Summary in Graph

Exam Summary (GO Classes CS Test Series 2025 | Algorithms | Subject Wise Test 1)

Qs. Attempted:	**0** (0 + 0)	**Correct Marks:**	**0** (0 + 0)
Correct Attempts:	**0** (0 + 0)	**Penalty Marks:**	**0** (0 + 0)
Incorrect Attempts:	**0** (0 + 0)	**Resultant Marks:**	**0** (0 + 0)

Total Questions:	**30** (20 + 10)
Total Marks:	**40** (20 + 20)
Exam Duration:	**90 Minutes**
Time Taken:	**0 Minutes**

EXAM RESPONSE | EXAM STATS | FEEDBACK

Technical

Q #1 | Multiple Choice Type | Award: 1 | Penalty: 0.33 | Algorithms

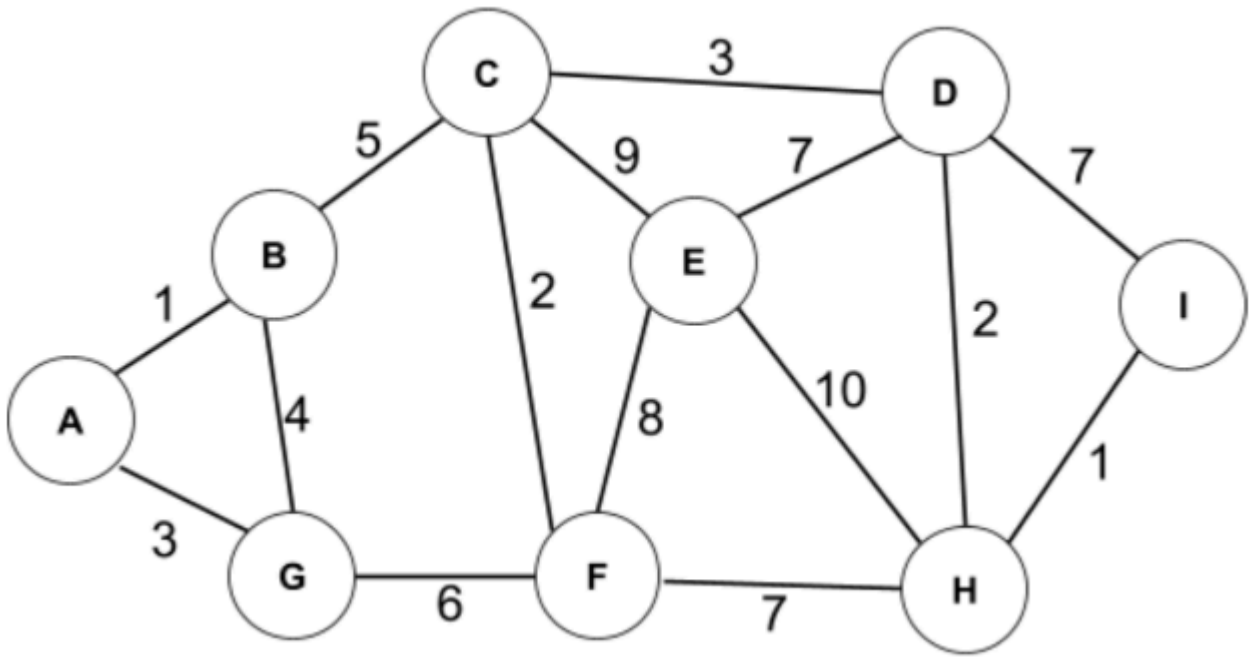
Consider an array A containing n distinct integers, where a local minimum is defined as an element that is smaller than its neighbors (for boundary elements, there is only one neighbor). For instance, in $A = [10, 6, 4, 3, 12, 19, 18]$, the local minima are 3 and 18. Given that finding the absolute minimum requires $\Omega(n)$ time, what is the tightest bound on the time complexity for an efficient algorithm to find a local minimum?

A. $\mathrm{O(n)}$
B. $O(\log n)$
C. $O(n \log n)$
D. $O(\sqrt{n})$

Your Answer: | Correct Answer: B | Not Attempted | Time taken: 00min 00sec | Discuss

Q #2 | Multiple Select Type | Award: 1 | Penalty: 0 | Algorithms

Consider the following graph.



Which of the following edges is/are NEVER the part of any minimum spanning tree?

A. (B, C)
B. (D, E)
C. (E, F)
D. (B, G)

Your Answer: | **Correct Answer: C;D** | Not Attempted | Time taken: 00min 00sec | Discuss

Q #3 Multiple Choice Type | Award: 1 | Penalty: 0.33 | Algorithms

You are working on an embedded device (an ATM) that only has 4KB (4,096 bytes) of free memory, and you wish to sort the 2,000,000 transactions withdrawal history by the amount of money withdrawn (discarding the original order of transactions).

A. Insertion Sort
B. Merge Sort
C. Counting Sort
D. Heap Sort

Your Answer: | **Correct Answer: D** | Not Attempted | Time taken: 00min 00sec | Discuss

Q #4 Multiple Select Type | Award: 1 | Penalty: 0 | Algorithms

Determine which of the following statements is/are TRUE regarding asymptotic notation:

A. If $f(n) = \frac{(n+1)\cdot n}{2}$, then $f(n) \in O\left(n^2\right)$.
B. If $f(n) = \frac{(n+1)\cdot n}{2}$, then $f(n) \in \Theta\left(n^2\right)$.
C. If $f(n) = \frac{(n+1)\cdot n}{2}$, then $f(n) \in \Theta\left(n^3\right)$.
D. $n^{1.1} \in O\left(n(\log n)^2\right)$.

Your Answer: | **Correct Answer: A;B** | Not Attempted | Time taken: 00min 00sec | Discuss

Q #5 Multiple Choice Type | Award: 1 | Penalty: 0.33 | Algorithms

We are given a set of n lists with sizes $n_1, n_2, \ldots, n_n$. A merge operation takes two lists as input and requires $\ell + r$ operations if the lists are of sizes ℓ and r. What strategy should we use to merge all lists into a single list using pairwise merges while minimizing the total number of operations?

A. Merge the two smallest lists at each step, similar to building a Huffman tree based on list sizes as frequencies.
B. Merge the largest two lists at each step to reduce the number of operations needed.
C. Merge lists in any arbitrary order, as the total number of operations will remain constant regardless of order.
D. Start with the largest list and merge it with each other list consecutively.

Your Answer: | **Correct Answer: A** | Not Attempted | Time taken: 00min 00sec | Discuss

Q #6 Multiple Select Type | Award: 1 | Penalty: 0 | Algorithms

Which of the following sorting algorithms never compare the same pair of elements more than once? Select all that apply.

A. Insertion Sort
B. In-place Heap Sort
C. Merge Sort
D. In-place Quick Sort

Your Answer: | **Correct Answer: A;C;D** | Not Attempted | Time taken: 00min 00sec | Discuss

Q #7 Multiple Choice Type | Award: 1 | Penalty: 0.33 | Algorithms

Consider the recurrence relation:

$$T(n) = T(n/3) + T(n/6) + n^{\sqrt{\log n}}$$

Assuming $T(n) = 1$ for n smaller than some constant c, what is the solution to this recurrence?

A. $T(n) = \Theta\left(n^{\sqrt{\log n}}\right)$
B. $T(n) = \Theta(n\sqrt{\sqrt{\log n}})$
C. $T(n) = O(n\sqrt{\log n})$
D. $T(n) = \Theta(n \log n)$

Your Answer: | **Correct Answer: A** | Not Attempted | Time taken: 00min 00sec | Discuss

Q #8 Multiple Choice Type | Award: 1 | Penalty: 0.33 | Algorithms

Given the following alphabet and set of frequencies:

$\{(a, 0.12), (b, 0.38), (c, 0.1), (e, 0.25), (f, 0.06), (d, 0.05), (g, 0.01), (h, 0.03)\}$, find the Huffman codes for the symbols g and h after constructing the Huffman tree.

When building your Huffman tree, place the branch of lower weight on the left. A left branch corresponds to a 0 in the codeword, while a right branch corresponds to a 1.

A. g: 110100, h: 110101
B. g: 110000, h: 110001

C. g: 111000, h: 111001
D. g: 110010, h: 110011

Your Answer: | Correct Answer: A | Not Attempted | Time taken: 00min 00sec | Discuss

Q #9 | Multiple Choice Type | Award: 1 | Penalty: 0.33 | Algorithms

Consider the matrix-chain multiply problem for a subchain of matrices : $A_iA_{i+1}\dots A_j$. We want to parenthesize the chain to get the minimum number of scalar multiplications possible (call this operation Scal). We decide to split the subchain at location k, and parenthesize it there. $\text{Scal}(i,j)$ is the therefore the number of scalar multiplications to compute the subproducts $A_{i\dots k}$ and $A_{k+1\dots j}$, plus the cost to multiply these together.

The dimensions of every A_i are $d_{i-1} \times d_i$, so the cost of multiplying these 2 matrices is $d_{i-1} \times d_k \times d_j$. What is the recurrence relation for the $\text{Scal}()$ operation?

1. $\text{Scal}(i,j) = \begin{cases} 0 & i = 0 \\ \min\left(\text{Scal}(i,k) + \text{Scal}(k+1,j) + d_{i-1}d_kd_j \right. & i < j \end{cases}$
2. $\text{Scal}(i,j) = \text{Scal}\big(i, k + \text{Scal}(k+1,j) + d_{i-1}d_kd_j$
3. $\text{Scal}(i,j) = \begin{cases} 0 & i = j \\ \min\limits_{i \le k < j} \left(\text{Scal}(i,k) + \text{Scal}(k+1,j) + d_{i-1}d_kd_j \right. & i < j \end{cases}$

A. 1
B. 2
C. 3
D. None

Your Answer: | Correct Answer: C | Not Attempted | Time taken: 00min 00sec | Discuss

Q #10 | Numerical Type | Award: 1 | Penalty: 0 | Algorithms

Suppose we have an undirected, connected, weighted graph with n vertices such that:

- Every weight is an integer greater than zero.
- No two edges have the same weight.

What is the lowest cost a minimum spanning tree (MST) could have for a graph with 20 nodes that meets the description above?

Your Answer: | Correct Answer: 190 | Not Attempted | Time taken: 00min 00sec | Discuss

Q #11 | Multiple Select Type | Award: 1 | Penalty: 0 | Algorithms

Suppose we have a set of n values. There are always at least one negative value and at least one positive value in the set.
What is the worst case time complexity to find FIRST (i.e. first value from left) negative value of set?

A. If values are stored in an array and arranged in ascending order then it takes $\theta(n)$ in worst case.
B. If values are stored in an array and arranged in descending order then it takes $\theta(n)$ in worst case.
C. If values are stored in a linked list (only head is known)and arranged in ascending order then it takes $\theta(n)$ in worst case.

D. If values are stored in a linked list (only head is known) and arranged in descending order then it takes $\theta(n)$ in worst case.

Your Answer: | Correct Answer: D | Not Attempted | Time taken: 00min 00sec | Discuss

Q #12 | Multiple Select Type | Award: 1 | Penalty: 0 | Algorithms

Let $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ be a simple undirected graph, and s be a particular vertex in it called the source. For $x \in \mathrm{V}$, let $d(x)$ denote the shortest distance in G from s to x. A breadth-first search (BFS) is performed starting at s.

Which of the following is/are ALWAYS true?

A. There are no back edges
B. There are no forward edges
C. For each tree edge (u, v), we have $d[v] = d[u] + 1$
D. For each cross edge (u, v), we have $d[v] = d[u] + 1$

Your Answer: | Correct Answer: A;B;C | Not Attempted | Time taken: 00min 00sec | Discuss

Q #13 | Multiple Select Type | Award: 1 | Penalty: 0 | Algorithms

Given two strings $\mathrm{A}[1, \ldots, m]$ and $\mathrm{B}[1, \ldots, n]$, a **common supersequence** C is a string that has both A and B as subsequences. For example, two possible common supersequences of

$$\mathrm{A} = \text{“apple”} \text{ and } B = \text{“pear”}$$

are

$$\mathrm{C}_1 = \text{“pearapple”} \text{ and } \mathrm{C}_2 = \text{“applear”}.$$

The following is a backtracking algorithm for finding the length of the **shortest** common supersequence of A and B :

```
SCS(A[1,...,m], B[1,...,n], i, j):
if i = 0
        return j
else if j = 0
        return i
else if A[i] = B[j]
        return ____X____
else
        return ____Y____
```

What should be replaced by X and Y such that the algorithm works correctly?

A. X is $\mathrm{SCS}(i-1, j-1) + 1$
B. X is $\mathrm{SCS}(i-1, j-1) + 2$
C. Y is $\min(\mathrm{SCS}(i, j-1), \mathrm{SCS}(i-1, j))$
D. Y is $\min(\mathrm{SCS}(i, j-1), \mathrm{SCS}(i-1, j)) + 2$

Your Answer: | Correct Answer: A | Not Attempted | Time taken: 00min 00sec | Discuss

Q #14 | Multiple Choice Type | Award: 1 | Penalty: 0.33 | Algorithms

Consider an array of 2n elements in the form $1, 2n-1, 2, 2n-2, 3, 2n-3, 4, 2n-4, \ldots, n, n.$

For example, here is the array when $\mathrm{n} = 8$:
$1, 15, 2, 14, 3, 13, 4, 12, 5, 11, 6, 10, 7, 9, 8, 8$

How many comparisons does the Insertion sort algorithm make as a function of n in the worst case?

A. $\sim \frac{1}{4}n^2$
B. $\sim \frac{1}{2}n^2$
C. $\sim n^2$
D. $\sim 2n^2$

Your Answer: | **Correct Answer: C** | Not Attempted | Time taken: 00min 00sec | Discuss

Q #15 Multiple Choice Type | Award: 1 | Penalty: 0.33 | Algorithms

Let n be a power of 2 . Consider an array structured as shown below.

$$2n, 2n, \ldots . 2n, 0, 1, 2, 2, 4, 4, 4, 4, 8, 8, 8, 8, 8, 8, 8, 8, \ldots . n, n, \ldots . n$$

The first half of the array contains $2n$ repetitions of the value $2n$. The second half contains $2n$ numbers: one number 0 , one number 1 , two repetitions of the number 2 , four repetitions of the number 4 , eight repetitions of the number 8 , sixteen repetitions of the number 16 , and so on, until the number n appears n times.
For example, here is the array when $n = 4$:

$$8, 8, 8, 8, 8, 8, 8, 8, 0, 1, 2, 2, 4, 4, 4, 4$$

How many comparisons does the merge sort algorithm make as a function of n in the worst case?

A. $\sim n \log n$
B. $\sim 2n \log n$
C. $\sim 4n \log n$
D. $\sim \frac{9}{2}n \log n$

Your Answer: | **Correct Answer: B** | Not Attempted | Time taken: 00min 00sec | Discuss

Q #16 Multiple Select Type | Award: 1 | Penalty: 0 | Algorithms

The following shows a series of steps in order at certain points in a sorting algorithm, although there may be addi-
tional steps omitted in between. Which sorting algorithm(s) could be being run here? Select all that apply.

Initial Array:
10, 5, 8, 3, 4, 2, 9, 1, 6, 7
Some Later Step:
5, 6, 7, 10, 9, 8, 4, 3, 2, 1
Final Step:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10

A. Insertion Sort
B. In-place Heap Sort
C. Merge Sort
D. In-place Quick Sort

Your Answer: | **Correct Answer: B** | Not Attempted | Time taken: 00min 00sec | Discuss

Q #17 Multiple Choice Type | Award: 1 | Penalty: 0.33 | Algorithms

Suppose we choose the median of five items as the pivot in quicksort. If we have an N element array, then we find the median of the elements located at the following positions: left (=0), right ($= N - 1$), center (the average of left and right, rounded down), leftOfCenter (the average of left and center, rounded down), and rightOfCenter (the average of right and center, rounded down). The median of these elements is the pivot. What is the worst case running time of this version of quicksort?

A. $\mathrm{O}(\mathrm{n} \log n)$
B. $\mathrm{O}(\mathrm{n}^2)$
C. $\mathrm{O}(\mathrm{n}^2 \log n)$
D. $\mathrm{O}(\mathrm{n})$

Your Answer: | **Correct Answer: B** | Not Attempted | Time taken: 00min 00sec | Discuss

Q #18 Multiple Choice Type | Award: 1 | Penalty: 0.33 | Algorithms

The preorder and postorder traversal of a full binary tree is given as:
Preorder: C,T,U,W,X,S,A,Z,O
Postorder: W,X,U,S,T,Z,O,A,C
What is the inorder traversal of this tree?

A. $W, U, X, T, S, C, Z, A, O$
B. $W, U, X, T, S, C, Z, O, A$
C. $W, U, X, T, S, Z, C, A, O$
D. Cannot Determine

Your Answer: | **Correct Answer: A** | Not Attempted | Time taken: 00min 00sec | Discuss

Q #19 Multiple Select Type | Award: 1 | Penalty: 0 | Algorithms

Which of the following statements is/are FALSE regarding 0/1 Knapsack to maximize the value of the items in a Knapsack of weight w?

A. The optimal solution cannot always be obtained using a greedy algorithm.
B. A smaller capacity Knapsack is an optimal sub-structure to solve this problem if using DP.
C. There is a DP solution for this algorithm that can be easily modified to return the contents of the knapsack along with the value of the knapsack.
D. Bottom-up algorithms are usually recursive whereas top-down algorithms are usually iterative.

Your Answer: | **Correct Answer: B;D** | Not Attempted | Time taken: 00min 00sec | Discuss

Q #20 Multiple Choice Type | Award: 1 | Penalty: 0.33 | Algorithms

Suppose we're given an array of n distinct elements, which was sorted initially but then was rotated by some unknown number of positions. For example, the given array was rotated 3 positions to the left (or we can 4 say positions to the right). But the number of times these rotations have been done is not known to us.

40	50	60	70	10	20	30

Now, what is the time complexity of the most efficient algorithm that returns the largest element of the array?

A. $\mathrm{O}(\mathrm{n})$
B. $\mathrm{O}(\mathrm{n} \log n)$

C. $\mathrm{O}(\log n)$
D. $\mathrm{O}((\log n)^2)$

Your Answer: | Correct Answer: C | Not Attempted | Time taken: 00min 00sec | Discuss

Q #21 | Multiple Choice Type | Award: 2 | Penalty: 0.67 | Algorithms

The following algorithm takes an array $A[i..j]$ of numbers (where $i \leq j$) and returns a pair of numbers ($\min, \max$) where min is the smallest value in $A[i..j]$ and max is the largest value in $A[i..j]$. Each comparison in the algorithm is in a box.

```
function minMax(A[i..j])
  if j ≤ i+1 then    // So, n ≤ 2
    if [A[i] ≤ A[j]] then return (A[i],A[j]) else return (A[j],A[i])
  else    // So, n > 2.
    k ← i + ⌊(j−i)/2⌋
    (min1, max1) ← minMax(A[i..k])
    (min2, max2) ← minMax(A[k+1..j])
    if [min1 ≤ min2] then min ← min1 else min ← min2
    if [max1 ≥ max2] then max ← max1 else max ← max2
    return (min, max)
```

$T(n) =$ the total number of comparisons done in running minMax on array of size n, where $n \geq 1$.
Which of the following is a recurrence for $T(\bar{n})$?

A. $t(1) = t(2) = 1$ and $t(n) = 2t(n/2) + 2$ when $n > 2$.
B. $t(1) = t(2) = 3$ and $t(n) = 2t(n/2) + 3$ when $n > 2$.
C. $t(1) = t(2) = 1$ and $t(n) = 2t(n/3) + 3$ when $n > 2$.
D. $t(1) = t(2) = 1$ and $t(n) = 2t(n/2) + 3$ when $n > 2$.

Your Answer: | Correct Answer: A | Not Attempted | Time taken: 00min 00sec | Discuss

Q #22 | Multiple Select Type | Award: 2 | Penalty: 0 | Algorithms

Which of the following statements is/are TRUE?

A. Every directed acyclic graph (DAG) has exactly one topological ordering.
B. If a DFS in an undirected graph G contains exactly one back edge, then G can be made acyclic by removing one edge.
C. If G is a directed graph that can be made acyclic by removing exactly one edge, then any DFS in G will contain exactly one back edge.
D. Given a valid start-time numbering of a directed graph, there is only one corresponding finish-time numbering.

Your Answer: | Correct Answer: B;D | Not Attempted | Time taken: 00min 00sec | Discuss

Q #23 | Multiple Choice Type | Award: 2 | Penalty: 0.67 | Algorithms

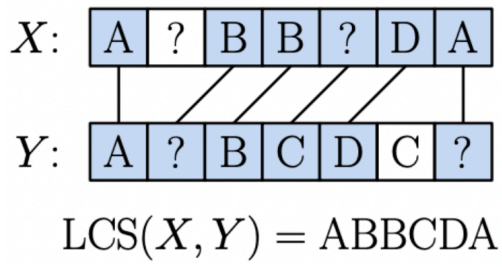
Suppose you are given an array $A[1 \ldots n]$ of sorted integers that has been circularly shifted k positions to the right. For example, $[35, 42, -5, 15, 27, 29]$ is a sorted array shifted $k = 2$ positions.
Assuming all integers are distinct, which of the following $O(\log n)$ algorithms correctly finds the largest element in A ?

A. Use binary search, comparing $A[0]$ with $A[\text{mid}]$. If $A[0] < A[\text{mid}]$, the max is in the right half; $A[0] > A[\text{mid}]$, the max is in the left half.
B. Compare $A[\text{last}]$ with $A[\text{mid}]$. If $A[\text{last}] < A[\text{mid}]$, search the right half; $A[\text{last}] > A[\text{mid}]$, search the left half.
C. Compare $A[\lceil n/4 \rceil]$ with $A[\lceil 3n/4 \rceil]$. If the left is greater, search between them; otherwise, search the outer quarters of the array.
D. Compare adjacent elements to find the larger one, then search in the direction of this larger element.

Your Answer: | **Correct Answer: A;C** | Not Attempted | Time taken: 00min 00sec | Discuss

Q #24 Multiple Select Type | Award: 2 | Penalty: 0 | Algorithms

Consider a variant of Longest common subsequence (LCS) named '$\text{LCS with wild character}$ "?"
It is the same as LCS except for one additional feature that a character "?" can match any other character but not with "?".



$$\text{LCS}(X, Y) = \text{ABBCDA}$$

The diagram above depicts that "?" can match with any character that we want but not with "?".

Which of the following is/are the correct recurrence relation for such "LCS ?" Where $\text{lcs}(i, j)$ represents LCS with wild character in $\text{X}[1 \ldots i]$ and $\text{Y}[1 \ldots j]$.

A. $$\text{lcs}(i,j) = \begin{cases} \text{lcs}(i-1, j-1) + 1 & \begin{cases} \text{if } x_i = y_j \neq \text{“?”} \\ \text{or either } x_i \text{ or } y_j \text{ (but not both) equals “?”} \end{cases} \\ \max(\text{lcs}(i-1, j), \text{lcs}(i, j-1)) & \text{otherwise.} \end{cases}$$

B. $$\text{lcs}(i,j) = \begin{cases} \text{lcs}(i-1, j-1) + 1 & \text{if } x_i = y_j \neq \text{“?”} \\ \max(\text{lcs}(i-1, j), \text{lcs}(i, j-1)) & \text{if } x_i \neq y_j \end{cases}$$

C. $$\text{lcs}(i,j) = \begin{cases} \text{lcs}(i-1, j-1) + 1 & \text{if } x_i = y_j \neq \text{“?”} \\ \max(\text{lcs}(i-1, j), \text{lcs}(i, j-1)) & \text{if } x_i = y_j = \text{“?”} \end{cases}$$

D. $$\text{lcs}(i,j) = \begin{cases} \text{lcs}(i-1, j-1) + 1 & x_i = y_j \neq \text{“?”} \\ \max(\text{lcs}(i-1, j), \text{lcs}(i, j-1)) & \begin{cases} x_i = y_j = \text{“?”} \\ \text{or} \\ x_i \neq y_j \end{cases} \end{cases}$$

Your Answer: | **Correct Answer: A** | Not Attempted | Time taken: 00min 00sec | Discuss

Q #25 Multiple Choice Type | Award: 2 | Penalty: 0.67 | Algorithms

A hiker faces the $0/1$ Knapsack problem. There are 7 items to be packed into the knapsack, each with value v_i and weight w_i as shown in the following table.

i	1	2	3	4	5	6	7
v_i	3	6	8	1	2	5	7
w_i	7	3	5	1	4	2	6

The knapsack, which is initially empty, can hold a maximum weight of 24, so some item(s) must be left behind, and fractions of items cannot be packed. The optimality criterion is to maximize the total value of the items that are placed in the knapsack. The hiker fills the knapsack one item at a time, using a heuristic algorithm that is greedy on value density, where the value density of an item is its value/weight ratio. When this heuristic algorithm is used, what is the total value of the items that are packed, and is this total optimal?

	Total Value	Optimal/Not Optimal
A.	29	Optimal
B.	29	Not Optimal
C.	30	Optimal
D.	30	Not Optimal

Your Answer: | Correct Answer: B | Not Attempted | Time taken: 00min 00sec | Discuss

Q #26 | Multiple Select Type | Award: 2 | Penalty: 0 | Algorithms

Let $f(n)$ and $g(n)$ be two positive increasing functions.

Which of the following option is/are TRUE?

A. $f(n) + g(n) = \Theta(\max(f(n), g(n)))$
B. $f(n) + o(f(n)) = \Theta(f(n))$. o is small-oh
C. $f(n) + g(n) = \Theta(\min(f(n), g(n)))$
D. $f(n) = \Theta(f(n/2))$

Your Answer: | Correct Answer: A;B | Not Attempted | Time taken: 00min 00sec | Discuss

Q #27 | Multiple Select Type | Award: 2 | Penalty: 0 | Algorithms

Which of the following is/are TRUE?

A. Dijkstra's algorithm may not terminate if the graph contains negative-weight edges.
B. Given a graph $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ with positive edge weights, the Bellman-Ford algorithm and Dijkstra's algorithm can produce different shortest-path trees despite always producing the same shortest-path weights.
C. The Bellman-Ford algorithm applies to instances of the single-source shortest path problem which do not have a negative-weight directed cycle, but it does not detect the existence of a negative-weight directed cycle if there is one.
D. On a connected, directed graph with only positive edge weights, Bellman-Ford runs asymptotically as fast as Dijkstra.

Your Answer: | Correct Answer: B | Not Attempted | Time taken: 00min 03sec | Discuss

Q #28 | Multiple Choice Type | Award: 2 | Penalty: 0.67 | Algorithms

What will be the tightest bound on the worst-case time complexity of the following function $\mathbf{sun}()$?

```
    void sun(int n) {
        int k = 0;
        for(int i = 1; i < n; i *= 2) {
            k++;
 5.     }
        for(int i = 0; i < n; i++) {
            if(k % 5 == 0) {
                k--;
            }
10.         else {
                for(int j = 0; j < k; j++) {
                    printf("let's go to the beach");
                }
            }
15.     }
    }
```

A. $O(n \log n)$
B. $O(\log n)$
C. $O\left(n^2\right)$
D. $O\left(n^3\right)$

Your Answer: | **Correct Answer: A** | Not Attempted | Time taken: 00min 00sec | Discuss

Q #29 | Multiple Choice Type | Award: 2 | Penalty: 0.67 | Algorithms

Let d[] represent the discovery time (first visit) and f[] represent the finish time (last visit) for each vertex in a depth-first search (DFS) of a directed graph G. Therefore, each vertex u has an interval (d[u], f[u]).
Suppose r is root of some DFS tree, and its incident edges are removed from G to form a new graph G'.
Without access to G', you only have the original d[] and $f[]$ arrays.
How would you adjust d[] and f[] to make them valid for G' based on the new DFS order?

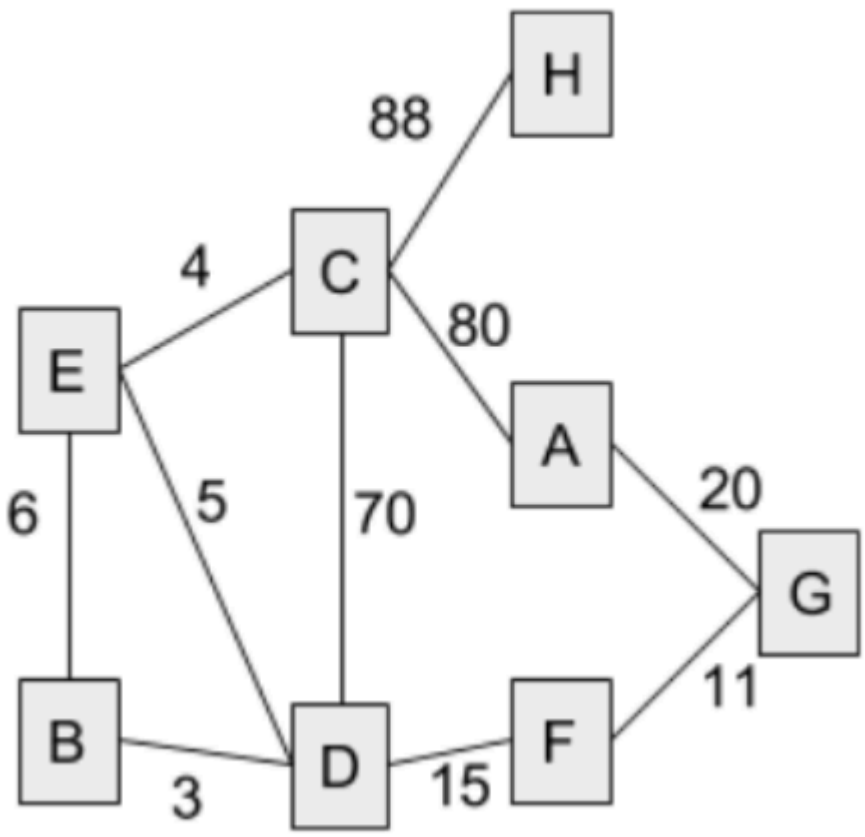
A. Subtract one from the d[] and $f[\,]$ values for any vertex whose intervals are in [d[r], $f[r]$], and subtract 2 from $d[\,]$ and $f[\,]$ for any vertex v where $d[v] > f[r]$. Leave the others unchanged.
B. Set $d[r]$ and $f[r]$ to zero, then add 1 to d[] and f[] for vertices with intervals in [d[r], $f[r]$].
C. Adjust d[ ] and f[ ] by adding 1 to all values.
D. Subtract one from the d[] and f[] values for all vertices.

Your Answer: | **Correct Answer: A** | Not Attempted | Time taken: 00min 00sec | Discuss

Q #30 | Multiple Choice Type | Award: 2 | Penalty: 0.67 | Algorithms

Consider the following weighted undirected graph:
Suppose we run Kruskal's Algorithm on the above graph and the algorithm has reached completion, but now we want to add another edge to the graph of integer weight w. Match the weights in Column A to options in

Column B such that adding an edge of that weight to the graph will, might, or will not change the edges in a MST.



Column A	Column B
A. w < 4	I. Will Change
B. 5 < w < 7	II. Might Change
C. 20 < w	III. Will not change
D. 90 < w	

A. A-I , B-II, C-II, D-III
B. A-III , B-II, C-II, D-I
C. A-III, B-I, C-II, D-II
D. A-I, B-II, C-III, D-III

Your Answer: **Correct Answer: A** Not Attempted Time taken: 00min 00sec Discuss