties:

The algorithm always finds a valid matching. The matching found by the algorithm is optimal, i.e., it has the maximum possible size. We will prove these properties step by step.

1. Valid Matching;

A Bipartite Matching is a set of edges in a bipartite graph such that no two edges share a common endpoint. The algorithm uses a max-flow approach to find this matching. Here's how the algorithm works:

Create a source node 's' and a sink node 't'. Add edges from 's' to all nodes in the left partition of the bipartite graph (call it A). Add edges from all nodes in the right partition of the bipartite graph (call it B) to 't'. Assign capacities of 1 to all these edges. Add edges between nodes in A and B based on the desired matching. Find a max-flow from 's' to 't' in the augmented flow network. The algorithm ensures that each node in A is matched with at most one node in B. This is because the capacity of the edge from 's' to each node in A is 1, so the max-flow cannot push more than one unit of flow through any such edge. Similarly, each node in B is matched with at most one node in A.

Thus, the algorithm always finds a valid matching since it ensures that no two edges share a common endpoint.

2. Optimal Matching:

To prove that the matching found by the algorithm is optimal, we'll use the Max-Flow Min-Cut theorem:

Max-Flow Min-Cut Theorem: In any flow network, the value of the maximum flow is equal to the capacity of the minimum cut.

In our case, the flow network consists of the bipartite graph with capacities on edges, and the minimum cut separates the source 's' from the sink 't'. The capacity of this minimum cut is equal to the maximum possible matching in the bipartite graph.

Since the algorithm finds a max-flow from 's' to 't' in this flow network, the value of the maximum flow is indeed equal to the capacity of the minimum cut, which is the maximum possible matching. Therefore, the matching found by the algorithm is optimal.

In conclusion, we have formally proved that the Bipartite Matching algorithm using a max-flow algorithm always finds a valid matching and that the matching it finds is optimal.

PROBLEM 4 *Algorithms and Society: Ethical and Social Issues*

In society the term *algorithms* is frequently used for what some call *algorithmic decision systems*. These are systems that rely on large amounts of data and algorithms that use AI or machine learning to make decisions in a wide range of important issues in society.

https://en.wikipedia.org/wiki/Automated_decision-making

While these systems are different in nature that almost all the algorithms topics we've studied in this course, some kind of algorithmic process is at the heart of such systems. It is appropriate for a computing student studying algorithms to be aware of this use of the term and to understand examples of such systems and the social and ethical challenges they pose.

Below is a list of articles, etc., that touch on algorithm-based systems and social or ethical issues in society. Choose one article that interests you, and answer the following questions.

1. List the title of the of the reading that you chose.

Solution:

Machine Bias

2. In no more than five or six sentences, summarize how one algorithmic decision system discussed in the reading may lead to negative or undesirable consequences for individuals or societies. (The reading may discuss more than one, but you only need to write about one.)

Solution:

One of the algorithmic decision systems discussed in the article is risk assessment software and how it is used in the criminal justice system. This software predicts the likelihood of individuals committing crimes and is found to be biased against black defendants. The article uses the example of Brisha Borden, a black teenager who was rated as high risk for future crimes after only a minor theft while a white man named Vernon Prater with a history of serious crimes was rated low risk. In the end software's predictions were incorrect Borden did not reoffend, while Prater committed another crime. This example highlights the system's tendency to falsely label black defendants as high risk at almost twice the rate as white defendants, potentially leading to harsher sentencing in the criminal justice system.

3. In a few sentences, what actions do you think the computing personnel or organizations that create such systems could do to reduce possible negative or undesirable consequences? (Keep your answers brief!)

Solution:

To reduce negative consequences of algorithmic decision systems, when designing these systems we should ensure diversity in the teams developing these systems, incorporating perspectives from different racial and socio-economic backgrounds to decrease the chance of bias as much as possible. It is also important to regularly update these algorithms to identify and correct biases found and ensure they are transparent with the public and addressed accordingly. Going further we should also collaborate with experts in ethics, sociology, and criminal justice to understand and address the broader societal impacts of these systems, since as software engineers we should not be held responsible for deciding what ethics to implement in our systems. This can also be done through testing phases that specifically look for disparate impacts on different demographic groups and also open dialogue with the communities impacted by these technologies to understand their concerns and incorporate their feedback.

Readings (choose one):

1. *Bias in AI-based models for medical applications: challenges and mitigation strategies*
<https://www.nature.com/articles/s41746-023-00858-z>
2. *Machine Bias* (This is a longer article on risk assessments in criminal sentencing that got a lot of national attention.)
<https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing>
3. *Why colleges are using algorithms to determine financial aid levels*
<https://www.highereddive.com/news/colleges-enrollment-algorithms-aid-students/692601>
4. *Designing Ethical Self-Driving Cars*
<https://hai.stanford.edu/news/designing-ethical-self-driving-cars>
5. *Algorithms that Run the World, an interview with Cathy O'Neil* (author of the book "Weapons of Math Destruction")
<https://thedecisionlab.com/podcasts/algorithms-that-run-the-world-with-cathy-oneil>