Final Exam

Graded

Student

Nick Zhu

Total Points

64.5 / 100 pts

Question 1

MCQ — **12** / 20 pts

1.1 **MCQ 1** — **4** / 4 pts

✔ **– 0 pts** Correct, the answer is 1.8

1.2 **MCQ 2** — **4** / 4 pts

✔ **– 0 pts** Correct, the answer is |E|(|V|+|E|) = Big omega(|V|)

1.3 **MCQ 3** — **4** / 4 pts

✔ **– 0 pts** Correct, the answer is decreasing order of finish time

1.4 **MCQ 4** — **0** / 4 pts

✔ **– 4 pts** Incorrect, the answer is increasing the weight of el to wh

1.5 **MCQ 5** — **0** / 4 pts

✔ **– 4 pts** Incorrect, the answer is no path

Question 2

Short Answers — **21** / 30 pts

2.1 **Short Ans 1** — **6** / 6 pts

✔ **– 0 pts** Correct

2.2 **Short Ans 2** — **6** / 6 pts

✔ **– 0 pts** Correct

2.3 **Short Ans 3** — **3** / 6 pts

✔ **– 1.5 pts** Wrong for the first element

✔ **– 1.5 pts** Wrong for the second element

2.4 **Short Ans 4** — **6** / 6 pts

✔ **– 0 pts** Correct

2.5 **Short Ans 5** — **0** / 6 pts

✔ **– 6 pts** Wrong

Question 3

Finding Stuff **18** / 20 pts

3.1 **3(a)** **3** / 3 pts

✔ **– 0 pts** Correct

3.2 **3(b)** **3** / 5 pts

✔ **– 2 pts** No/insufficient explanation

3.3 **3(c)** **12** / 12 pts

✔ **– 0 pts** Correct

Question 4

Doing Stuff **13.5** / 30 pts

4.1 **4(a)** **4** / 4 pts

✔ **– 0 pts** Correct

4.2 **4(b)** **0** / 3 pts

✔ **– 3 pts** Wrong

topological sort takes O(n + m) exceeding O(n)

4.3 **4(c)** **7.5** / 15 pts

✔ **– 7.5 pts** Partially wrong

BFS does not work. Consider the following case: A->B->C, A->C. BFS starts from A, then B and C would both be chosen but C cannot run simultaneously with B.

4.4 **4(d)** **2** / 8 pts

✔ **– 6 pts** Completely wrong

Start time and finish time should be considered. E.g, a tree with B, C depends on A, and D, E depends on C. If AB>AC, you need to run 3 tasks in parallel, but if AB<BC, you only need 2.

Basic Algorithms (Section 5) Final Exam
Instructor: Jiaxin Guan

May 8, 2025

1. Write your name and net ID below (and on the top of each page if you detach pages).
2. You have 110 minutes to complete this exam.
3. You may use two double-sided, letter-size "cheatsheets". The use of phones, computers, or other reference material during the exam is not permitted.
4. You may use any algorithm or theorem we saw in class (or homework) without proof, as long as you state it correctly. For all questions asking you to give algorithms, you do not have to give detailed pseudo-code, or even any pseudo-code. It is enough to give a clear description of your algorithm.
5. Use the provided blue books to do any scratch work. The only work that will be graded is that written in the provided solution space on the exam. Both the blue books and finals will be collected at the end of the exam.
6. Read each question before solving any and start with the problems you are most confident about solving.
7. This exam contains 14 pages (including this cover page). Good luck.

Question	Points	Score
Multiple Choices	20	
Short Answers	30	
Finding Stuff	20	
Doing Stuff	30	
Total:	100	

Write your name:	Write your Net ID (NYU email):
Nick Zhu	xz4687

1 Multiple Choices (20 points)

Choose the best answer for each of the questions below. No justification is needed.

1. Consider the following list of letters and their corresponding frequencies.

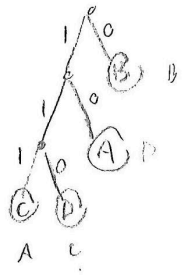
Letter	A	B	C	D
Frequency	0.2	0.5	0.1	0.2

$0.2\times2+0.5\times1+0.1\times3+0.2\times3$
$=0.4+0.5+0.3+0.6=1.8$

What is the average bits per letter (ABL) of the corresponding Huffman encoding for these symbols? ..... (C)

(A) 2.0
(B) 1.7
(C) 1.8
(D) 1.9

B 0 0.5
D 10 0.2
A 11 0.2
C 110 0.1



B: 0 1 0.5
A: 10 2 0.2
D: 110 3 0.2
C: 111 3 0.1

2. Let $G=(V,E)$ be **an acyclic, undirected graph**. Which of the following asymptotic statements is **false** with regard to this graph? ..... (A)

(A) $|E|\cdot(|V|+|E|)=\Omega(|V|)$ ✓ undirected.
(B) $|E|^2/|V|=o(|V|\log|V|)$ ✓ $|E|\le|V|-1$
(C) $|E|^2|V|=O(|V|^3)$
(D) $|V|\log|V|+|E|\log|E|=\Theta(|V|\log|V|)$ $|E|=0$?

3. After running DFS on a directed, acyclic graph (DAG), we can get the topological ordering of the vertices by sorting them according to ..... (B)

(A) Increasing order of discovery time
(B) Decreasing order of finish time
(C) Increasing order of finish time
(D) Decreasing order of discovery time

4. Let $G = (V, E)$ be a connected, undirected graph, and let T be a minimum spanning tree (MST) of G. Let e_h denote the heaviest edge in T, with weight w_h, and let e_l denote the lightest edge in T, with weight w_l.

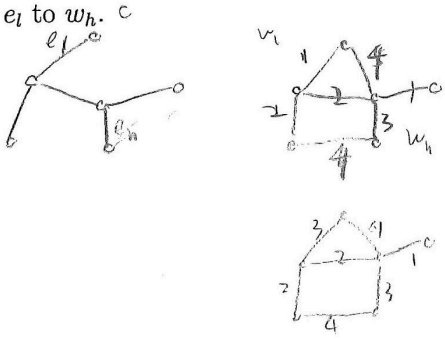
Under which of the following modifications to the graph is T **not** guaranteed to remain a MST? ..... (A)

(A) Adding a new edge (one not originally in G) of weight w_h.

(B) Removing an edge that is not in T.

(C) Decreasing the weight of e_h to w_l.

(D) Increasing the weight of e_l to w_h.



5. After running the Floyd-Warshall algorithm on a graph $G = (V, E)$, we have that $DP(u, v, |V|) = \infty$ for some vertices u and v. What does this indicate? ..... (C)

(A) There is no path from vertex u to vertex v.

(B) The shortest path from vertex u to vertex v has not been computed yet.

(C) There is a negative weight cycle in the graph.

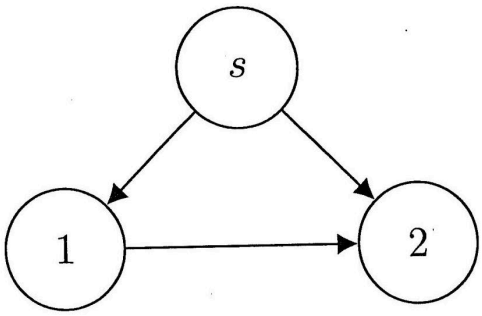
(D) There is a negative edge on the shortest path from vertex u to vertex v.

2 Short Answers (30 points)

Give a short answer (a sentence or two) to each of the questions below. No justification is needed for full credit. However, in the case of a wrong answer, partial credits might be awarded if some progress is shown.

1. The following figure shows a directed acyclic graph (DAG) with a designated source vertex labeled s. All the other vertices are **labeled with the number of distinct simple paths** from s to that vertex. Recall that a simple path is a path that does not revisit any vertex.

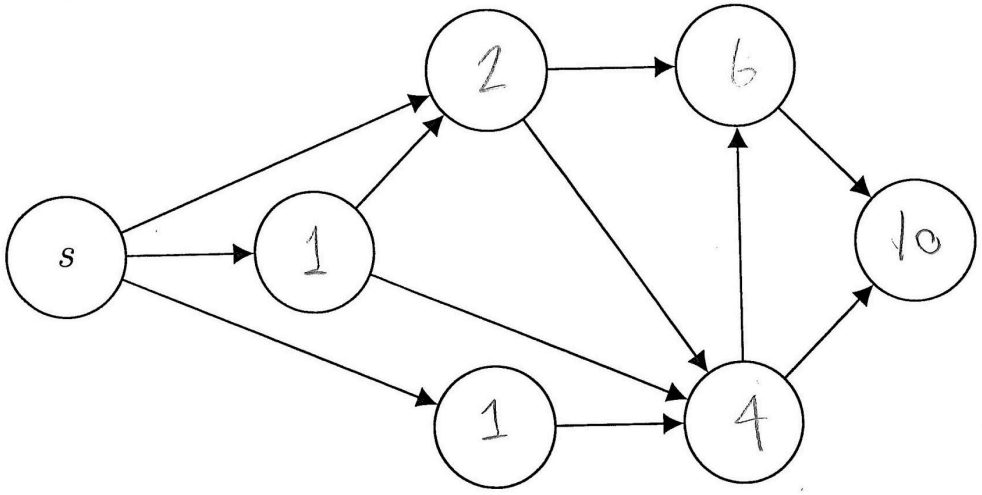
   Example:



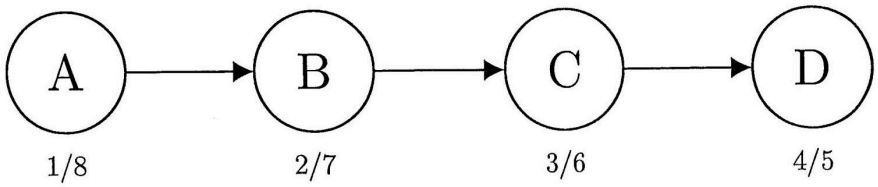
   In the above example:

   - The bottom left vertex has 1 simple path from s,
   - The bottom right vertex has 2 simple paths from s.

   Now complete the graph below by labeling each vertex with the number of simple paths from s:

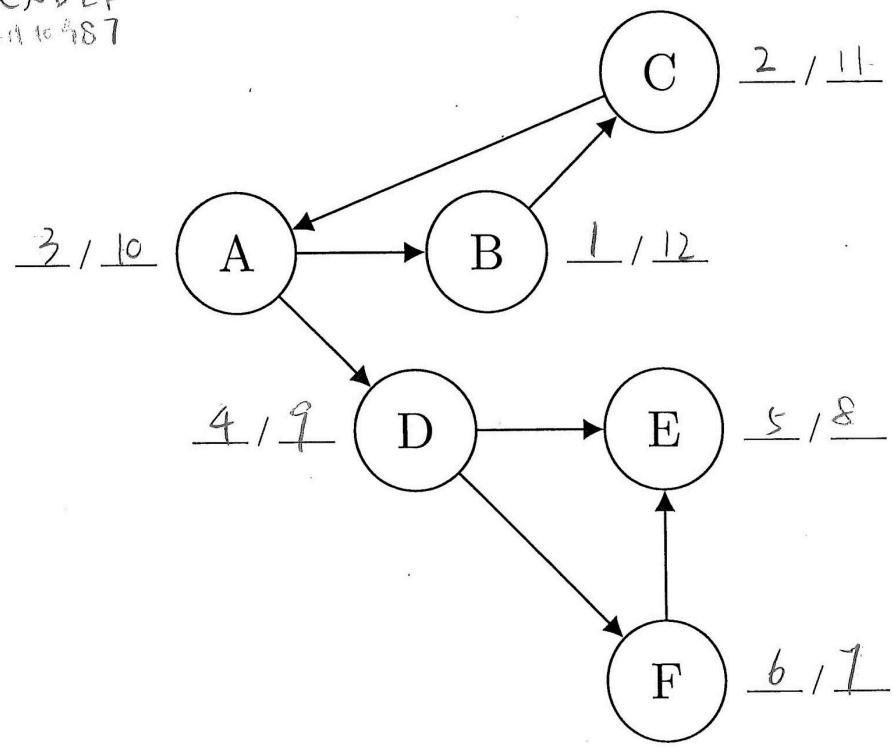


2. We can represent a DFS run by labeling the discovery time and finish time for each vertex. For each vertex, we attach a label "d/f" where d is the discovery time, and f is the finish time for the vertex. See example below:



Using this convention, illustrate a DFS run on the following graph where **the edge AB will be classified as a <u>backward</u> edge.**

BCADEF
12 11 10 9 8 7



3. Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of points on the real line. Your task is to find the smallest set P of intervals of length 10 (i.e., each interval is of the form $[s, s+10]$ for some real number s) such that every point in X lies within at least one interval in P (i.e. P "covers" all of the points in X).

Exactly one of the following two statements is incorrect:

(A) There exists an optimal set P that contains an interval starting at the smallest uncovered point x_i.

(B) There exists an optimal set P that contains an interval covering the maximum number of currently uncovered points.

The incorrect statement is: B.

Provide a counterexample to the incorrect strategy by filling in the missing two values below:

$$X = \{\underline{19}, 20, 24, 26, 29, \underline{30}\}$$

4. Suppose we have a magical implementation of a priority queue that supports Push, ExtractMin, and DecreaseKey operations in constant time (i.e., $O(1)$ time per operation). What is the tightest possible asymptotic bound on the worst-case runtime of Dijkstra's algorithm on a graph $G = (V, E)$ using this magical priority queue?

Answer: $O(|V|+|E|)$

$|V| + |V| \times T(\text{Push}) + |V| \times T(\text{ExtMin}) + |E| \cdot T(\text{DecKey})$

$= |V| + |V| + |V| + |E| = 3|V| + |E| = O(|V|+|E|)$

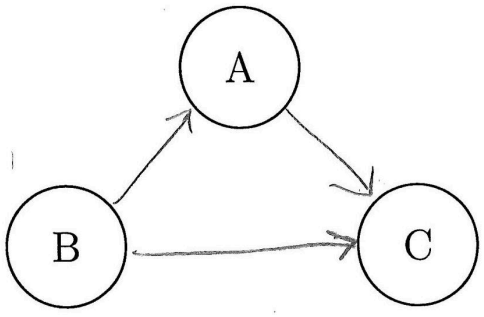
worst case: $|E| = |V|(|V|-1)$ for directed graphs

$= |V|^2 - |V|$

5. In the Kosaraju–Sharir algorithm for computing strongly connected components (SCCs) of a directed graph G, the second step involves running DFS on the transposed graph G^T (i.e., G with all edge directions reversed). The DFS is performed by iterating through the vertices in **decreasing** order of their finish times from the first DFS on G.

   Your friend Steve proposes an alternative idea: instead of constructing G^T, why not skip the transpose step and simply run a second DFS directly on G, this time iterating through the vertices in **increasing** order of their finish times?

   Provide a counterexample with three vertices to show that Steve's proposal does not correctly compute the strongly connected components. (*Hint: All you need to do is to add 3 edges to the graph below.*)



3 Finding Stuff (20 points)

Your friend hands you an $n \times n$ matrix A with the following properties:

- Each row is strictly increasing: for all $i \in \{1, 2, \ldots, n\}$, we have $A[i, 1] < A[i, 2] < \cdots < A[i, n]$.
- Each column is strictly increasing: for all $j \in \{1, 2, \ldots, n\}$, we have $A[1, j] < A[2, j] < \cdots < A[n, j]$.

(a) (3 pts) The following 3×3 matrix must contain each number from the set $\{1, 2, \ldots, 9\}$ exactly once, and satisfy the properties described above: each row and each column must be strictly increasing. Fill in the missing entries to complete the matrix.

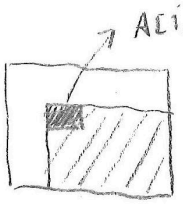
	$j=1$	$j=2$	$j=3$
$i=1$	1	3	4
$i=2$	2	5	8
$i=3$	6	7	9

(b) (5 pts) Suppose you are searching for a specific value x in matrix A, and you examine the entry $A[i, j]$ only to find that $A[i, j] > x$. Which part(s) of the matrix can you conclusively rule out as **not** containing x? Briefly explain.

(c) (12 pts) Give an efficient algorithm that determines whether a value x is in the matrix A. For full credit, your algorithm should have a worst case runtime of $O(n^2 \log n)$. Briefly justify the correctness and runtime of your proposed algorithm. You may assume that n is a power of 2 and all values in A are unique, if that helps.

$= O(n^2 \log n^2) \Rightarrow O(\log n)$
$(O((\log n^2))$

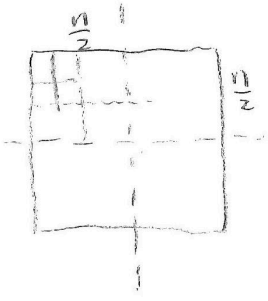
b)



→ the shaded part can be conclusively rule out as not containing x.

i.e A[x,y] with $x \geq i$ and $y \geq j$

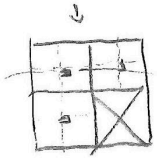
Use this page if you need additional space for your solution to Problem 3.



consider split the matrix in to 4 sub-matrix with equivalent size.

Start by checking if $A[\frac{n}{2}+1, \frac{n}{2}+1] > x$

1° $A[\frac{n}{2}+1, \frac{n}{2}+1] > x$, the we can ~~etu~~ rule out the sub-matrix on the bottom-right corner.



this is the most complicated case when the move on to check

$A[\frac{n}{4}+1, \frac{n}{4}+1]$, $A[\frac{1}{4}n+1, \frac{n}{4}+1]$

$A[\frac{n}{4}+1, \frac{3}{4}n+1]$

2° $A[\frac{n}{2}+1, \frac{n}{2}+1] = x$
we found it!

this limit the possible condition and we can further narrow where the x can be.

3° $A[\frac{n}{2}+1, \frac{n}{2}+1] < x$

we move on to check the bottom-right corner matrix on the

correctness: this search gives an optimal structure, and ~~go~~ include all the possible conditions

time: we ~~dont~~ use divide and conquer

$T = O(\log n)$

Use this page if you need additional space for your solution to Problem 3.

4 Doing Stuff (30 points)

You are given a list of n tasks $\{x_1, x_2, \ldots, x_n\}$. Each task x_i has:

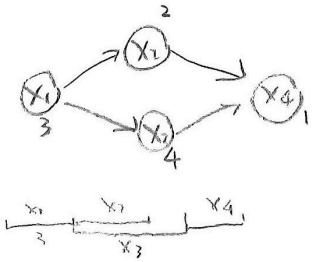
- A duration $t_i > 0$ representing the time it takes to complete the task once started, and
- A set of dependencies — other tasks that must be completed before x_i can begin.

It is guaranteed that there are no circular dependencies.

Your goal is to minimize the total amount of time required to finish all the tasks.

(a) (4 pts) Consider the following task set:

Task	t_i	Dependencies
x_1	3	–
x_2	2	x_1
x_3	4	x_1
x_4	1	x_2, x_3



Give the minimum total time needed to complete all tasks in each of the following cases:

(i) You can perform only one task at a time.

(ii) You can perform any number of tasks at the same time. Note that this does not shorten the time t_i needed for each individual task x_i.

Answers:

(i) 10 (ii) 8

(b) (3 pts) Show how to compute the minimum total time required to complete all tasks in $O(n)$ time **if only one task can be executed at a time.** No justification is needed.

(c) (15 pts) Describe an efficient algorithm to compute the minimum total time required to complete all tasks **when you are allowed to run an unlimited number of tasks in parallel.** For full credit, your algorithm should have a worst case runtime of $O(n+m)$, where m is the total number of dependencies. Briefly justify the correctness and runtime of your proposed algorithm.

(d) (8 pts) Now that you've figured out the minimum amount of time needed using your algorithm in part (c), you also want to make sure you don't exhaust yourself. So you want to find out what is the maximum number of tasks that you need to handle simultaneously at any point in time in order to achieve that minimum total time. Describe an efficient algorithm to compute this value, with a worst case runtime of $O((n+m)\log(n+m))$. Briefly justify the runtime of your proposed algorithm. No justification of correctness is needed.

b) we just simulate the whole process going through every vertices in the topo order and then sum them up.

c) we first compute the topological order of these n tasks, using time of $O(n+m)$,

→ then we run BFS over to the topo. order graph, also using time of $O(n+m)$

→ after this, we get a BFS tree, we can then simulate the whole process starting from root s.

→ for each level of the BFS tree, we take the max t_i after finishing exploring this tree, we can sum up the max t_i at each level, and we get the minimum total time.

d) consider SCC
using the idea of

Use this page if you need additional space for your solution to Problem 4.

Use this page if you need additional space for your solution to Problem 4.