CSE 421 Section 2

Graph Traversal and Algorithm Proofs

Administrivia

Announcements & Reminders

- **HW1**
  - Was due yesterday, 10/2
  - Remember the **late problems** policy (NOT assignments)
    - Total of up to **10 late problem days**
    - At most **2 late days per problem**

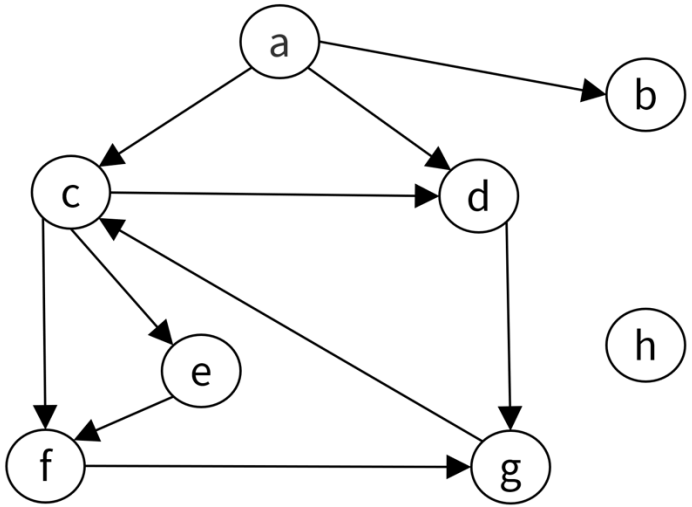
- **HW2**
  - Due Wednesday 10/11 @ 11:59pm
  - Don’t wait to start

- **Pseudocode handout**
  - Read it at home

Warmup: BFS/DFS

Problem 1 – BFS/DFS review

a) Run BFS and record the layer of each vertex, starting with a in layer 0.
b) Run DFS, record the start/end times, and classify the edges. When there are multiple choices for the next vertex, pick the alphabetically earliest one.



Feel free to work with the people around you!

Proving algorithms correct

Problem 2 – Investigating algorithm proofs

The purpose of this problem is to help you:

- Figure out how to **start a proof** about algorithms.
- **Check the correctness** of proofs.

We will take a proof that you have already seen in lecture and critically analyze it to check correctness.

Problem 2 – Investigating algorithm proofs

Dive right in, and we'll summarize the takeaways afterwards.

a) Answer the questions embedded in the proof in your packet as you read. They are marked with ▷ and italics.
b) Discuss with people near you:
    - What is the general structure of a proof that an algorithm is correct?
    - How is the proof related to the pseudocode?

Feel free to work with
the people around you!

Problem 2 – Investigating algorithm proofs

A **loop invariant** is a property that is true before, during, and after the loop.

- They are very useful!
- Formally prove with induction, but you can omit proof on HW/exams if trivial.

Examples:

- If $v \in R$, then there exists a path from s to v.
- $s \in R$.

Non-examples:

- If there exists a path from s to v, then $v \in R$.
  - Only true after loop end, so need to use loop exit condition to prove.

Problem 2 – Investigating algorithm proofs

The structure of an algorithm proof

- Validity:
  - Make sure that every line is possible, may require ad-hoc methods.
- Termination (if you have while-loops):
  - Find a **measure of progress** that increases/decreases every iteration, but is **bounded** by the problem setup.
- Correctness:
  - Start with “We will show that the output matches the desired result,” then unwrap definitions.
  - Combine **loop invariants** and the **loop exit condition** to prove results.
    - Prove invariants by induction, if non-obvious.

Problem 2 – Investigating algorithm proofs

Validity was obvious in our case, but not all algorithms.

```
Initialize each person to be free
while (some p in P is free) {
    Choose some free p in P
    r = 1st person on p's preference list to whom p has not yet proposed
    if (r is free)
        tentatively match (p,r)    //p and r both engaged, no longer free
    else if (r prefers p to current tentative match p')
        replace (p',r) by (p,r)    //p now engaged, p' now free
    else
        r rejects p
}
```

existence not obvious!

Problem 2 – Investigating algorithm proofs

How to solve HW problems

1. Think of an algorithm (we'll give more tips for this next section)
2. Prove its **validity**, **termination**, and **correctness** as in the last slide.
    - Write the same level of detail as in sections/lectures.
3. Just for yourself, **expand the details like we did today** to check that it's correct.
    - If you can't, your TAs probably can't either (and will mark your proof as wrong or incomplete).
        - Go back to step 1 or step 2 and try again.

Remember, **if your algorithm is wrong, it's impossible to prove it to be correct!**

Applications of graph algorithms

Tips for algorithms with graphs

Ask yourself the following questions:

1. What are the vertices?
2. What are the edges?
3. What am I looking for in the graph?

Problem 3 – Judging books

You have a large collection of books and want to arrange them by color. You wish to put only books of a single color on any given shelf. Every pair of books is either “same color” or “not same color”, and this relation is an equivalence relation (reflexive, symmetric, and transitive).

Input: A list of books, and a list of pairs that are the same color
Expected output: The best upper bound on the number of shelves you will need

Example input: books u, v, w, x, and pairs (u, v), (v, w)
Output: 2

Problem 3 – Judging books

Input: A list of books, and a list of pairs that are the same color
Expected output: The best upper bound on the number of shelves you will need

1. What are the vertices?
2. What are the edges?
3. What am I looking for in the graph?

Feel free to work with the people around you.

Problem 3 – Judging books

You have a large collection of books and want to arrange them by color. You wish to put only books of a single color on any given shelf. Every pair of books is either “same color” or “not same color”, and this relation is an equivalence relation (reflexive, symmetric, and transitive).

Input: A list of books, and a list of pairs that are the same color
Expected output: The best upper bound on the number of shelves you will need

Now write the algorithm (two sentences) and think about the proof.
Feel free to work with the people around you.

Problem 4 – Water jugs

You have a 5-gallon jug and 3-gallon jug, which start out empty. Your goal is to have **4 gallons of water in the 5-gallon jug** and **0 gallons of water in the 3-gallon jug**. Unfortunately, you are only allowed the following operations:

- Fill any of your jugs completely.
- Pour one of your jugs into the other, until the first jug is empty or the second is full.
- Empty out all the water in a jug.

a) Describe a method to reach the goal. (No need to use any general algorithm yet, just solve the puzzle however you like.)

Feel free to work with the people around you.

Problem 4 – Water jugs

b) Solve with a graph algorithm and state the running time (no proof during section):

Input: Jug sizes a and b, with target amounts x and y, respectively.
Expected output: The minimum number of steps to reach the target amount, or "unreachable".

1. What are the vertices?

2. What are the edges?

3. What am I looking for in the graph?

Feel free to work with the people around you.

Last note on applying graph algorithms

In section today, we saw two examples of **directly calling a graph algorithm**, as if it were a library function.

Sometimes, in other problems, you might need to **reimplement a graph algorithm** and tweak a line or two to solve the problem.

Summary

- For algorithms, prove **validity, termination, and correctness**.
  - Make sure that you can expand all details in your head.
  - Find a **measure of progress** to prove termination.
  - Start with “We will show that the output matches the desired result,” then expand definitions, use **observations that are true in every iteration**, and use the **loop exit condition** to prove correctness.
- Model problems using graphs, then apply or tweak algorithms like BFS/DFS/etc.

Thanks for coming to section this week!