CS 344 - Spring 2021
Exam 2
100 points total

Instructions

Academic Integrity. You should type the following text on the first page of your midterm.

I have not discussed this exam with anyone except the professor and the TAs of CS 344. I have not used any online resources during the exam except for those accessible from the Canvass website.

What to Write

- On the first page write your full name, netID, and the academic integrity statement above. Do not write anything else.
- Start each problem on a new page.
- Typed assignments receive 2 extra points.
- If your assignment is hand-written please write very legibly.
- Assignment should be submitted as a single PDF in the same way as a HW problem.

Skipping Problems Remember that if you skip a problem you get 25% credit. If you give an answer that is on the right track you will get more than 25% in partial credit, so don't be afraid to attempt problems if you think you know how to approach them. But keep in mind that if you simply don't know the answer, writing a very wrong answer may get 0%.

The 344 Library: Throughout the test, any algorithm you design may use any of the functions or data structures we covered in class without explanation. In particular, here are some functions from our library on graph algorithms that you may find it useful to take advantage of.

- The following functions take as input a specific source $s \in V$ and output dist(s,v) for all vertices $v \in V$; they work on both directed and undirected graphs.
  - BFS(G,s)
  - Dijkstra(G,s)
  - DAG-SP(G,s)
  - BellmanFord(G,s)
    - Bellman Ford returns "Negative Cycle Detected" if a negative cycle is detected
- Johnson(G) returns dist(x,y) for all pairs of vertices x, y in V
- TopSort(G). After running this algorithm, you can assume that the vertex set $V = \{v_1, ..., v_n\}$ is in topological order.
  - TopSort(G) returns "NOT a DAG" if the input graph G is not a DAG.

Graph Problems For all problems that involve a graph, you can assume that the graph G has no isolated vertices. In particular, you can always assume that $|E| \geq |V|/2$.

Dynamic Programming Problems Problems 3 and 4 are dynamic programming problems. For both of these problems, your answer should follow the same basic template as from HW 4. In particular, your solution must contain ALL of the following, in the exact order below. *To repeat, this only applies to problems 3 and 4.*

1. What the values in your table correspond to. So e.g. for longest increasing subsequence (from class) I would write something like: $T[i]$ is the length of the longest increasing sequence ending at $A[i]$
2. The DP-relation that computes one table entry in terms of other table entires. So for longest increasing subsequence I would write something like: $T[i] = \max_j T[j] + 1$, where the maximum is taken over all $j < i$ with $A[j] < A[i]$.

3. How do you initialize the table: so for longest increasing subsequence I would write $T[0] = 1$.
4. Which entry of the table you return at the end of the algorithm. So for example for longest increasing subsequence I would write: return FindMax(T)
5. Pseudocode for the final algorithm.

List of NP-hard problems For this exam, you can assume that the following problems are NP-hard. You will want to use one of the problems on this list to solve problem 5.

- HamCycle(G)
- Clique(G,k)
- VertexCover(G,k)
- 2-partition(S)
- SubsetSum(S,B)

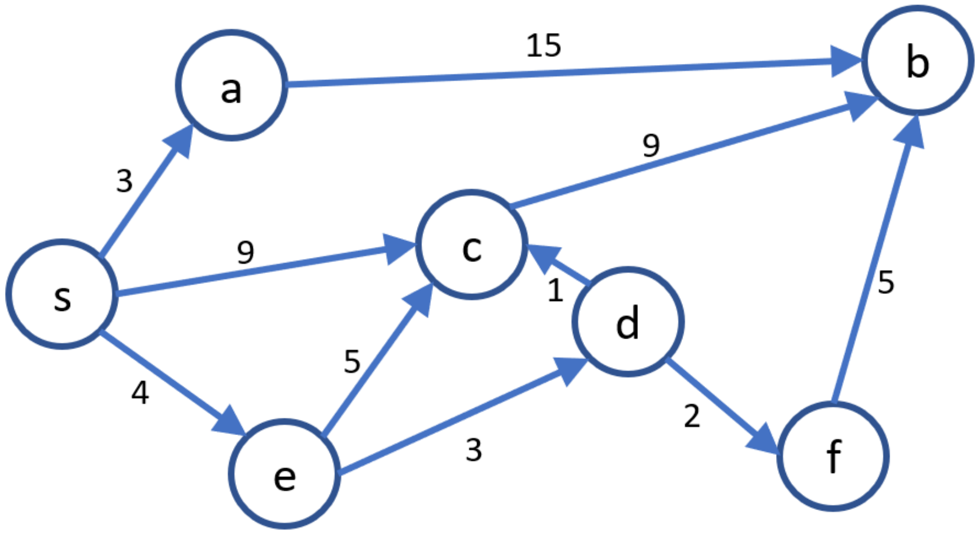

Figure 1: Graph for problem 1 part 1.

1 Problem 1 (20 points)

Part 1 (10 points) Say that you run Dijkstra(G,s) on the graph in Figure 1. Recall that Dijkstra's explores vertices one at a time. What is the order in which Dijkstra(G,s) explores vertices when run the graph G and source s below? What is the distance label of each vertex when it is explored?

No need to justify your work. Your solution only needs to include the following table:

- The table should have 2 rows and 7 columns (one column per vertex)
- The first row should include the vertices of G in the order in which they are explored.
- The second row should include the label of each vertex when that vertex is explored. So for example, if when b is explored it has label $d(b) = 1000$, then you put the number 1000 in the second row of the column of vertex b. (NOTE: 1000 is not actually the correct answer for vertex b.)

part 2 (10 points) Consider the DAG G in Figure 2. This graph has exactly two valid topological orderings. You should write down BOTH of them. No need to justify work. Your solution only needs to include the two orderings; no need for explanation or anything else.

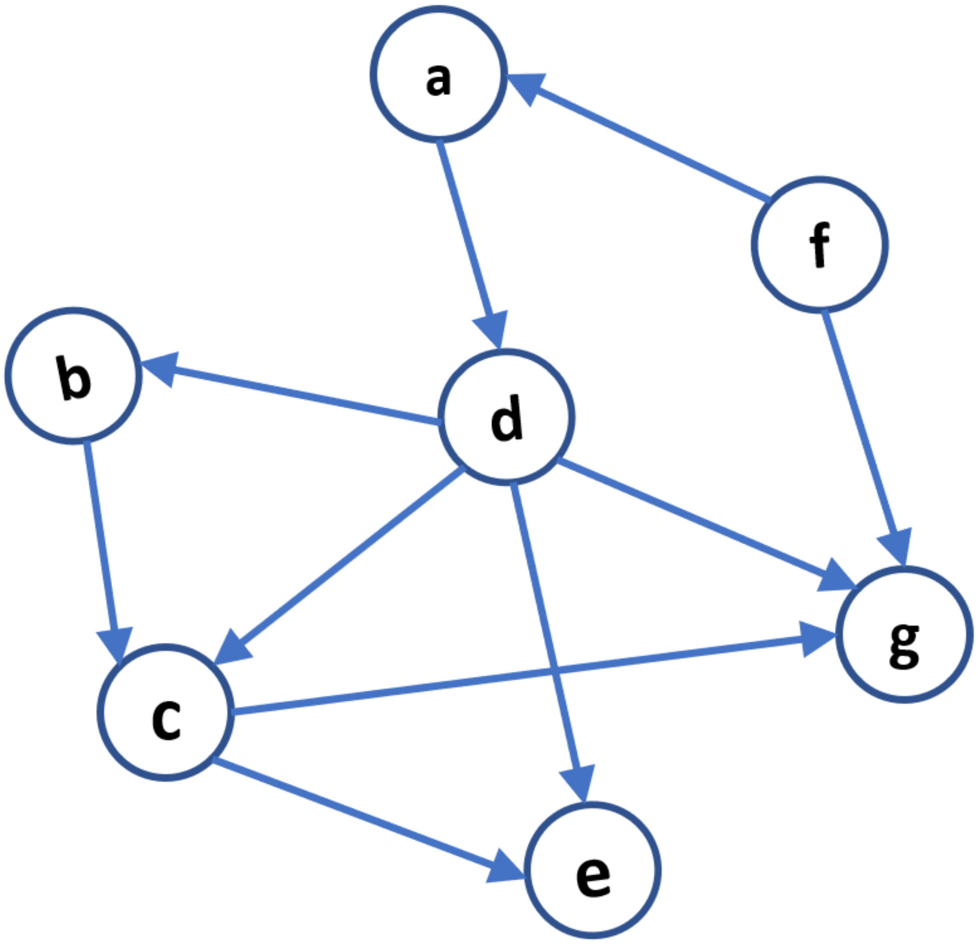

Figure 2: Graph for problem 1 part 2.

2 Problem 2 (15 points)

Both parts of this problem are about the reduced weights in Johnson's algorithm. Recall that Johnson's algorithm takes as input a graph G with n vertices and (possibly negative) edge weights $w(x, y)$ and outputs a graph G' which has the same vertices/edges as G, but with different edge weights. Let $w'(x, y)$ be the edge-weights in G', and recall that Johnson's algorithm guarantees that $w'(x, y) \geq 0$

Part 1 (5 points) Say that in the original input graph G, all weights were non-negative; that is $w(x, y) \geq 0$ for all edges (x, y). Now say you ran Johnson's algorithm on this graph G and ended up with G'. What can you say about the relationship between $w'(x, y)$ and $w(x, y)$? This relationship between $w'(x, y)$ and $w(x, y)$ will hold true for ALL edges (x, y). *You must briefly justify your answer*; I'm not look for a complex proof. There is a very simple 1-3 sentence explanation.

NOTE: the problem sounds open ended, but it is not. There is an extremely concrete thing you can say about the relationship between $w(x, y)$ and $w'(x, y)$. There is also an extremely concrete (and simple) explanation for why this relationship always holds. If you are saying something vague then you are not on the right track.

NOTE: the relationship between $w(x, y)$ and $w'(x, y)$ crucially depends on the fact that we are assuming in this part that $w(x, y) \geq 0$ for all edges (x, y).

Part 2 (10 points) Now consider a different graph G, which has exactly 10 vertices $v_1, v_2, ..., v_{10}$ and also has the property that all edge weights $w(x, y)$ are either -1, 0, or 1. Now say that you ran Johnson's algorithm on this graph and ended up with the graph G'. What is the largest possible edge weight in G'? That is, what is the maximum possible $w'(x, y)$? You need to give an example of a graph G and a specific edge (x, y) such that running Johnson's on G to create G' leads to this maximum possible $w'(x, y)$.

GRADING NOTE: the correctness of your example will be a huge part of your grade. So don't just give the final number without example.

No justification needed for this problem. all of you need to include is the example graph, the specific edge (x, y) in the example graph, and the value of $w'(x, y)$ achieved in your example graph.

3 Problem 3 (25 points)

In this problem we consider a variant of subset sum. Consider the following problem:

- INPUT:
  - A set $S = \{s_1, ..., s_n\}$ of positive integers.You can think of S as an array, so you can access any $s_i = S[i]$ in $O(1)$ time.
  - Each element $s_i \in S$ is colored either red or blue. You can think of each color as a field in the objects s_i. So you can access any s_i.color in $O(1)$ time.
  - A target positive integer B
- OUTPUT: the algorithm should output TRUE if there exists a subset $S' \subseteq S$ such that $\sum_{x \in S'} = B$ and S' contains at most 100 red elements. The algorithm should output FALSE otherwise. Note that you don't need to output the set S' itself; just true/false.

Show a DP algorithm that solves the above problem in $O(nB)$ time. You should include all of the elements of a dynamic programming solution mentioned at the beginning of the exam.

4 Problem 4 (22 points)

Consider the following problem:

- INPUT
  - A DAG $G = (V, E)$
  - Two specific vertices $s, t \in V$
- OUTPUT: the *number* of different paths in G from s to t that have ≤ 100 edges. You don't have the output the actual paths; you just have to figure out how many of them there are.

Show a dynamic programming algorithm that solves the above problem in $O(|E|)$ time. You should include all the elements of dynamic programming solution mentioned at the beginning of the exam. You should also include a brief (1-2 sentences) justification for why the running time is $O(|E|)$.

NOTE: reminder that you can assume in this class that all arithmetic operations (multiplication, addition, etc.) take $O(1)$ time.

5 Problem 5 (18 points)

Recall the suitcase packing problem from HW 4, problem 5. Consider the natural true/false version of this problem:

Suitcase(S,V,W)

- INPUT:
  - set $S = \{s_1, ..., s_n\}$, where each element s_i has an s_i.value and an s_i.weight
  - a target value V
  - a maximum weight W
- OUTPUT: return TRUE if there exists a set $S' \subseteq S$ such that the total value of S' is *at least* V and the total weight of S' is *at most* W. Note the at least and at most: intuitively, we want to find a suitcase with large value and small weight, which is why we require value *at least* V and weight *at most* W.

  Formally, the algorithm should output TRUE if there exists a set $S' \subseteq S$ such that BOTH of the following hold:

  - $\sum_{i=1}^{n} s_i.\text{value} \geq V$
  - $\sum_{i=1}^{n} s_i.\text{weight} \leq W$

The Problem: Show that the above problem Suitcase(S,V,W) is NP-complete.

HINT: note the list of NP-hard problems at the beginning of this exam. You will want to use one of those in your proof that suitcase packing is NP-complete.