I certify that the work submitted with this exam is mine and was generated in a manner consistent with this document, the course academic policy on the course website, and the Rutgers academic code. I also certify that I will not disclose this exam to others who are taking this course and who will take this course in the future without the authorization of the instructor.

Name (please print): _____

NetID: _____

Section: _____

Signature: _____

Date: _____

Name: _____   **NetID: _____**   **Section: _____**

Problem Number(s)	Possible Points	Earned Points
1	10	
2	15	
3	10	
4	10	
5	10	
6	10	
7	10	
8	10	
9	15	
Total	100	

Exam Time: 12 hours, 9 problems (14 pages, including this page)

- Write your name on this page and the last page, put your initials on the rest of the pages.
- If needed, use the last page to write your answer.
- Show your work to get partial credits.
- Show your rational if asked. Just giving an answer can't give you full credits.
- You may use any algorithms (procedures) that we learned in the class.
- Keep the answers as brief and clear as possible.

Name: _____   NetID: _____   Section: _____

1. **(10 points) Circle True or False (no partial credit)**

    (1) T   F   Given a directed graph G and a vertex v in the graph, breath first search (BFS) can be used to detect if v is part of a cycle in the graph.

    (2) T   F   Let P be a shortest path from some vertex s to some other vertex t in a directed graph. If the weight of each edge in the graph is *decreased* by one, then P will still be a shortest path from s to t.

    (3) T   F   Bellman-Ford algorithm can be used to detect if there exist negative circles in a directed graph.

    (4) T   F   Kruskal's algorithm is always correct even in graphs with negative edge weights.

    (5) T   F   For any flow network, there is only one unique way to assign flow value to the edges so as to achieve the maximum flow for the network.

2. **(15 points) Graph Basics**

    (1) (5 points) For the directed graph in the right side, provide the *adjacency matrix representation* and the *adjacency list representation* of the graph.
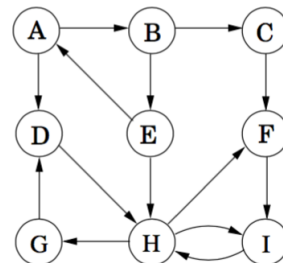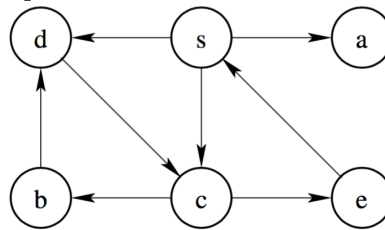
Name: _____   **NetID:** _____   **Section:** _____

(2) (5 points) DFS and its application: Run the DFS-based topological sort algorithm on the following graph. Whenever there is a choice of vertices, choose the one that is alphabetically first. Give the resulting topological ordering of the vertices.
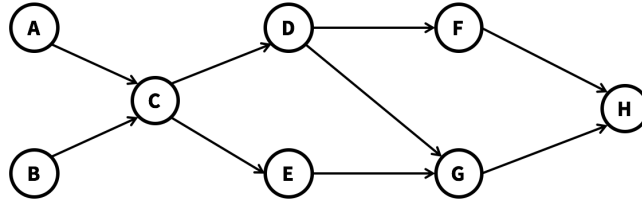

(3) (5 points) BFS and its application: Run BFS algorithm on the following graph starting with vertex s. Whenever there is a choice of vertices, choose the one that is alphabetically first. What is the order that the vertices are visited? What is the shortest path from vertex s to vertex b?

Name: _____   NetID: _____   Section: _____

3. **(10 points) Dijkstra's Algorithm**
   We have the following directed graph G, where the number on each edge is the
   cost of the edge.
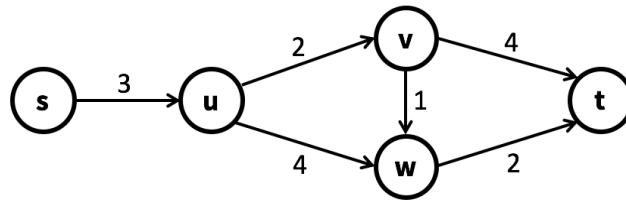
   (1) (8 points) Step through Dijkstra's Algorithm on the graph starting from
       vertex s, and complete the table below to show what the arrays *d* and *p* are at
       each step of the algorithm. For any vertex x, *d[x]* stores the current shortest
       distance from s to x, and *p[x]* stores the current parent vertex of x.

	d[s]	d[u]	d[v]	d[w]	d[t]	p[s]	p[u]	p[v]	p[w]	p[t]
Initialization	0	∞	∞	∞	∞	None	None	None	None	None
Immediately after iteration 1	0	3	∞	∞	∞	None	s	None	None	None
Immediately after iteration 2										
Immediately after iteration 3										
Immediately after iteration 4										
Final status										

   (2) (2 points) After you complete the table, provide the returned shortest path
       from s to t and the cost of the path.

Name: _____   NetID: _____   Section: _____

4. **(10 points) Monte-Carlo Randomized Algorithm**
   Suppose we have the following undirected graph, and we know that the two
   bolded edges (B-E and G-E) constitute the global minimum cut of the graph.
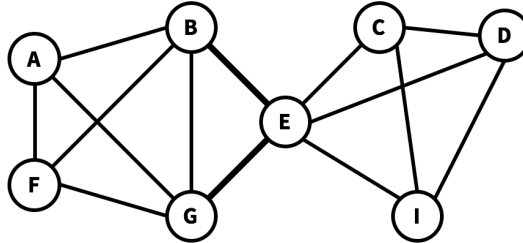

   (1) (5 points) If we run the Karger's algorithm for just one time to find the global
       minimum cut, what is the probability for the algorithm to find the minimum
       cut correctly? Please show your reasoning process, just showing the final
       answer will get no point.

   (2) (5 points) How many times do we need to run the Karger's algorithm if we
       want to guarantee that the probability of success is greater than or equal to
       0.95, by "success" we mean that there is at least one time the Karger's
       algorithm correctly found the minimum cut. [You do not have to work out the
       exact value of a logarithm]

Name: _____   **NetID:** _____   **Section:** _____

5. **(10 points) Minimum Spanning Tree**
    Consider the following weighted undirected graph:
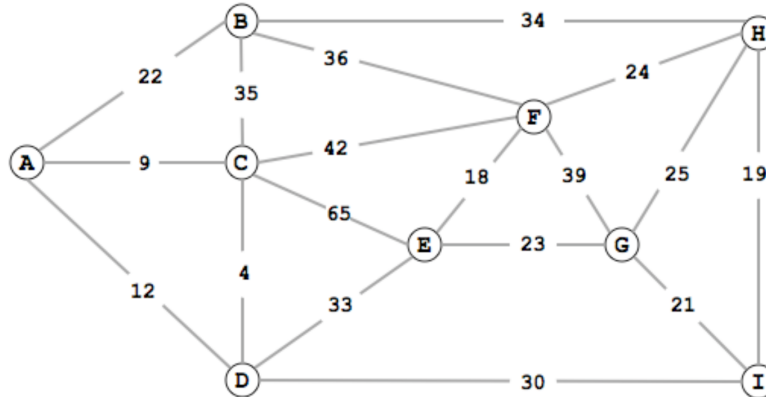
(1) (5 points) Assume we run Prim's MST algorithm starting at vertex A. List the edges that get added to the tree in the order in which the algorithm adds them. [You can denote an edge by its two adjacent vertices, e.g., (A, B)].

(2) (5 points) Now we use Kruskal's algorithm to find the MST, also, list the edges that get added to the tree in the order in which the algorithm adds them. [You can denote an edge by its two adjacent vertices, e.g., (A, B)].

Name: _____   NetID: _____   Section: _____

6. **(10 points) Strongly Connected Components**
   You are hired by a game design company and one of their most popular games is *The Journey*. The game has a ton of quests, and ***for a player to win, the player must finish all the quests***.

   There are a total of *N* quests in the game. Here is how the game works: the player can *arbitrarily* pick one of the *N* quests to start from. Once the player completes a quest, they unlock some other quests. The player can then choose one of the unlocked quests and complete it, and so on.

   For instance, let's say that this game had only 4 quests: A, B, C, and D. Let's say that after you complete
   • quest A, you unlock quests [B, D].
   • quest B, you unlock quests [C, D].
   • quest C, you unlock nothing [ ].
   • quest D, you unlock quest [C].

   Is this game winnable? Yes, because of the following scenario:

   The player picks quest A to start with. At the end of the quest A, the unlocked list contains [B, D]. Say that player chooses to do quest B, then the unlocked list will contain [C, D]. Say that player chooses to complete quest C, then the unlocked list will contain quest [D]. Finally, player finishes quest D.

   Note that if the player had started with quest C instead of quest A, they would have lost this game, because they wouldn't have unlocked any quest and would be stuck. *But the game is still winnable, because there is **at least one** starting quest which makes the player win*.

   The Journey has *N* quests, enumerated as $\{n_1, n_2, \ldots, n_N\}$. We construct a directed graph G for the game, where each node in G represents a quest. If completing quest n_i will unlock quest n_j, then there is a directed edge <n_i, n_j> from n_i to n_j in the graph. Suppose the total number of edges in the graph is *M*.

   (1) (6 points) We call a quest as a *lucky quest* if starting from this quest will allow the player to win the game. In the example above, quest A is a lucky quest. Show that (a) All lucky quests are in the same strongly connected component of G, and that (b) Every quest in that component is a lucky quest. [We are expecting a short but rigorous proof for both claims]

(2) (4 points) Suppose there is at least one lucky quest. Give an algorithm that runs in time O(N+M) and finds a lucky quest. You may use any algorithm we have seen in class as subroutine. [We are expecting pseudocode or a description of your algorithm, and a short justification of the runtime]

Name: _____   **NetID: _____**   **Section: _____**

7. **(10 points) Greedy Algorithms**
   On Thanksgiving day, you arrive on an island with *n* turkeys. You've already had thanksgiving dinner (and maybe you prefer tofurkey anyway), so you don't want to eat the turkeys; but you do want to wish them a Happy Thanksgiving.

   However, the turkeys each have very different sleep schedules. Turkey *i* is awake only in a single closed interval $[a_i, b_i]$. You don't want to disturb the sleeping turkeys, so each time you find an awake turkey, you wish it a happy thanksgiving, and talk to it until it falls asleep again. However, you do want to wish and talk with as many turkeys as possible.

   Design a greedy algorithm which takes as input the number of turkeys *n*, and the *n* lists of intervals $[a_i, b_i]$, and outputs the maximum number of turkeys *m* that you can talk to. You are expected to write the pseudocode of your algorithm.

Name: _____   NetID: _____   Section: _____

8. **(10 points) Max-flow and Min-cut**
   Consider the following flow network.
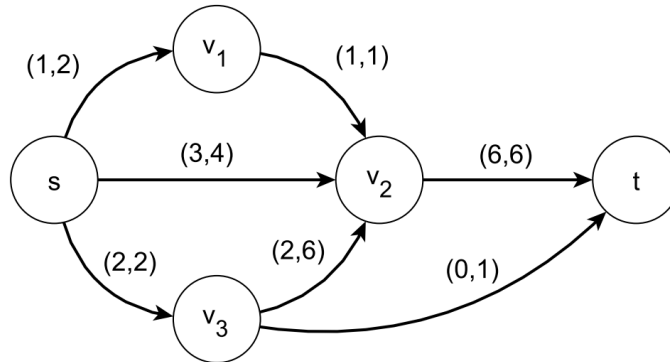
The figure describes a flow f and the capacity of the edges: if (x, y) appears next to an edge e, then the capacity of the edge c_e is y, and the flow f_e that goes through e in f is x. For example, if $e = (s, v_1)$, then $c_e = 2$ and $f_e = 1$.

(1) (4 points) Draw the residual network G_f of the above flow f. [Draw a graph containing all the nodes, edges, and the values on the edges]

Name: _____   **NetID:** _____   **Section:** _____

(2) (3 points) Find an augmenting path that will increase the flow by 1. You only need to list the vertices in the path and indicate the resulting flow in the following figure (using the same notation as the above figure).
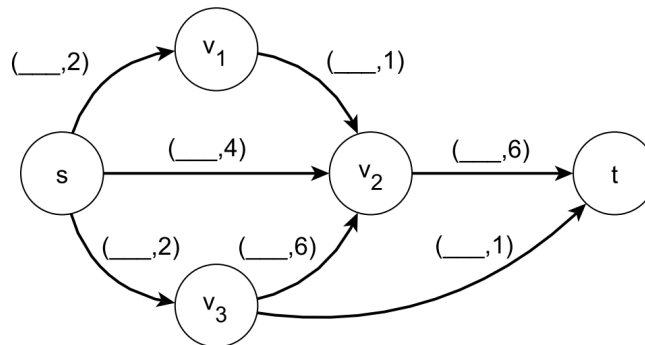
(3) (3 points) Find a minimum s-t cut in the graph (where the weight of an edge is its capacity). Briefly justify why the cut you found is a minimum cut.

Name: _____   NetID: _____   Section: _____

9. **(15 points) Dynamic Programming**
   A country has coins with k denominations $1 = d_1 < d_2 < \cdots < d_k$, and you want to make change for n cents using the smallest number of coins.

   For example, in the United States we have $d_1 = 1$, $d_2 = 5$, $d_3 = 10$, $d_4 = 25$, and the change for 37 cents with the smallest number of coins is 1 quarter, 1 dime, and 2 pennies, which are a total of 4 coins.

   To solve for the general case (change for n cents with k denominations $d_1 \ldots d_k$), we refer to dynamic programming to design an algorithm.

   (1) (5 points) We will come up with sub-problems and recursive relationship for you. Let $C[n]$ be the minimum number of coins needed to make change for n cents, then we have:
   $$C[n] = \begin{cases} \underset{\forall\, d_i \leq n}{\text{minimum}}\{C[n - d_i] + 1\}, & if\ n > 0 \\ 0, & if\ n = 0 \end{cases}$$
   Explain why the above recursive relationship is correct.
   [Formal proof is not required]

Name: _____   **NetID:** _____   **Section:** _____

(2) (5 points) Use the relationship above to design a dynamic programming algorithm, where the inputs include the k denominations $d_1 \ldots d_k$ and the number of cents n to make changes for, and the output is the minimum number of coins needed to make change for n.

Provide the pseudocode of your algorithm and briefly justify the runtime of your algorithm using big-O notation.

Name: _____   **NetID: _____**   **Section: _____**

(3) (5 points) Adapt your algorithm above by tracking some useful information during the DP procedure, so that it returns the actual method (i.e., the number of coins for each denomination) to change for n cents, not just the minimum number of denominations.

Name: _____ **NetID: _____** **Section: _____**

(You may use this page to write answers if needed. Please mark the problem number clearly)