Final Exam

Due Jun 9 at 11:59pm **Points** 100 **Questions** 18
Available Jun 7 at 8am - Jun 9 at 11:59pm 3 days **Time Limit** 110 Minutes

Instructions

You have 110 minutes to complete the exam.

You are allowed to use the following materials:

- Scratch Paper
- One sheet, 8.5x11", double-sided notes (typed or handwritten)
- Students may use any calculator
- Windows Calculator (and MacOS equivalent) "Programmer Mode" is OK!
- No material/equipment is allowed other than those specified.
- Any kind of cheating will not be tolerated.
  Good Luck

This quiz was locked Jun 9 at 11:59pm.

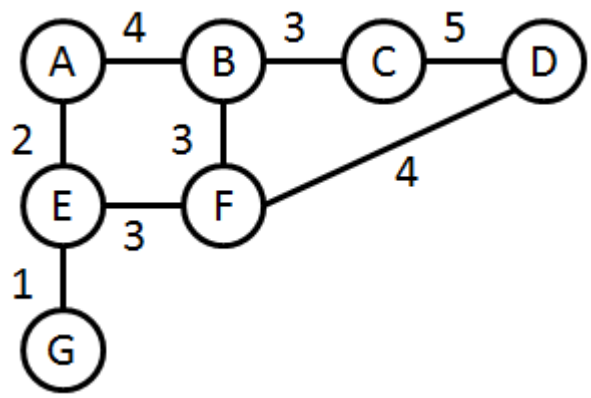
Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	110 minutes	87 out of 100

Score for this quiz: **87** out of 100
Submitted Jun 7 at 11:43pm
This attempt took 110 minutes.

Question 1 — 4 / 4 pts

What is the weight of the MST for the graph below? Give strictly a numeric answer.



Correct!

16

orrect Answers

16 (with margin: 0)

Question 2

0 / 3 pts

Given a weighted directed graph G = (V,E,w) and a shortest path P from s to t, if the weight of every edge is increased by one to produce G'= (V,E,w'), then P is also a shortest path in G'.

ou Answered

- (•) True

orrect Answer

- ( ) False

Question 3

3 / 3 pts

Given a weighted directed graph G = (V,E,w) and a shortest path P from s to t, if the weight of every edge is doubled to produce G'=(V,E,w'), then P is also a shortest path in G'.

Correct!

- (•) True

False

Question 4

3 / 3 pts

Let T be a complete binary tree with n vertices and edge weights of 1. Finding a shortest path from the root of T to a given vertex v ∈ T takes

O(n^2)

O(logn)

O(nlgn) time

Correct!

O(n) time

O(lg n) time.

Question 5

3 / 3 pts

Let M be a minimum spanning tree of G. Then, for any pair of vertices s and t, the shortest path from s to t in G is the path from s to t in M.

True

Correct!

False

Question 6

3 / 3 pts

If P does not equal NP, then there must exist a polynomial-time algorithm for 3-SAT.

- True
- False

Correct!

Question 7

3 / 3 pts

Every problem in P can be reduced to 3-SAT,

Correct!

- True

Every problem in P is in NP, and every problem in NP can be reduced to any NP-complete problem. Circuit-SAT is NP-complete.

- False

Question 8

3 / 3 pts

If we have a linear program where all of the coefficients and constants in every linear inequality are integers, then the optimal solution to this linear program must assign an integer value to every variable.

- True
- False

Correct!

Question 9

3 / 3 pts

You are using a polynomial time 2-approximation algorithm to find a tour t for the traveling salesman problem. Which of the following statements is true.

Correct!

- (selected) The cost of tour t is at most twice the cost of the optimal tour.
- All of the above
- The ratio of the cost of the optimal tour divided by the cost of tour t is 2.
- The cost of tour t is always 2 times the cost of the optimal tour.
- The tour t is never optimal.

Question 10

3 / 3 pts

If you discover a polynomial time algorithm for the 0-1 knapsack problem this will imply that P=NP.

Correct!

- (selected) True
- False

Question 11

3 / 3 pts

Given a set of items with the following sizes 4, 8, 5, 1, 7, 6, 1, 4, 2, 2 and bins of size 10. The items were packed into the minimum number of bins using the Next fit algorithm.

Choose the correct answer.

- B1(4), B2(8), B3(5,1), B4(7), B5(6,1,2) B6(4,2)

Correct!

- B1(4), B2(8), B3(5,1), B4(7), B5(6,1) B6(4,2,2)
- B1(4,5), B2(8,1), B3(7), B4(6,1), B5(4,2,2)
- B1(4,5,1), B2(8,1), B3(7,2), B4(6,4), B5(2)

Question 12

3 / 3 pts

Given a set of items with the following sizes 4, 8, 5, 1, 7, 6, 1, 4, 2, 2 and bins of size 10. The items were packed into the minimum number of bins using the Best fit algorithm.

Choose the correct answer.

- B1(4,5), B2(8,1), B3(7), B4(6,1), B5(4,2,2)
- B1(4), B2(8), B3(5,1), B4(7), B5(6,1,2) B6(4,2)

Correct!

- B1(4,5,1), B2(8,1), B3(7,2), B4(6,4), B5(2)
- B1(4), B2(8), B3(5,1), B4(7), B5(6,1) B6(4,2,2)

Question 13

3 / 3 pts

Given a set of items with the following sizes 4, 8, 5, 1, 7, 6, 1, 4, 2, 2 and bins of size 10. The items were packed into the minimum number of bins using the Worst fit algorithm.

Choose the correct answer.

B1(4), B2(8), B3(5,1), B4(7), B5(6,1,2) B6(4,2)

B1(4), B2(8), B3(5,1), B4(7), B5(6,1) B6(4,2,2)

B1(4,5,1), B2(8,1), B3(7,2), B4(6,4), B5(2)

Correct!

B1(4,5), B2(8,1), B3(7), B4(6,1), B5(4,2,2)

Question 14

12 / 12 pts

The Acme Furniture Company manufactures chairs, dining tables, sofas and ottomans. There are three departments in the company assembly, finishing and upholstery. Each day there are 40 labor hours available in the assembly department, 30 hours in finishing and 30 hours in upholstery. Each chair requires 1 hour for assembly, 1 hour for finishing and 1 hours for upholstery. Each dining table requires 3 hours for assembly, 2 hours for finishing and zero hours of upholstery. Each sofa requires 2 hours of assembly, 1 hour of finishing and 4 hours of upholstery. Each ottoman requires 1 hour for assembly, 1 hour for finishing and 2 hours for upholstery. Due to demand at least 4 chairs must be manufactured for each table manufactured and the number of sofas must be greater than the number of ottoman. The CEO of Acme Furniture would like to maximize total profit using only the available labor. If the profit per chair is $20, per dining table is $200, per sofa $300 and per ottoman $60, how many units of each item should be produced each day.

Formulate your answer and a linear program. Define all variables, give the objective function and list all constraints. **Do not solve**.

Your Answer:

let c = # of chairs; profit = $20 each

let t = # of tables; profit = $200 each

let s = # of sofas; profit = $300 each

let o = # of ottomans; profit = $60 each

Objective: Maximize daily profit --> Find max such this function is maximized: 20c + 200t + 300s + 60o

Constraint 1 - Assembly which has 40 labor hours available

- Chair: 1 hour of Assembly
- Table: 3 hour of Assembly
- Sofa: 2 hour of Assembly
- Ottoman: 1 hour of Assembly

Hence, 1c + 3t + 2s + 1o <= 40

Constraint 2 - Finishing which has 30 labor hours available

- Chair: 1 hour of Finishing
- Table: 2 hour of Finishing
- Sofa: 1 hour of Finishing
- Ottoman: 1 hour of Finishing

Hence, 1c + 2t + 1s + 1o <= 30

Constraint 3 - Upholstery which has 30 labor hours available

- Chair: 1 hour of Upholstery
- Table: 0 hour of Upholstery
- Sofa: 4 hour of Upholstery
- Ottoman: 2 hour of Upholstery

Hence, 1c + 0t + 4s + 2o <= 30

Constraint 4 - there must be 4 chairs manufactured for each 1 table ratio

Hence, 4t <= 1c

Constraint 5 - there must be more sofas manufactured than ottomans

Hence, s > u

Constraint 6 - cannot have a negative number of furniture manufactured, i.e. each value must be greater than or equal to 0

Hence, c >= 0 and t >= 0 and s >= 0 and o >= 0

Question 15

12 / 12 pts

Papa Mario of Mario's Pizzeria has baked a huge pizza and cut it into n slices, but he is clumsy and the pizza wasn't evenly sliced. The n slices have size $s_1,s_2,...,s_n$. There are n hungry students who each want to eat a slice of pizza. Suppose the i^{th} student would be happy with any slice whose size is at least t_i. Give an efficient algorithm to determine whether it is possible to distribute the pizza slices so everyone is happy.

(a) What algorithm paradigm is most appropriate for this problem?

- Divide-and-conquer
- Greedy algorithm
- Dynamic Programming
- Linear Programming
- Graph Algorithm
- None of these

(b) Verbally describe an efficient algorithm to solve this problem.

(c) What is the asymptotic running time of your algorithm?

Your Answer:

A) Greedy algorithm works best here

B) We can sort both i students and t pizza slices from smallest to largest. We can then match the student with the smallest size of pizza to keep him or her happy with the smallest pizza slice. If slice of pizza is not sufficient large enough then move up to next pizza slice size. If we are able to serve all students with this algorithm then everyone is happy. However, if there students left over without associated pizza slices at end then not everyone will be happy.

C) Since we are sorting 2 arrays: one of students and one of pizza slices we can use a fast sorting algorithm like merge sort for each which as a run time of O(n*logn). Doing this twice will result in 2xO(n*logn) which for large values of n has same run time as **O(n*logn)**

Question 16

12 / 12 pts

MalMart Maps wants to add a new feature to their map app: include a stop at MalMart in your route. In particular, they are looking for an algorithm for the following problem:

Input: A **directed** graph G = (V,E), with a positive length L(e) on each edge e; vertices s (start), m (the location of MalMart) and d (destination).

Output: The length of the shortest path from s to d that goes through m.

(a) Verbally describe in detail an efficient algorithm MalStop(G, L, m, s, d) to solve this problem.

(b) What is the asymptotic running time of your algorithm?

Your Answer:

A) Given this is a weighted, directed graph we can use Dijkstra's algorithm which is a special case of the shortest path problem. We start at vertex s (starting point) and implement Dijkstra's algorithm towards m (MalMart location). This is done by examining all the reachable vertices from s and taking the cumulative shortest route. We create 2 arrays, one for visited vertices and one for unvisited. Initially, all nodes are in unvisited array with an infinite distance from s. Once a node is deemed to be reachable, we update value of distance with shortest *cumulative* route from s. We then continue in this fashion until we have reached m, we then navigate from m to d (final destination point) again via Dijkstra's algorithm.

B) Dijkstra's algorithm runs at worst case **O(v^2)** where v is # of vertices on graph . Technically we will have to do this twice, once from s to m and again from m to d so this could be 2*O(v^2), but this is still O(v^2) as number of vertices on graph gets large

Question 17

12 / 12 pts

0-1 Integer Programming is similar to linear programming except the variables can only be 0 and 1. Consider the 0-1 Integer Programming Decision Problem (0-1 IPD) as defined below:

Instance: A set X of 0-1 integer variables ($x_i = 0$ or $x_i = 1$), a set of inequalities over these variables, a function f(x) to maximize and integer K.

Question: Does there exist an assignment of values to X such that all inequalities are true and f(X) >= K?

An example of an instance of 0-1 IP-D: : $x_1 + x_2 + x_3 < 3$, $x_1 - x_3 >= 1$, $f(x) = 4x_1 + x_3$, K = 2. The answer is yes, given certificate {$x_1 = 1$, $x_2 = 0$, $x_3 = 0$}

Prove that 0-1 IPD Problem is NP-Complete.

Your Answer:

To prove this is NP-Complete we must show algorithm is both a member of NP and is member of NP-Hard.

1) NP Membership - Since we have this certificate of {$x_1 = 1$, $x_2 = 0$, $x_3 = 0$} we can clearly demonstrate the 0-1 Integer Programming algorithm can be verified in polynomial time deterministically. Hence, Yes 0-1 Integer Programming algorithm is part of NP.

2) NP Hard Membership - We can see that a classic NP-Hard algorithm like the SAT problem (Boolean satisfiability problem) is less algorithmicly complex than 0-1 Integer Programming. This is because in 0-1 Integer Programming can have many variables (x1, x2, x3, x4, etc.) with many constraints. Therefore since SAT is NP-Hard and 0-1 Integer Programming is more complex than SAT, 0-1 Integer Programming must also be in NP-Hard as well.

Hence, since 0-1 Integer Programming algorithm is both in NP and is NP-Hard it **is proven that it must be NP-Complete.**

Question 18

2 / 12 pts

Let HAM-CYCLE be the decision problem of the Hamiltonian cycle. Suppose HAM-CYCLE ∈ P and HC is a polynomial-time algorithm/oracle that solves HAM-CYCLE.

(a) Design an algorithm that lists the vertices of a Hamiltonian cycle.

(b) Prove the correctness of your algorithm.

(c) Show that the time complexity of your algorithm is polynomial.

Your Answer:

A) For this graph to be a Hamiltonian cycle we have to start and end at same node and visit all nodes exactly once. Algorithm could be start at a random node, then move to next vertex that is reachable then add to array of reachable nodes. If this is indeed a Hamiltonian cycle we should then be able to traverse through all nodes in graph and finish right where started.

B)

C)

Quiz Score: **87** out of 100