Midterm 1

Name: Ran Liao

SID: 3034304227

Name and SID of student to your left:

3034298427 Yuan Changchang

Name and SID of student to your right:

Exam Room: ☐ 2060 VLSB ☐ 145 Dwinelle ■ 10 Evans ☐ Hearst Field Annex A1
☐ 100 GPB ☐ 120 Latimer ☐ 2050 VLSB
☐ 405 Soda (6-10 pm) ☐ 306 Soda (6-10 pm)

Please color the checkbox completely. Do not just tick or cross the box.

Rules and Guidelines

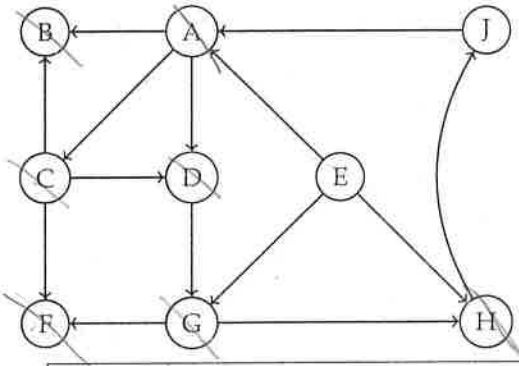
- **The exam is out of 110 points and will last 110 minutes.**
- Answer all questions. Read them carefully first. Not all parts of a problem are weighted equally.
- Write your student ID number in the indicated area on each page.
- Be precise and concise. **Write in the solution box provided.** You may use the blank page on the back for scratch work, but it will not be graded. Box numerical final answers.
- The problems may **not** necessarily follow the order of increasing difficulty. ***Avoid getting stuck on a problem.***
- Any algorithm covered in lecture can be used as a blackbox. Algorithms from homework need to be accompanied by a proof or justification as specified in the problem.
- Throughout this exam (both in the questions and in your answers), we will use ω_n to denote the first n^{th} root of unity, i.e., $\omega_n = e^{2\pi i/n}$.
- You may assume that comparison of integers or real numbers, and addition, subtraction, multiplication and division of integers or real or complex numbers, require $O(1)$ time.
- Good luck!

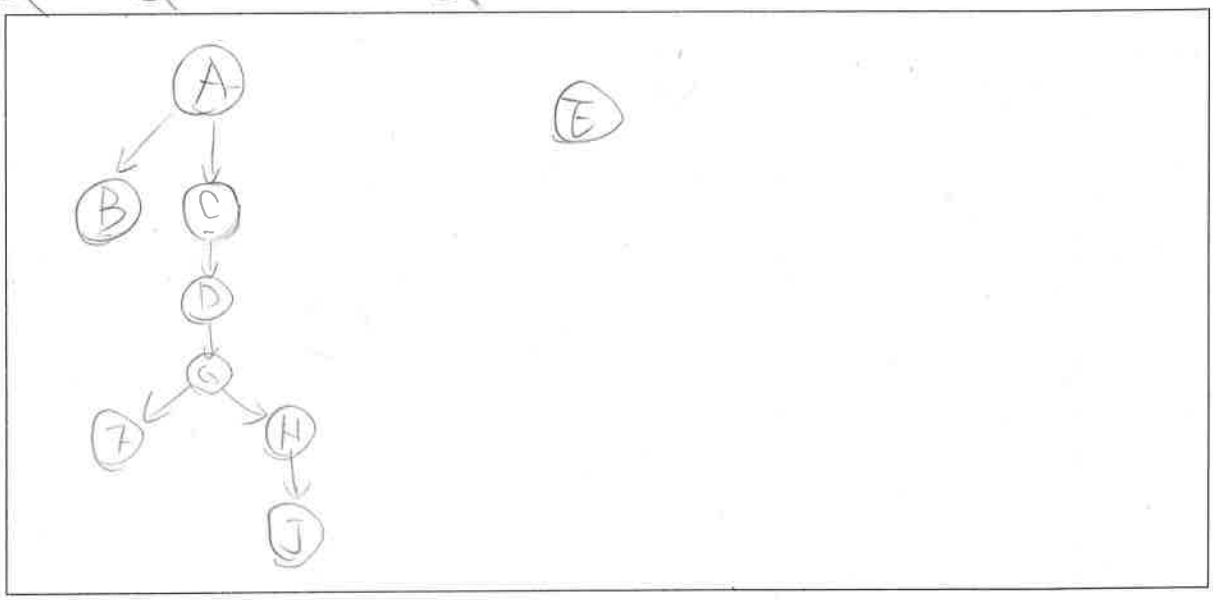
Discussion Section

Which of these do you consider to be your primary discussion section(s)? Feel free to choose multiple, or to select the last option if you do not attend a section. **Please color the checkbox completely. Do not just tick or cross the boxes.**

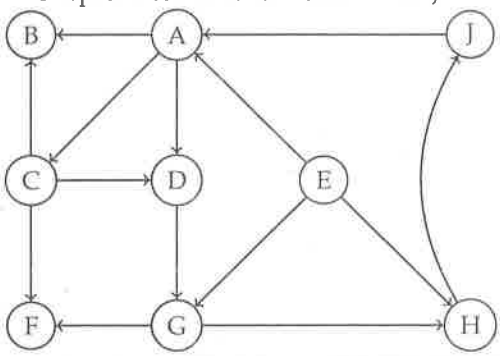
- ☐ Antares, Tuesday 5 - 6 pm, Mulford 240
- ☐ Kush, Tuesday 5 - 6 pm, Wheeler 224
- ☐ Arpita, Wednesday 9 - 10 am, Evans 3
- ☐ Dee, Wednesday 9 - 10 am, Wheeler 200
- ☐ Gillian, Wednesday 9 - 10 am, Wheeler 220
- ☐ Jiazheng, Wednesday 11 - 12 am, Cory 241
- ☐ Sean, Wednesday 11 - 12 am, Wurster 101
- ☐ Tarun, Wednesday 12 - 1 pm, Soda 310
- ■ Jerry, Wednesday 12 - 1 pm, Wurster 101
- ☐ Jierui, Wednesday 12 - 1 pm, Etcheverry 3113
- ☐ Max, Wednesday 12 - 1 pm, Etcheverry 3115
- ☐ James, Wednesday 2 - 4 pm, Dwinelle 79
- ☐ David, Wednesday 2 - 3 pm, Barrows 140
- ☐ Vinay, Wednesday 2 - 3 pm, Wheeler 120
- ☐ Julia, Wednesday 3 - 4 pm, Wheeler 24
- ☐ Nate , Wednesday 3 - 4 pm, Evans 9
- ☐ Vishnu, Wednesday 3 - 4 pm, Moffitt 106
- ☐ Ajay, Wednesday 4 - 5 pm, Hearst Mining 310
- ☐ Zheng, Wednesday 5 - 6 pm, Wheeler 200
- ☐ Neha, Thursday 11 - 12 am, Barrows 140
- ☐ Fotis, Thursday 12 - 1 pm, Dwinelle 259
- ☐ Yeshwanth, Thursday 1 - 2 pm, Soda 310
- ☐ Matthew, Thursday 1 - 2 pm, Dwinelle 283
- ☐ Don't attend Section.

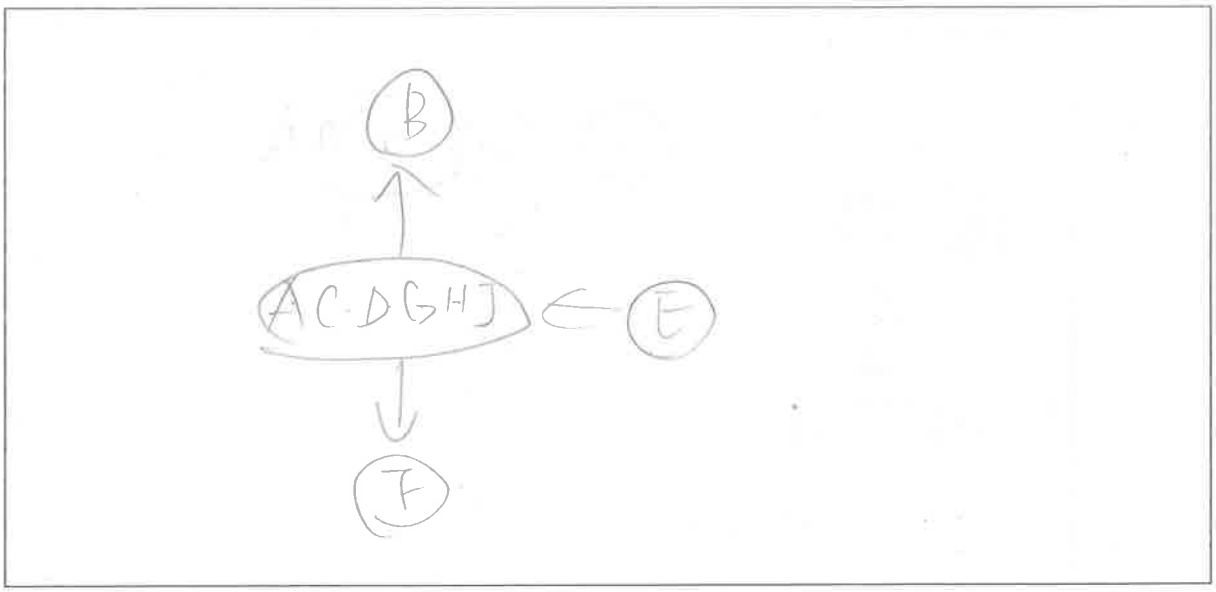
1. **(6 points)** Execute a DFS traversal on the graph shown below starting at node A, breaking ties alphabetically. **Draw the DFS Tree/Forest in the box below.**





2. (**4 points**) Draw the DAG of strongly connected components for the above graph (the directed graph is reproduced here for convenience)





$2^n > 2^{\sqrt{n}} > n^2$

3. **(12 points)** For each of these pairs of functions $(f(n), g(n))$, identify whether $f(n) = O(g(n))$ or $g(n) = \Omega(f(n))$ or $f(n) = \Theta(g(n))$. Pick the most specific of the three options in each case, and pick at most one option in each case.

$f(n)$	$g(n)$	$f(n) = O(g(n))$	$g(n) = \Omega(f(n))$	$f(n) = \Theta(g(n))$
n^2	$(n^2 + \log n)(n+5)$	○	○	●
3^n	$2^n \cdot n^4$	○	●	○
$2^{\sqrt{n}}$	$n^{\log n}$	○	●	○
$n \log n$	$n^{1.01}$	●	○	○

$\log n < n^{0.01}$

$n = (\log y)^2$

$\frac{2^{\sqrt{n}}}{n^{\log n}} = y$

$\frac{y}{(\log y)^{4 \log(\log y)}}$

4. **(12 points)** Write down the solutions to the following recurrence relations (only the final answer needed). Assume $T(0) = T(1) = 1$ and write down the most specific bound that you can derive.

(a) $T(n) = 8T(n/2) + O(n^2)$

$O(n^3)$

$\log_2 8 = 3 > 2$

(b) $T(n) = 4 \cdot T(n-2)$

$O(2^n)$

$T(n) = 4T(n-2)$ $\quad T(n-2) = 4T(n-4)$

$= 4^2 T(n-4) +$

$= 4^k T(n-2k) = 4^{\frac{n}{2}}$

$n - 2k = 0$

$2k = n$

$= 2^n$

$k = \frac{n}{2}$

(c) $T(n) = T(n - n^{1/3}) + 1$

$O(3^n)$

$T(n - n^{\frac{1}{3}}) = T(n - n^{\frac{1}{3}} - \sqrt[3]{n - n^{\frac{1}{3}}})$

$\log n \quad n^c \quad 2^n$

$\lim_{n \to \infty} n - n^{\frac{1}{3}} = 0$

$n^{\frac{1}{3}}(n^{\frac{2}{3}} - 1) = 1$

$x^3 - x - 1 = 0$

$(x-1)(x^2 + 2x + 1) = 0$

$x^3 + 2x^2 - x^2$

$81 \sim$

$27 \sim 24 \sim$

5. **(8 points)** Given a directed graph $G = (V, E)$, for a vertex v define $numVisited[v]$ as,

$numVisited[v]$ = Number of nodes including v that are visited if we start exploring the graph at node v. Equivalently, number of nodes visited by $explore(v)$ if we start DFS from vertex v with $visited[w]$ initialized to *False* for all vertices w

In a directed graph G, the values of $numVisited[v]$ are as follows:

Vertex	A	B	C	D	E	F	G	H	I	J	K
numVisited	5	5	6	5	6	2	6	6	2	10	10

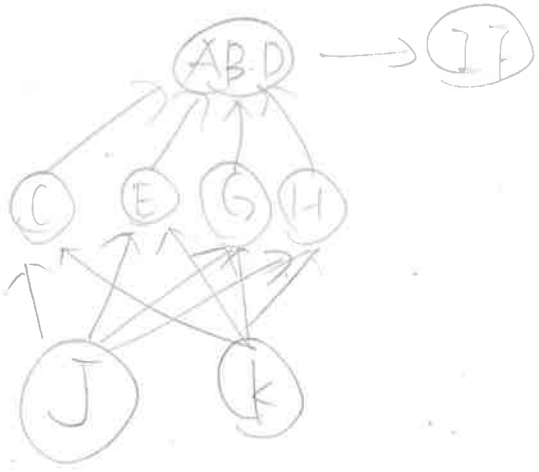
(a) Draw all the sink SCCs

{I, J}

(b) Draw all the source SCCs

{J}, {F}

(c) Draw the DAG of SCCs



u v

6. (**4 points**) Vertex v appears immediately after u in a linearization of a DAG, but then there is no edge $u \to v$. Then what can you say about the following statements?

   (a) There is a path from u to v.

   ○ Always True ○ Sometimes True ● Never True

   (b) There is a path from v to u.

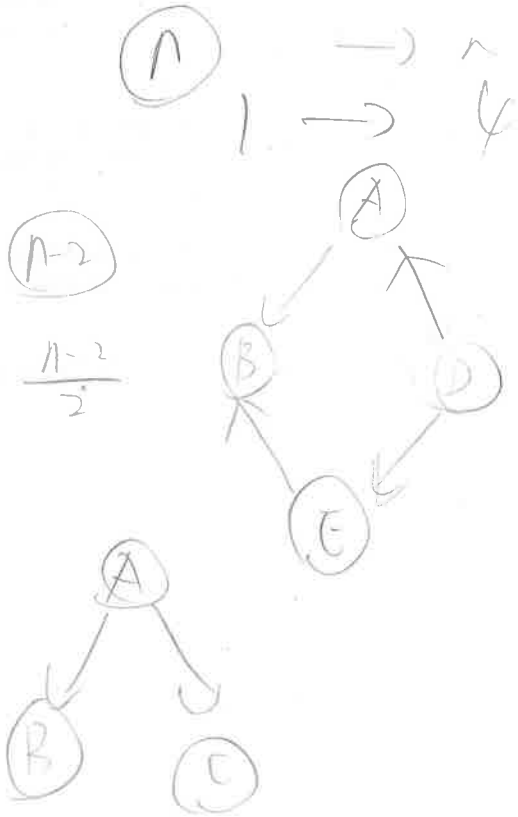
   ○ Always True ○ Sometimes True ● Never True

7. (**6 points**) In each of these cases, write the tightest bound possible.

   (a) In an undirected graph on n vertices, deleting an edge can increase the number of connected components by at most [1]

   (b) In a directed acyclic graph on n vertices, deleting an edge can increase the number of strongly connected components by at most [0]

   (c) In a directed graph on n vertices, deleting an edge can increase the number of strongly connected components by at most [n − 1]



8. **(16 points)**

For each of the statements below indicate whether they are TRUE or FALSE:

(a) In a particular execution of DFS on a directed graph G, it so happened that for every vertex v, the corresponding call to $explore(v)$ did not visit any new node but terminated immediately. Then G must have n strongly connected components.

● True ○ False

(b) All the cycles of a graph $G = (V, E)$ can be listed in time $O(|V| \cdot (|V| + |E|))$ by executing the DFS traversal $|V|$ times, once from each vertex.

● True ○ False

(c) Recall the recursive $Select(S, k)$ algorithm for selecting the k^{th} smallest element in a list of numbers S. This algorithm has asymptotically the same expected runtime for finding the smallest element ($k = 1$) or the median ($k = n/2$), but asymptotically different worst case runtimes.

○ True ● False

(d) Let $(v_1, \ldots, v_n)$ denote a linearized ordering of the vertices of a Directed Acyclic Graph (DAG) G. If we add an edge $v_n \to v_1$ then the resulting graph can still be a DAG.

● True ○ False

(e) 8^{th} roots of unity are a subset of 16^{th} roots of unity.

● True ○ False

(f) Fix n to be a power of 2. FFT can be used to evaluate a degree n polynomial at the $2n^{th}$ roots of unity in time $O(n \log n)$.

● True ○ False

(g) Fix n to be a power of 2. FFT can be used to evaluate a degree $2n$ polynomial at the n^{th} roots of unity in time $O(n \log n)$.

○ True ● False

(h) Adding a new edge to the graph always increases the number of DecreaseKey operations in Dijkstra's algorithm.

○ True ● False

(i) Strassen's matrix multiplication is a sick algorithm.

○ True ● False

9. **(5 points)** Let n be a power of 2 and let $\{1, \omega, \omega^2, \ldots, \omega^{n-1}\}$ denote the n^{th} roots of unity. Suppose

$$E[0, \ldots, n/2-1] \leftarrow FFT(a_0, a_2, \ldots, a_n)$$

and

$$O[0, \ldots, n/2-1] \leftarrow FFT(a_1, a_3, \ldots, a_{n-1})$$

then write pseudocode to compute $FFT(a_0, \ldots, a_n)$ given arrays E and O.

```
For i = 0 to n-1
    FFT(a0,a1 ... an)[i] = E[i % n/2] + w^i O[i % n/2]
```

10. **Testing efficiently (8 points)**

At most k people among a population of n people in a city have been afflicted by a deadly virus. The city council needs to identify the k people infected so as to treat them.

The infection can be detected by testing the patient's blood for the virus. Unfortunately, each test is very expensive and the city council would like to minimize the number of tests carried out.

One can mix the blood samples from a subset of people, and test at one shot if at least one in the subset is infected. Specifically, for any subset S of people, $test(S : subset)$ carries out one blood test and returns YES if at least one in S is infected and NO otherwise.

Describe an algorithm that uses $O(k \log n)$ tests to algorithm to find all infected people in the city. (try to use 6 sentences or less)

In each iteration, split all people into k equal-size subgroups. And test each of those subgroups. If return No, remove this subgroup. If return Yes, invoke this algorithm on that subgroup recursively.

11. **Moving Battalions**

You are given an undirected unweighted graph $G = (V, E)$ that represents a network of cities. There are two army battalions that are positioned at two designated vertices $s_1, s_2 \in V$.

The battalions move during the day and rest at night. Each day a battalion can move to a neighboring city, or stay where it is. The two battalions would like to reach their target destinations $t_1, t_2 \in V$ respectively.

No city can host both the battalions for a night, so the two battalions can never spend the same night in the same city. Construct a graph H such that we can recover the optimal plan by running BFS on H.

The number of vertices and edges in H should be polynomial in number of vertices in G, i.e., $|V(G)|^c$ for some fixed constant c.

(a) **(7 points)** Vertices of H:

For $\forall$ vertex $v, u \in V$. $(v \neq u)$
construct new vertex $V_{v,u}$ in H

(b) **(8 points)** Edges of H:

If there's a edge from $u \to v$
construct edge from $V_{u,k}$ to $V_{v,k}$ for and
$V_{k,u}$ to $V_{k,v}$ for $\forall k \in V$.
(make sure $k \neq u$, $k \neq v$)

$(x^{c_1} + x^{c_2} + \cdots x^{c_t})^k$.

12. **Adding Coins**

In the nation of Transformia, there are t different kinds of coins with integer denominations $c_1, \ldots, c_t$ in the range $[1, \ldots, n]$ in circulation.

We will now devise an algorithm that takes as input the values $c_1, \ldots, c_t$ in the range $[1, \ldots n]$ and an integer k, and lists all possible values that can be expressed as a sum of exactly k coins. Here the k coins expressing a value need not be distinct.

For example, if $\{c_1, c_2\} = \{1, 7\}$ and $k = 2$, then the set of values that can be expressed are $\{1+1, 1+7, 7+7\} = \{2, 8, 14\}$.

(a) **(6 points)** Find a polynomial which can be used to obtain all possible values that can be expressed as a sum of exactly k coins.

$(x^{c_1} + x^{c_2} + \cdots x^{c_t})^k$

(b) **(2 points)** We will now design an algorithm to compute the polynomial using FFT. What is the order of FFT we would use for this purpose?

$n \cdot k$

nk

(c) **(5 points)** Describe an algorithm that computes the polynomial as efficiently as you can. You can use *FFT, InverseFFT and Polynomial multiplication via FFT* in a blackbox fashion in your description. (Try to use 6 sentences or less)

Let the expression be $(x^{c_1} + x^{c_2} + \cdots + x^{c_t})^{\frac{k}{2}} (x^{c_1} + x^{c_2} + \cdots + x^{c_t})^{\frac{k}{2}}$

$\Downarrow$ A_1 $\Downarrow$ A_2

Compute A_1 and A_2 recursively and then use Polynomial multiplication via FFT to compute $A_1 A_2$

(d) **(1 point)** What is the runtime of your algorithm?

$nk[\log(nk)]^2$

$nk = x$

$F(n,k) = 2F\left(\frac{n}{2}, \frac{k}{2}\right) O(nk \log nk)$

$F(x) = 2F\left(\frac{x}{2}\right) + x \log x$

$F\left(\frac{x}{2}\right) = 2F\left(\frac{x}{2}\right) + \frac{x}{2} \log \frac{x}{2}$

$F(x) = 2^k F\left(\frac{x}{2^k}\right) + x \log x + x \log \frac{x}{2} + \cdots + \log \frac{x}{2}$

$= x\left(\log\left(\frac{x}{2^0} \cdot \frac{x}{2} \cdot \frac{x}{4} \cdots \frac{x}{2^{\log x}}\right)\right)$

$x \log \frac{x^{\log x}}{2^{\frac{\log x(\log x+1)}{2}}}$ $\quad O + \log x \frac{(\log x+1)}{2}$

13. **Testing efficiently again**(Extra Credit)

The city council from Question 10 is now dealing with a new virus. The good news is that the new virus has only infected exactly two citizens in the population of n people. The bad news is that the test for the new virus fails if we mix the blood of both the infected parties. Formally, the *test*(S : *subset*) for a subset of citizens S returns *YES* if S contains exactly one infected citizen, but returns *NO* if S contains zero or two infected citizens.

Describe an algorithm to identify the two infected citizens using at most $O(\log n)$ tests. (try to use 6 sentences or less)

Split citizens into 2 subgroups and test each subgroup respectively. If both tests return Yes, there's one infected citizen in each subgroups. We just need to keep spliting them until identify the infected citizen. If both tests return No, invoke this algorithm recursively on both of those subgroups.