Final

Name:

SID:

Exam Room:

Name of student to your left:

Name of student to your right:

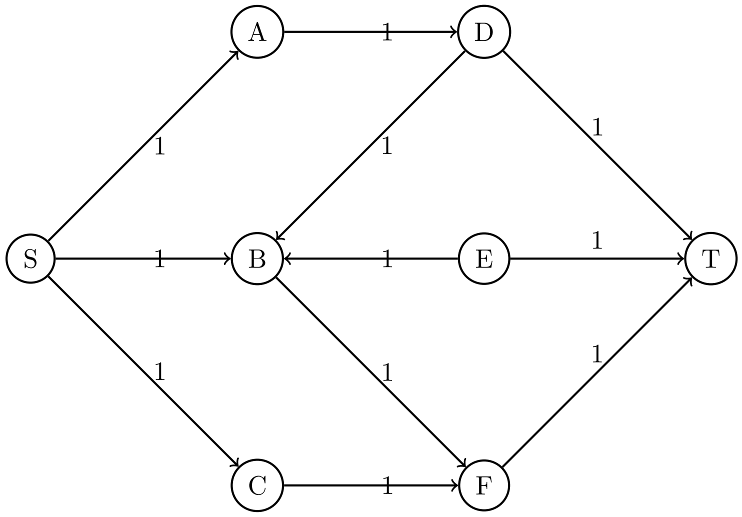
Do not turn this page until your instructor tells you to do so.

- After the exam starts, write your name on every odd-numbered page. We reserve the right to deduct points if you do not, and you will not be allowed to do so after time is called.
- For short question, your answers must be written clearly inside the box region. Any answer outside the box will not be graded. For longer questions, if you run out of space, you must clearly mention in the space provided for the question if part of your answers is elsewhere.
- The number of points roughly indicate the amount of time (in minutes) worth spending on each problem. Try to answer all questions. Not all parts of a problem are weighted equally. Before you answer any question, read the problem carefully. Be precise and concise in your answers.
- You may consult only *three sheets of notes*. Apart from that, you may not look at books, notes, etc. Calculators, phones, computers, and other electronic devices are NOT permitted.
- There are **26** pages on the exam (counting this one). Notify a proctor immediately if a page is missing.
- **By "reduction" in this exam it is always meant "polynomial-time reduction." Also, when you are asked to prove that a problem is NP-complete, no need to show that it is in NP, unless asked to do so.**
- **You have 170 minutes: there are 15 questions on this exam worth a total of 170 points.**

Name:

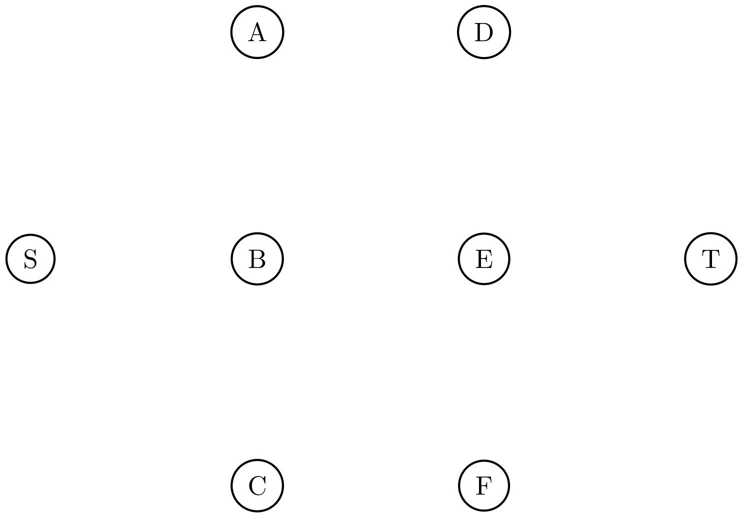
1 Maximum Flow (6 points)

Consider the following directed graph with all edge capacities equal to 1.



In the first step of Maximum-Flow algorithm, we increase the flow along $S \to A \to D \to B \to F \to T$ by one unit.

1. Draw the residual graph after this step.



2. What happens next in the execution of Max-Flow algorithm? (the algorithm does not necessarily run for three steps)

   - Send $\boxed{}$ unit(s) of flow on path $S \rightarrow \boxed{} \rightarrow T$
   - Send $\boxed{}$ unit(s) of flow on path $S \rightarrow \boxed{} \rightarrow T$
   - Send $\boxed{}$ unit(s) of flow on path $S \rightarrow \boxed{} \rightarrow T$

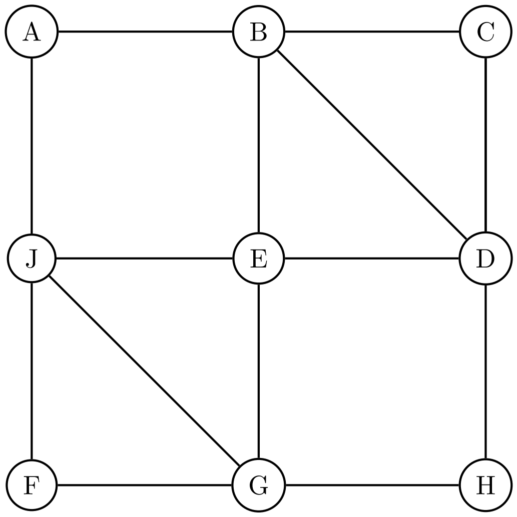
3. What is the minimum $S - T$ cut in the graph?

   - S-side of the partition = $\{S, \boxed{}\}$
   - T-side of the partition = $\{T, \boxed{}\}$

Name:

2 Minimum Spanning Tree (6 points)

Consider the following graph, and the list of edges in increasing order of their weights.



1. (A, B)
2. (B, C)
3. (C, D)
4. (B, E)
5. (B, D)
6. (A, J)
7. (F, J)
8. (D, E)
9. (E, G)
10. (D, H)
11. (G, J)
12. (G, H)
13. (E, J)
14. (F, G)

Run Kruskal's algorithm and Prim's algorithm (starting at node E for the latter). Which are the second, fourth, and seventh edges that are added to the resultant MSTs, in each of the two algorithms?

	Kruskal's Algorithm	Prim's Algorithm
Step 2		
Step 4		
Step 7		

3 Vertex Cover (6 points)

Suppose $x_1 = \frac{3}{4}, x_2 = \frac{1}{3}, x_3 = \frac{2}{3}, x_4 = \frac{2}{3}, x_5 = \frac{1}{4}, x_6 = \frac{2}{3}, x_7 = \frac{1}{3}$ be the optimal solution to the linear programming relaxation of vertex cover on a graph G (all vertex weights are 1).

1. Choose the pairs of vertices that are NOT edges in the graph. Bubble in your answers. [Note: This part will be graded automatically. Please mark your answer clearly.]
   - ○ (1,2)
   - ○ (2,3)
   - ○ (2,4)
   - ○ (4,5)
   - ○ (3,6)
   - ○ (6,7)
2. The size of the smallest vertex cover in G is at least $\boxed{}$
3. Choose the vertices that belong to the vertex cover output by the approximation algorithm. Bubble in your answers. [Note: This part will be graded automatically. Please mark your answer clearly.]
   - ○ 1
   - ○ 2
   - ○ 3
   - ○ 4
   - ○ 5
   - ○ 6
   - ○ 7
4. The algorithm yields a $\boxed{}$ approximation on this graph.

Name:	

4 Zero-Sum Games (3 points)

Consider a zero-sum game given by the following matrix (indicating the payoffs to the row player)

	C_1	C_2	C_3
R_1	1	2	3
R_2	2	2	1
R_3	1	4	2

Suppose the column player goes second and has fixed the following probabilistic strategy:

C_1	C_2	C_3
0.3	0.2	0.5

what is the best strategy for the row player going first?

R_1	R_2	R_3

5 Linear Programming (6 points)

1. Write the dual of the following linear program.

$$\begin{aligned}
&\max\ \ 3x_1 + x_2 + 4x_3 + 4x_4 + x_5 \\
&x_1 + x_2 + x_3 + x_4 + x_5 \leq 2 \\
&x_1 - x_2 + 2x_3 + 3x_4 \leq 1 \\
&x_1, x_2, x_3, x_4, x_5 \geq 0
\end{aligned} \tag{1}$$

2. Prove that the optimum of the linear program given in part 1 is at most 5.

Name:

6 Factoring Numbers via Circuit SAT (5 points)

Let $n = 1010001010100010101111110000010101010111111111111111111000000000111000000000111$.

Describe a circuit C (in at most two sentences), such that solving the CircuitSAT problem on the instance C will yield a factor p dividing n.

7 Find the satisfying one (5 points)

You are given a 3-SAT formula Φ and two assignment sets x_1, x_2 such that one of the assignment sets satisfies Φ, while the other satisfies at most 90% of the clauses in Φ.

Describe an $O(1)$-time algorithm that finds the satisfying assignment among x_1, x_2 with probability 0.99.

Main Idea:

Prove:

Name:

8 Updating Distances (8 points)

We have a directed graph $G = (V, E)$, where each edge (u, v) has a length $\ell(u, v)$ that is a positive integer. Let n denote the number of vertices in G. In other words, $n = \|V\|$. Suppose we have previously computed a $n \times n$ matrix $d[\cdot, \cdot]$, where for each pair of vertices $(u, v) \in V$, $d[u, v]$ stores the length of the shortest path from u to v in G. The $d[\cdot, \cdot]$ matrix is provided to you. Now we add a single edge (a, b) to get the graph $G' = (V; E')$, where $E = E \cup \{(a, b)\}$. Let $\ell(a, b)$ denote the the length of the new edge. Your job is to compute a new distance matrix $d'[\cdot, \cdot]$, which should be filled in so that $d'[u, v]$ holds the length of the shortest path from u to v in G', for each $u, v \in V$.

1. Write a concise and efficient algorithm to fill in the $d'[\cdot, \cdot]$ matrix. You should not need more than about 3 lines of pseudocode. (psuedocode only)

2. What is the run-time of your algorithm?

9 Finding Bridges (20 points)

A bridge of an undirected graph G is an edge whose removal disconnects G.

1. An edge is a *bridge* if and only if it is not part of any
2. Suppose we perform a DFS of the graph G then a *bridge* can be (bubble all possibilities): [Note: This part will be graded automatically. Please mark your answer clearly.]

   ◯ Tree edge

   ◯ Back edge
3. Suppose we perform a DFS of the graph G starting from s. At some point during the execution of the DFS, let us suppose the current node is v_t. Suppose the path from the root of DFS tree to the current node v_t is given by,

$$s = v_0 \to v_1 \to v_2 \ldots \to v_{t-1} \to v_t$$

   and the DFS encounters a back edge $v_t \to v_i$ for some $i < t$. Which edges of the graph are definitely not bridges?
4. Use the hints above to design an $O(|E|)$ time modification of DFS to find all the bridges in an undirected graph G. For simplicity, let us assume that assume that G is connected.

   Briefly state your main idea and fill in the blanks in the following pseudocode. You don't need to use all the lines that we provide.

   Main Idea:

Name:

Pseudocode:

Let

$$\begin{aligned} depth[u] &= \text{depth of node } u \text{ in DFS tree} \\ depth &= \text{current depth of DFS} \\ visited[u] &= \text{boolean indicating whether visited } u \text{ already} \\ isBridge[u,v] &= \text{True if } (u,v) \text{ is a bridge in the graph, and initialized to False for every edge at the beginning} \\ earliest[u] &= \underline{\hspace{6cm}} \\ treeEdge[u] &= \text{Tree edge leading to u} \end{aligned}$$

```
procedure Explore($u$):
for each edge$(u, v) \in E$ do
    if visited[v] = True then
        ________________________________
        ________________________________
        ________________________________
        ________________________________
        ________________________________
    else
        treeEdge[v] = (u,v)
        Previsit(v)
        Explore(v)
        Postvisit(v)
    end if
end for
end procedure

procedure Previsit($u$):
$depth = depth + 1$
depth[u] = depth
visited[u] = True
end procedure

procedure Postvisit($u$):
$depth = depth - 1$
________________________________
________________________________
________________________________
________________________________
________________________________
________________________________
________________________________
end procedure
```

10 Gambling (15 points)

You walk into a casino. You have M dollars of money, and want to play exactly n rounds of games. Let us call it a *success* if at the end of the n games, you have exactly $2M$ dollars (no more, and no less).

For each of the n rounds, you can choose to play either Game A or Game B. Game A costs \$1 to play, and returns \$2 with probability 0.6 (and \$0 with probability 0.4). Game B costs \$3 and returns \$15 with probability 0.2 (and \$0 with probability 0.8).

(The returns do not include the cost, so if you play and win Game A, your net gain is \$1.)

We will now design a dynamic programming algorithm to compute the probability of *success* of the optimal strategy.

Define the subproblem as the following:

$$T[m, \ell] = \text{Optimal strategy's probability of } \textit{success}\text{, if you have } m \text{ dollars at the end of } \ell \text{ games}$$

1. Base cases:

2. Recurrence relation:

3. The run-time of the algorithm is $\Theta(\boxed{})$

Name:

11 Roadside Assistance (20 points)

You are the CEO of a towing company that serves a network of roads connecting n cities. The road network is given by an undirected graph $G = (V, E)$ where $V = \{1, \ldots, n\}$ is the set of cities, and E denotes the set of roads connecting pairs of cities in V.

On each road $(i, j) \in E$, there are w_{ij} accidents that occur each day, which need road-side assistance. An accident occuring on road (i, j) can only be serviced by a tow-truck from city i or city j. Each accident needs exactly one tow-truck for assistance.

At each city i, the company parks t_i tow-trucks.

1. (10 points) Describe a polynomial-time algorithm to determine whether the company can service all the accidents. If so, determine how the company should service the accidents, i.e., For each road $(i, j) \in E$, the algorithm must determine how many of the w_{ij} accidents on the road are assisted by trucks from i, and how many by trucks from j?

   Describe the main idea of the algorithm precisely, proof of correctness is not needed.

2. (10 points) The company realized that it is paying too much parking fees for the trucks at the cities. Parking a truck in city i costs c_i.

   The company would like to rearrange all the trucks, so as to minimize the total parking costs, while still being able to assist all the accidents on every road.

   Write a linear program to determine how to rearrange the trucks, so that they can still assist all the w_{ij} accidents on each road (i, j), but minimize the total parking cost.

   (a) What are the variables of the linear program?

   (b) What is the objective function?

(c) What are the constraints?

Name:

12 Complete the sentences: *(26 points)*

When asked for a bound, always give the tightest bound possible.

1. The number of strongly connected components in an n vertex DAG is at least $\boxed{}$.
2. Suppose $E[i, j]$ is the ij^{th} entry of the table in the dynamic programming based algorithm for edit distance, then $E[5, 6]$ can be computed using the entries: $\boxed{}$
3. Every directed graph can be decomposed in to a $\boxed{}$
4. The solution to the recurrence $T(n) = 8T(n/4) + O(n^2)$ is $\boxed{}$.
5. Suppose $T(2^n) = T(n) + 1$ and $T(1) = 1$ then $T(n) = \boxed{}$.
6. $\boxed{}$ edges occur in the DFS tree only if the graph is directed.
7. Running DFS starting from a node in a $\boxed{}$ yields a strongly connected component of the graph.
8. Minimum spanning tree of a graph never contains the heaviest edge in every $\boxed{}$
9. Minimum spanning tree of a graph always contains the lightest edge in every $\boxed{}$
10. Suppose a hash function $h : \{0, 1, \dots, p-1\} \to \{0, 1, \dots, p-1\}$ chosen from a universal hash family then $\Pr[h(2) = 2 \cdot h(1) \mod p] = \boxed{}$
11. There are 8 symbols $\{A, B, C, D, E, F, G, H\}$ all of whose frequencies are within the range $[\frac{1}{8} - 10^{-3}, \frac{1}{8} + 10^{-3}]$. The Huffman encoding of these symbols will consist of strings $\boxed{}$ in some order.
12. The cost of every TSP tour in a graph is always $\boxed{}$ than the cost of a minimum spanning tree. (positive weights, complete graph)
13. Two polynomials $p(x), q(x)$ given in the (point,value) representation can be multiplied in time $\boxed{}$. (both are degree at most n)

13 True or false? (12 points)

Bubble in the right answer. No explanation needed. No points will be subtracted for wrong answers, so guess all you want! This part will be graded automatically. Please mark your answer clearly.

1. Dijkstra's algorithm does not work on every DAG with negative edge weights.

   ◯ True

   ◯ False

2. On a graph with integer capacities, the size of the minimum cut is integral.

   ◯ True

   ◯ False

3. On a graph with only integer capacities, the total maximum flow between any chosen pair s, t can be non-integral.

   ◯ True

   ◯ False

4. If a linear program has an integral optimal value then it has a unique solution.

   ◯ True

   ◯ False

5. The set of all functions from $\{0, 1, \ldots, p-1\}$ to $\{0, 1, \ldots, p-1\}$ is a universal hash family.

   ◯ True

   ◯ False

6. Minimum Vertex Cover is polynomial-time solvable on a tree.

   ◯ True

   ◯ False

7. Doubling the capacities of all edges of a minimum cut in a graph G, doubles the maximum flow.

   ◯ True

   ◯ False

Name:

8. Simplex algorithm can be used to solve linear programs in polynomial time.

   ◯ True

   ◯ False

9. The number of sub-problems in computing edit distance between two strings $x[1 \dots m]$ and $y[1 \dots n]$ is $O(mn)$.

   ◯ True

   ◯ False

10. The value of the edit distance between two strings $x[1, \dots, m]$ and $y[1, \dots, n]$ can be computed using $O(m+n)$-space.

    ◯ True

    ◯ False

11. For any proof, if a cheating verifier cannot recover the prover's secret witness from an interactive proof then the proof is zero-knowledge.

    ◯ True

    ◯ False

12. Minimum Vertex Cover is known to be an NP problem.

    ◯ True

    ◯ False

14 True or False or Maybe..? (12 points)

In the remaining questions there are four possible answers: (1) True (T); (2) False (F); (3) True if and only if **P** $=$ **NP** *(=); (3) True if and only if* **P** $\neq$ **NP** *(≠).*

Bubble in the most appropriate one. **Note:** *By "reduction" in this exam it is always meant "polynomial-time reduction."*
This part will be graded automatically. Please mark your answer clearly.

13. There is a reduction from Bipartite Matching to Independent Set.

   ◯ True

   ◯ False

   ◯ =

   ◯ ≠

14. There is a reduction from Independent Set to Maximum Flow.

   ◯ True

   ◯ False

   ◯ =

   ◯ ≠

15. There is a polynomial-time algorithm for 3-SAT.

   ◯ True

   ◯ False

   ◯ =

   ◯ ≠

16. There is a reduction from Factoring to Rudrata Path.

   ◯ True

   ◯ False

   ◯ =

   ◯ ≠

17. Every problem in **NP** can be written as an integer linear program.

    ◯ True

    ◯ False

    ◯ $=$

    ◯ $\neq$

18. Minimum Spanning Tree is an NP problem.

    ◯ True

    ◯ False

    ◯ $=$

    ◯ $\neq$

15 NP-completeness *(20 points)*

Note: By "reduction" in this exam it is always meant "polynomial-time reduction." For the reductions in Problem **??** mention the problem you are using, direction and construction of the reduction). Also, when you are asked to show that a problem is **NP**-complete, no need to show that it is in **NP**, unless asked to do so.

You may assume that the following problems are known to be **NP**-hard.

- Rudrata Path or Hamiltonian Path
- Hamiltonian Cycle
- Vertex Cover
- Independent Set
- 3-SAT
- CircuitSAT
- Integer Programming
- Clique
- 3D-Matching
- 4-SAT
- 3-Coloring

Name:

(8 points) 1) Balanced 3SAT: In the balanced 3-SAT formula, the input is a 3-SAT formula and the goal is to find a satisfying assignment $x = (x_1, \ldots, x_n)$ such that exactly half the variables are assigned 0, and half assigned 1.

Proof: *We will reduce the problem ...* *to the problem ...*

Given an instance Φ of the problem ... *we construct an instance Ψ of the problem ...*

as follows ...

The proof that this is a valid reduction is as follows:

∎

(6 points) 2) Degree-Bounded Spanning Tree: Given a graph G and integers $(b_1, \ldots, b_n)$, find a spanning tree T for the graph such that the degree of the i^{th} vertex in the tree is at most b_i.

Proof: *We will reduce the problem ... to the problem ...*

Given an instance Φ of the problem ... we construct an instance Ψ of the problem ...

as follows ...

The proof that this is a valid reduction is as follows:

■

Name:

(6 points) 3) Disjoint Sets: Given a collection of subsets $\mathcal{S} = \{S_1, \ldots, S_m\}$ of $\{1, \ldots, n\}$ and an integer k, pick k of these subsets $\{S_{i_1}, \ldots, S_{i_k}\}$ that are pairwise disjoint, i.e., $S_{i_a} \cap S_{i_b} = \emptyset$ for all $a \neq b$.

Proof: *We will reduce the problem ... to the problem ...*

Given an instance Φ of the problem ... we construct an instance Ψ of the problem ...

as follows ...

Proof of correctness of reduction not necessary ■

$$\boxed{\boxed{\text{DO NOT TEAR ANY PAGES OFF YOUR EXAM.}}}$$

(Extra page for scratch work.)