Midterm 1

Name:

SID:

Name and SID of student to your left:

Name and SID of student to your right:

Exam Room:

☐ 10 Evans ☐ 1 Pimentel ☐ 2050 VLSB ☐ Wheeler Auditorium ☐ 540AB Cory ☐ Other

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

Rules and Guidelines

- **The exam is out of 85 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.
- Any algorithm covered in the lecture can be used as a blackbox.
- 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., $\omega_n = e^{2\pi i/n}$. So ω_{16} will denote the first 16^{th} root of unity, i.e., $\omega_{16} = e^{2\pi i/16}$.
- 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.**

☐ Fred Zhang Wheeler 202 • Tu 5-6pm

☐ Jialin Li Etcheverry 3105 • Tu 5-6pm

☐ Tiffany Chien Etcheverry 3109 • W 9-10am

☐ Sidhanth Mohanty Wheeler 130 • W 9-10am

☐ Teddy Tran Moffitt Library 106 • W 10-11am

☐ Emaan Hariri Hearst Field Annex B5 • W 12-1pm

☐ Dee Guo Wheeler 224 • W 12-1pm

☐ Arpita Singhal Dwinelle 234 • W 1-2pm

☐ Gillian Chu Barrows 136 • W 1-2pm

☐ Rachel De Jaen Barrows 155 • W 1-2pm

☐ Joshua Turcotti Wheeler 20 • W 2-3pm

☐ Varun Jhunjhunwalla Wheeler 202 • W 2-3pm

☐ Vishnu Iyer Etcheverry 3113 (Advanced Section) • W 2-3pm

☐ Avni Singhal Wheeler 30 • W 3-4pm

☐ Jiazheng Zhao Dwinelle 229 • W 3-4pm

☐ Rishi Veerapaneni Wheeler 224 • W 3-4pm

☐ Jeff Xu Evans 9 • W 3-4pm

☐ Noah Kingdon Wheeler 202 • W 5-6pm

☐ Neha Kunjal Hildebrand B51 • Th 11-12pm

☐ Christina Jin Evans 9 • Th 12-1pm

☐ Noah Krakoff Moffitt Library 103 • Th 1-2pm

☐ Ajay Raj Wheeler 202 • Th 2-3pm

1 Comparing asymptotics (4 points)

For each question, fill in all circles that apply.

		$f = O(g)$?	$g = O(f)$?
$f(n) = 4^n$	$g(n) = 16^{\log_2(n)}$	○	○
$f(n) = (\sqrt{n} + n)(30\sqrt{n})$	$g(n) = n^2$	○	○
$f(n) = n^3$	$g(n) = n^{\log_3(26)}$	○	○
$f(n) = n^{0.001}$	$g(n) = \log_2 n$	○	○

2 True or False? (5 points)

Mark your choice for each of the following. Fill in the bubble completely, as incomplete markings will not be given credit.
Grading: $+1$ for each correct answer, 0 for leaving blank, and -1 for each incorrect answer. Any negative score will affect your entire exam.

(a) (**1 point**) A DAG does not necessarily have a unique topological ordering.

◯ True ◯ False

(b) (**1 point**) On graphs with negative edge weights, Dijkstra does not work since it does not necessarily halt; otherwise, once it halts, it outputs the correct solution.

◯ True ◯ False

(c) (**1 point**) If DFS on a directed graph $G = (V, E)$ produces exactly one back edge, then it is always possible to remove an edge e from the graph G such that $G' = (V, E - \{e\})$ is a DAG.

◯ True ◯ False

(d) (**1 point**) If the directed graph G contains a cycle, and removing an edge from G can make it acyclic, then any DFS on G would produce exactly one back-edge.

◯ True ◯ False

(e) (**1 point**) If DFS on a directed graph $G = (V, E)$ produces two back edges, there must be at least two strongly connected components which each have at least 2 vertices in the original graph

◯ True ◯ False

3 Recurrences (8 points)

Write down the solutions to the following recurrence relations (only the final answer is needed). Write down the tightest bound that you can derive. You may use big-O notation.

(a) **(2 points)** $T(n) = 23T(\frac{n}{3}) + 2n^3$

(b) **(2 points)** $T(n) = 3T\left(n^{\frac{1}{3}}\right) + 5n$, and $T(3) = 3$.

(c) **(2 points)** $T(n) = 8T(n-3) + 1$, and $T(0) = T(1) = T(2) = 1$.

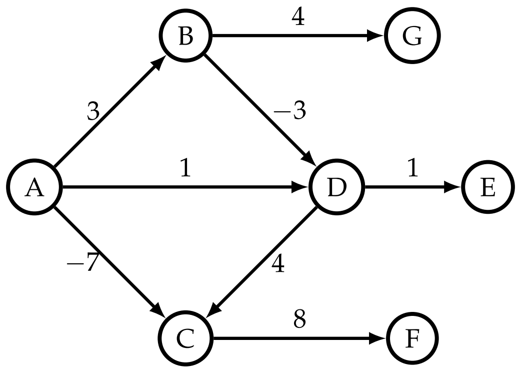
(d) **(2 points)** $T(n) = T(n/5) + T(4n/5) + 3n^2$

4 Dijkstra's (6 points)

Execute Dijkstra's algorithm on the following graph starting at vertex A and breaking ties alphabetically.
Here is the algorithm for reference. Assume that decreasekey does nothing if the vertex is not in the heap.

Algorithm 1 Dijkstras(G, l, s)

```
for all $u \in V$ do
    dist(u) = $\infty$
    prev(u) = null
end for
dist(s) = 0
$H$ = makequeue($V$) (using dist-values as keys)
while $H$ is not empty do
    u = deletemin($H$)
    for all edges $(u,v) \in E$ do
        if $\text{dist}(v) > \text{dist}(u) + l(u,v)$ then
            $\text{dist}(v) = \text{dist}(u) + l(u,v)$
            $prev(v) = u$
            decreasekey($H$, $v$)
        end if
    end for
end while
return dist
```



Fill in the following table with the shortest paths computed by Dijkstra's algorithm:

A	B	C	D	E	F	G
0						

5 Factorials (10 points)

In this question, assume you can multiply two d-bit integers in time $O(d^{1.59})$. Given an integer n, design an algorithm to compute $n!$ that runs in time $O(n^{1.59} \log^c n)$ for some constant $c > 0$. *Hint: divide and conquer.*

(a) Describe your algorithm **succinctly** below.

(b) Provide a rigorous analysis for the runtime of your algorithm. Recall you showed on homework that $\log(n!) = \Theta(n \log n)$.

6 Fast Force Computation (10 points)

5 charged particles are placed on a line. Particle i is placed at point i on the x-axis with charge c_i. The force on particle j in the system is

$$f_j = \sum_{i<j} \frac{c_i}{(i-j)^2} - \sum_{i>j} \frac{c_i}{(i-j)^2}.$$

(a) Set up two polynomials $p(x)$ and $q(x)$ so that $f_1, \ldots, f_5$ appear as coefficients of $p(x) \cdot q(x)$. (It is okay if $p(x) \cdot q(x)$ have other *irrelevant* coefficients as well.) Your answer should depend on $c_1, \ldots, c_5$.

$p(x) =$

$q(x) =$

(b) What is the index of the coefficient of $p(x) \cdot q(x)$ corresponding to f_3? (e.g. if it is the coefficient of x^2, write 2.)

7 Tree matches (20 points)

Let T be an unweighted and undirected tree on vertex set $V = \{1, \dots, 2n\}$. $E(T)$ is the set of T's edges, and $V(T)$ is the set of its vertices. We call a collection of n pairs $P = \{(u_1, v_1), \dots, (u_n, v_n)\}$ a *valid pairing* if for $i = 1, \dots, n$, $u_i \neq v_i$ and each vertex $v \in V$ occurs in exactly one of the n pairs. We define the *cost* of a valid pairing P as

$$f_T(P) := \sum_{i=1}^{n} d_T(u_i, v_i)$$

where $d_T(a, b)$ is the length of the shortest path between a and b in T. In this question we will see an $O(n)$ time algorithm to compute the minimum cost of a valid pairing; in particular, to compute

$$\mathsf{OPT}(T) := \min_{P \text{ valid pairing}} f_T(P).$$

We emphasize that $\mathsf{OPT}(T)$ is a *number* — namely the minimum value attained by f (We don't ask you to output the a pairing with minimum cost.)

(a) (**5 points**) Let $P^* = \{(u_1, v_1), \dots, (u_n, v_n)\}$ be a valid pairing such that $f_T(P^*) = \mathsf{OPT}(T)$ and let p_t denote the unique path between vertices u_t and v_t in T. Prove that for any $i, j \in \{1, \dots, n\}$ such that $i \neq j$, p_i and p_j are *edge-disjoint*. We say two paths p_i and p_j are *edge-disjoint* if the collection of edges used by p_i does not intersect the collection of edges used by p_j. You are encouraged to illustrate your proof idea in a diagram.
Hint: What if for some $i < j$, p_i and p_j shared an edge e? Can you make a small change to P^* to obtain a new pairing P' with $f(P') < f(P^*)$?

(b) (**5 points**) Let P^* and p_t be defined as in part (a) and let $L \subseteq E(T)$ be a subset of edges defined as follows:

$$L := \{e : \exists p_t \text{ such that } e \in p_t\}.$$

Deleting an edge $e \in E(T)$ splits T into two trees $T_{e,1}$ and $T_{e,2}$. Prove that $e \in L$ *if and only if* $|V(T_{e,1})|$ and $|V(T_{e,2})|$ are both odd numbers.
Try to use the result of part (a) to show that if $|V(T_{e,1})|$ and $|V(T_{e,2})|$ are both even, then $e \notin L$. You will additionally need to argue that if $|V(T_{e,1})|$ and $|V(T_{e,2})|$ are both odd, then $e \in L$.

(c) (**10 points**) Use the result of part (b) to devise an algorithm to compute $\mathsf{OPT}(T)$. Full points will be given for an $O(n)$-time algorithm.
Hint: Observe that $\mathsf{OPT}(T) = f(P^*) = |L|$ *where L is defined in part (b).*

8 Trickster negative cycles (22 points)

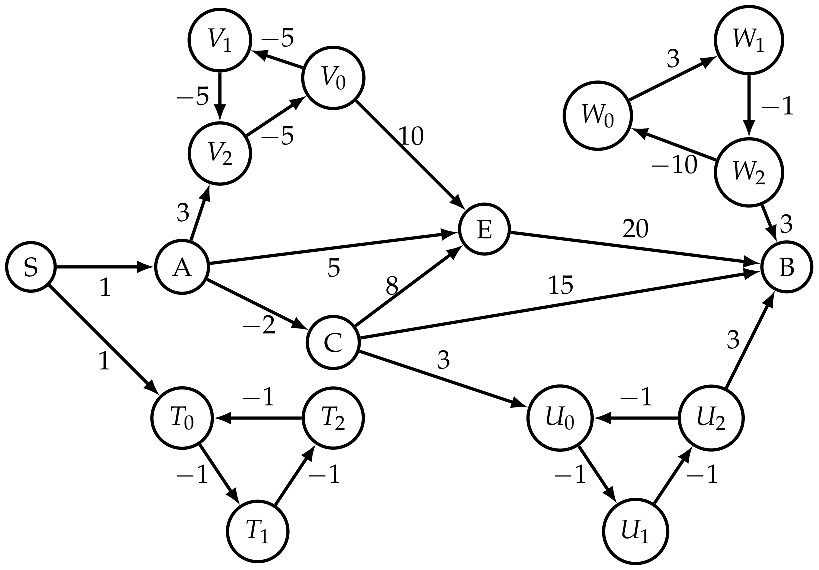
(a) (**4 points**) Draw a weighted directed graph G with the following properties:

(i) it contains a *negative-weight cycle*,

(ii) it has two vertices S and T such that the length of the shortest path between S and T is *positive*.

Make sure to explicitly write the weight of each directed edge in the graph you draw! The distance between S and T in your graph should be positive.

(b) (**8 points**) Consider the following graph:



The graph has exactly four negative cycles: $T_0T_1T_2$, (which we will call cycle T), $U_0U_1U_2$ (which we will call cycle U), $V_0V_1V_2$ (which we will call cycle V), $W_0W_1W_2$ (which we will call cycle W).

We say that the distance path between nodes X and Y is *not well-defined* if for any path from X to Y, you can find a strictly shorter path (e.g. there's a path of length 10 from X to Y, one of length 5 from X to Y, one of length 0, one of length -5, etc)

(i) Consider what happens if all negative cycles except one are removed from the graph. You would like the distance from S to **B** to be well-defined.

Mark all of the negative cycles for which keeping only that negative cycle results in the distance from S to B being well-defined.

○ T ○ U ○ V ○ W

(ii) Consider what happens if all negative cycles except one are removed from the graph. You would like the distance from S to **E** to be well-defined.

Mark all of the negative cycles for which keeping only that negative cycle results in the distance from S to E being well-defined.

○ T ○ U ○ V ○ W

(iii) Consider what happens if all negative cycles except one are removed from the graph. You are interested if Bellman-Ford starting from S then reports that distances are not well-defined.

Mark all of the negative cycles for which keeping only that negative cycle results in Bellman-Ford starting from S reporting that distances are not well-defined.

○ T ○ U ○ V ○ W

(iv) Now consider what happens if **all** negative cycles are removed from the graph, and you run Bellman-Ford starting from S. True or false: The shortest-paths tree produced by Bellman-Ford in this case is independent of the order in which it updates the edges.

○ True ○ False

The main insight from the previous problem is the shortest path can still be well-defined between some pairs of vertices despite the existence of negative cycles.

(b) (**10 points**) Give an algorithm that takes in a weighted directed graph $G = (V, E)$ (potentially with negative weight cycles) along with two vertices s and t as input, and outputs the length of the shortest path between s and t if it is well-defined and the string "`no well-defined shortest path`" otherwise.

Any algorithm that correctly solves this problem and runs in time polynomial in $|V|$ and $|E|$ will receive full credit. You may freely use algorithms covered in lecture in a black-box fashion. *Hint: modify the graph and feed the modified graph to the Bellman–Ford algorithm.*

(i) Give a description of your algorithm.

(ii) Give a run-time analysis of your algorithm.