This quiz has been regraded; your new score reflects 3 questions that were affected.

Midsem

- Due Oct 9, 2024 at 12:30pm
- Points 100
- Questions 11
- Available Oct 9, 2024 at 11am - Oct 9, 2024 at 12:30pm 1 hour and 30 minutes
- Time Limit None

Instructions

This quiz is no longer available as the course has been concluded.

Attempt History

	Attempt	Time	Score	Regraded
LATEST	Attempt 1	45 minutes	8 out of 100	100 out of 100

Correct answers are hidden.

Score for this quiz: 100 out of 100
Submitted Oct 9, 2024 at 11:46am
This attempt took 45 minutes.

Question 1
10 / 10 pts

The asymptotic upper bounds of six algorithms are given below.

$O\left(10^n\right)$

$O\left(n^2\right)$

$O\left(n^n\right)$

$O\left(\log_2 n\right)$

$O\left(n!\right)$

$O\left(n\right)$

Arrange the algorithms in ascending order of growth rate.

Your Answer:

O(log_2(n)), O(n), O(n^2), O(10^n), O(n!), O(n^n)

⣿

Question 2

10 / 10 pts

(10 points) Given two sorted arrays A and B. The size of each array is n. Write pseudocode of an algorithm that merges A and B into a single sorted array in O(n) time.

Your Answer:

```
sort(A, B):
  i = 0
  j = 0
  result = []
  while i < n and j < n:
    if A[i] < B[j]:
      result.append(A[i])
      i = i + 1
    else:
      result.append(B[j])
      j = j + 1

  while i < n:
    result.append(A[i])
    i = i + 1
  while j < n:
    result.append(B[j])
    j = j + 1

  return result
```

⁞⁞
Question 3
10 / 10 pts

For each of the following two statements, answer Yes or No. If Yes, give a short explanation. If No, give a counterexample.

a. Let G be an arbitrary connected, undirected graph with a distinct cost $c(e)$ on every edge e. Suppose $e*$ is the cheapest edge in G; that is, $c(e*) < c(e)$ for edge $e \neq e*$. Is there a minimum spanning tree T of G that contains the edge $e*$?

b. Suppose we are given an instance of the Minimum Spanning Tree Problem on a graph G, with edge costs that are all positive and distinct. Let T be a minimum spanning tree for this instance. Now suppose we replace each edge cost c_e by its square, $(c_e)^2$, thereby creating a new instance of the problem with the same graph but different costs. Is T still a minimum spanning tree for this new instance?

Your Answer:

a. Yes, in a graph where all edges have distinct weights, the cheapest edge e* must be included in any MST as long as it does not form a cycle. In any connected, undirected graph with distinct edge costs, the cheapest edge can always be added without violating the conditions of a minimum spanning tree. The lightest or cheapest edge must be cut from the tree to form the MST.

b. Yes, squaring all edge weights preserves the order of the edge weights, meaning that if edge a was cheaper than edge b in the original graph, than a^2 cost will still be cheaper than b^2 cost in the new graph. Since constructing an MST only relies on edge weights and the relative ordering has not changed, the same tree T will still be the minimum spanning tree even after squaring the weights.
⁞⁞
Question 4
10 / 10 pts

You are given an integer array cost where cost[i] is the cost of i-th step on a staircase. Once you pay the cost, you can either climb one or two steps. You can either start from the step with index 0, or the step with index 1. Write pseudocode of an algorithm that returns the minimum cost to reach the top of the floor.

Your Answer:

```
minCost(cost):
  n = length(cost)
```

```
if n == 0:
  return 0
if n == 1:
  return cost[0]

dp = array of size n+1
dp[0] = 0
dp[1] - 0

for i from 2 to n:
  dp[i] = min(dp[i-1] + cost[i-1], dp[i-2] + cost[i-2])

return dp[n]
```

Question 5

Original Score: 0 / 4 pts Regraded Score: 4 / 4 pts

This question has been regraded.

Sorting an array of size n requires comparing each element with every other element. Therefore sorting cannot be done in less than $O\left(n^2\right)$ time.

True [✓ False]

Question 6

Original Score: 0 / 4 pts Regraded Score: 4 / 4 pts

This question has been regraded.

An alignment between two genome sequences is when every symbol in one genome is matched to exactly one symbol in the other genome.

True

[✓ False]

Question 7

4 / 4 pts

Dynamic Programming is based on a time-memory tradeoff.

[✓ True]

False

Question 8

4 / 4 pts

Breadth-First Search can be used to find if a graph is bipartite.

[✓ True]

False

Question 9

Original Score: 0 / 4 pts Regraded Score: 4 / 4 pts

This question has been regraded.

There is no known polynomial-time algorithm to predict the optimal secondary structure of single-stranded RNA molecules.

True

[✓ False]

Question 10

20 / 20 pts

You are given an m x n grid where each cell can have one of three values:

0 representing an empty cell, 1 representing a fresh orange, or 2 representing a rotten orange. Every minute, any fresh orange that is adjacent to a rotten orange becomes rotten. Here, adjacent means left, right, up or down (not diagonal).

We want to find how much time we have until all oranges get rotten. Write pseudocode to solve this problem.

If there will always be one or more fresh oranges, return -1.

Your Answer:

```
orangesRotting(grid):
  m = number of rows in grid
  n = number of columns in grid

  queue = new Queue()
  freshOranges = 0

  for i from 0 to m-1:
    for j from 0 to n-1:
      if grid[i][j] == 2:
        queue.enqueue((i,j))
      if grid[i][j] == 1:
        freshOranges += 1

  if freshOranges == 0:
    return 0

  minutesPassed = 0
  directions = [(0,1), (0,-1), (1,0), (-1,0)]

  while queue is not empty:
    size = queue.size()
    for i from 0 to size-1:
      x, y = queue.dequeue()
      for direction in directions:
        newX = x + direction[0]
        newY = y + direction[1]
```

```
          if 0 <= newX < m  and 0 <= newY < n and grid[newX][newY] == 1:
            grid[newX][newY] = 2
            queue.enqueue((newX, newY))
            freshOranges -= 1
    minutesPassed += 1

  if freshOranges > 0:
    return -1

  return minutesPassed
```

Question 11
20 / 20 pts

There is a row of n houses, where each house can be painted one of three colors: red, blue, or green. The cost of painting each house with a certain color is different. You have to paint all the houses such that no two adjacent houses have the same color.

The cost of painting each house with a certain color is represented by an n x 3 cost matrix costs.

For example, costs[0][0] is the cost of painting house 0 with the color red; costs[1][2] is the cost of painting house 1 with color green, and so on.

Write pseudocode of an algorithm that returns the minimum cost to paint all houses.

Your Answer:

```
minCost(costs):
  if costs is empty:
    return 0

  n = number of houses (found from rows in costs array)

  redCost = costs[0][0]
  blueCost = costs[0][1]
```

```
greenCost = costs[0][2]


for i from 1 to n-1:
    newRedCost = costs[i][0] + min(blueCost, greenCost)
    newBlueCost = costs[i][1] + min(redCost, greenCost)
    newGreenCost = costs[i][2] + min(redCost, blueCost)


    redCost = newRedCost
    blueCost = newBlueCost
    greenCost = newGreenCost


return min(redCost, blueCost, greenCost)
```

Quiz Score: 100 out of 100