COMS 3110: Homework 5
Due: December 5^{th}, 11:59pm
Total Points: 50

Late submission policy. An assignment that is submitted one day late will get a penalty of 10%. An assignment that is submitted two days late will get a penalty of 20%. Assignments submitted after two days will not be graded and will get no points. For example, if an assignment is due on Friday by midnight, a submission on Saturday will be penalized by 10%, and a submission on Sunday will be penalized by 20%. A submission on Monday will get no points.

Submission format. Homework solutions must be **handwritten**. You may either write directly on a tablet or write on paper and scan your work. In both cases, the final submission must be compiled into a **single PDF file**. Typed solutions will only be accepted with prior, explicit permission from the instructor. We reserve the right **NOT** to grade homework that does not follow these requirements. Name your submission file: `<Your-net-id>-3110-hw5.pdf`. For instance, if your netid is `asterix`, then your submission file will be named `asterix-3110-hw5.pdf`. Each student must hand in their own assignment. If you discussed the homework or solutions with others, a list of collaborators must be included with each submission. Each of the collaborators has to write the solutions in their own words (copies are not allowed). Also, recall the AI policy:

- You may **not use AI tools to generate complete solutions** to assignments, coding problems, or any assessed work that is intended to measure your independent knowledge and skills.
- Whenever you use AI tools to assist in your assignment, you must **include a paragraph at the end of the assignment explaining what you used the AI for and include all the prompts you used to get the results. Failure to do so is in violation of the academic honesty policies**.

General Requirements

- When proofs are required, do your best to make them both clear and rigorous. Even when proofs are not required, you should justify your answers and explain your work.
- When asked to present a construction, you should show the correctness of the construction.

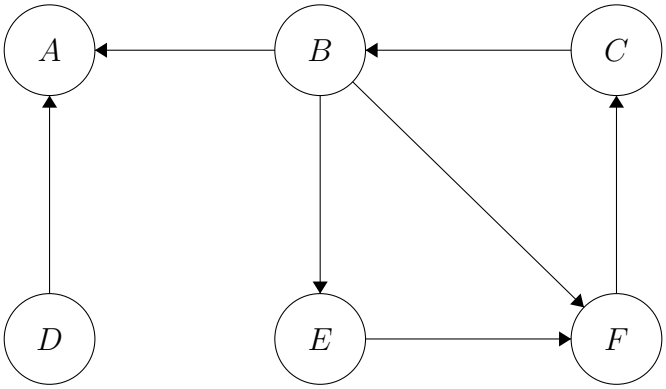
Expectations for Runtime Analysis in Algorithm Problems When a problem explicitly asks for one or more of the following components, these are the expectations for a correct solution.

- **Provide runtime:** You are only required to state the final runtime of the algorithm, and full credit will be awarded if your answer is correct. However, if your answer is incorrect, no partial credit will be given unless you also provide a derivation.
- **Provide runtime and derivation:** Show step-by-step reasoning, including the number of iterations of loops and the runtime of each line of pseudocode, leading to the total runtime.
- **Provide exact number of iterations:** Specify the precise number of times each loop executes for the given input size.
- **Provide upper bound on the number of iterations:** State a guaranteed maximum number of iterations for each loop; this may be larger than the exact count. This is **not asymptotic Big-O bound**.
- **Provide asymptotic bound on the number of iterations:** Describe the growth rate of number of iterations using big-O notation.

Some Useful Equalities

- $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
- $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$
- $2^{\log_2 n} = n$, $a^{\log_b n} = n^{\log_b a}$, $n^{n/2} \leq n! \leq n^n$, $\log x^a = a \log x$
- $\log(a \times b) = \log a + \log b$, $\log(a/b) = \log a - \log b$
- $a + ar + ar^2 + ... + ar^{n-1} = \frac{a(r^n-1)}{r-1}$
- $1 + \frac{1}{2} + \frac{1}{2^2} + ... + \frac{1}{2^n} = 2(1 - \frac{1}{2^{n+1}})$
- $1 + 2 + 4 + ... + 2^n = 2^{n+1} - 1$

1. **(6 pts)**

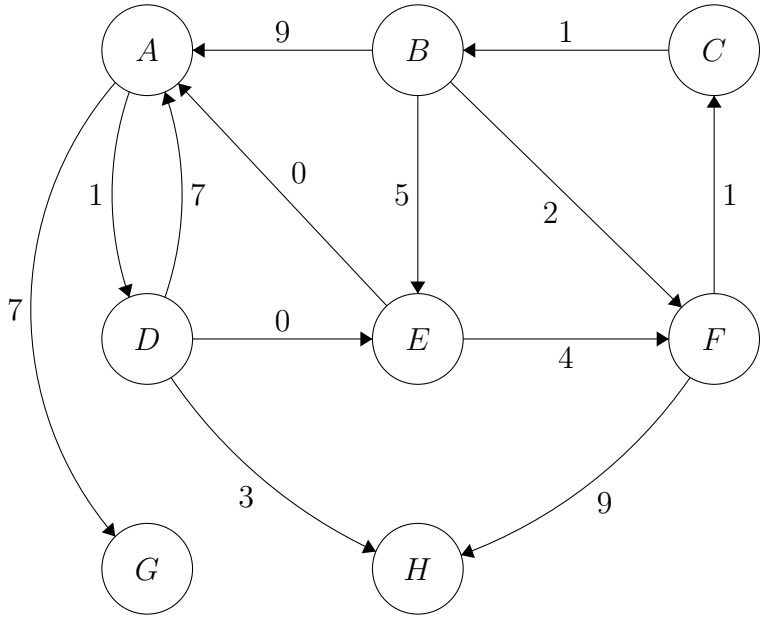


Execute a DFS on the preceding graph. Assume that all vertices are processed in alphabetical order.

(a) **(3 pts)** During the DFS, vertices are assigned a start and finish time using a single counter that is incremented after each use. The counter's initial value is 1. For each vertex, list it's start and finish time.

(b) **(3 pts)** For each edge in the graph, indicate whether it was a *tree, back, forward,* or *cross edge* in the aforementioned DFS.

2. **(8 pts)**



Recall Dijkstra's algorithm using a binary heap as presented in class. Consider the following weighted graph and write the valuation of d-value for each vertex at the end of each iteration of while-loop (that iterates until the priority queue is empty) in the algorithm. The source vertex is C.

(*Add or remove rows from the table, if needed, for your solution.*)

Iteration	A	B	C	D	E	F	G	H
Initial d-value	∞	∞	0	∞	∞	∞	∞	∞
1								
2								
3								
4								
5								
6								
7								
8								

3. **(10 pts)** An power company has built a power plant in a region that has yet been without electricity. They intend to run power lines along the roads to every home in this region using as little line as possible. You're responsible for planning where the electrical lines should be installed. You've modeled the region with the power plant and each home as a unique vertex, roads between them as edges, and edge weights as the length of the road.

   (a) **(1 pts)** State which graph problem is equivalent to solving this task.

   (b) **(1 pts)** Identify an efficient algorithm for computing a solution to this task and state the runtime.

(c) **(6 pts)** Shortly before undertaking the task of building out the planned power lines, one of the roads that was going to be used became impassible due to a bridge collapsing. The homes on either end of the road are still reachable via other roads, however. Give an efficient algorithm the uses the graph of the region you've made, the previously computed solution to the problem, and the now-impassible road to compute an updated solution to the problem. State the runtime of your algorithm. This algorithm should be more efficient than the solution to part (b).

(d) **(2 pts)** Argue that your algorithm in part (c) produces a correct solution.

4. **(14 pts)** Solve the following recurrence relations using the recurrence tree method. For each one, you must show the entire table, the equality you derive from the table, the final solution as a function of n, and conclude the asymptotic bound.

   (a) **(7 pts)**

$$T(n) = \begin{cases} 3T(\frac{n}{5}) + c_1 n & \text{if } n > 1 \\ c_2 & \text{if } n \leq 1 \end{cases}$$

(b) **(7 pts)**

$$T(n) = \begin{cases} T(\frac{n}{3}) + \log_3 n & \text{if } n \geq 10 \\ 1 & \text{if } n < 10 \end{cases}$$

5. **(12 pts)** You are in a rectangular maze organized in the form of $M \times N$ cells/locations. You are starting at the upper left corner (grid location: $(1, 1)$) and you want to go to the lower right corner (grid location: (M, N)). From any location, you can move either to the right, down, or diagonally.

   So, from cell (i, j) you can move to cell $(i, j + 1)$, cell $(i + 1, j)$, or to cell $(i + 1, j + 1)$. Cost of moving right or down is 2, while the cost of moving diagonally is 4.

   Every cell on the grid, including location $(1, 1)$, contains diamonds with a value between 1 and 10. If you visit a cell, you earn an amount that is equal to the value of the diamond in the cell. Your objective is to go from the start corner to the destination corner. Your profit along a path is the total value of the diamonds you picked minus the sum of the all the costs incurred along the path. Your goal is to find a path that maximizes the profit.

   (a) **(6 pts)** Write a recurrence *opt* where $opt(i, j)$ is the maximum profit obtainable at grid location (i, j).

(b) **(6 pts)** Based on your recurrence, write an iterative dynamic programming algorithm that returns the maximum possible profit. The algorithm must take as input a 2d array representing the maze where each entry is the value of the diamond at that location. Provide the runtime of your algorithm.