

Select the right rank of the following functions in order of growth. That is, find an arrangement  $f_1, f_2, \dots, f_5$  satisfying  $f_1 = O(f_2), f_2 = O(f_3), \dots$  and so forth.

[2 points]

a)  $17 \leq \lg(n) \leq \lg(\lg(n)) \leq \lg^2(n) \leq 1/(3^n)$

b)  $17 \leq \lg(\lg(n)) \leq \lg(n) \leq \lg^2(n) \leq 1/(3^n)$

c)  $17 \leq \lg(\lg(n)) \leq \lg(n) \leq 1/(3^n) \leq \lg^2(n)$

d)  $1/(3^n) \leq 17 \leq \lg(\lg(n)) \leq \lg(n) \leq \lg^2(n)$

An upper bound cost to find the largest element in a min-heap of length  $n$  is: [2 points]

a)  $O(\lg n)$

b)  $O(1)$

c)  $O(n \lg n)$

d)  $O(n)$

Suppose that a **Depth First Search** on a *directed graph* yields a path of **tree edges** from vertex  $u$  to vertex  $v$ . If there is also an **edge** from  $u$  to  $v$ , then the edge  $(u, v)$  is: [2 points]

a) tree edge

b) back edge

c) forward edge

d) cross edge

The number of possible minimal spanning trees of a graph is:

[2 points]

a) 0

b) 1

c) 0 or 1

d) 1 or more

If a value of an entry in a max-heap is **decreased**, how long does it take to fix the heap property in the *worst case*? [2 points]

a)  $O(n \lg n)$

b)  $O(n)$

c)  $O(\lg n)$

d)  $O(n^2)$

Consider the following algorithm:

Algorithm **PrimalityTest**

**Input:** A positive integer  $n \geq 2$

**Output:** true if  $n$  is prime and false otherwise

1.  $k = \lfloor n^{0.5} \rfloor$
2. **for**  $j = 2$  **to**  $k$
3.   **if**  $j$  divides  $n$  **then return** false
4. **end for**
5. **return** true

Let  $T(n)$  be the time complexity function that represents how many times “**if**  $j$  divides  $n$ ” is executed. Select the right bound  $O$ -function for  $T(n)$ .    [2 points]

a)  $O(n)$

b)  $O(n^{0.5})$

c)  $O(1)$

d)  $O(n^{1.5})$

Suppose we have a quicksort algorithm that at each stage divides the array into two parts, one of size  $an$  and the other of size  $(1 - a)n$ , where  $0 < a \leq 1/2$  is a constant. It can find the pivot to accomplish this split, and perform the split in time exactly equal to  $n$ . The right recurrence relationship describing the run time of this version of quicksort is: **[2 points]**

a)  $T(n) = T(an) + T((1-a)n)$

b)  $T(n) = T(an) + T(n) + n$

c)  $T(n) = T(an) + T((1-a)n) + n$

d)  $T(n) = T(an) + n$

If an undirected graph has *at least* as many edges as vertices ( $|E| \geq |V|$ ), then the graph [2 points]

- a) contains exactly one cycle.
- b) contains at most one cycle.
- c) contains at least one cycle.
- d) is acyclic (has no cycles).

How long does it take to determine if a *given vertex* has *zero degree* in an undirected graph  $G = (V, E)$ ?      **[2 points]**

- a)  $O(V + E)$  for adjacency matrix and  $O(V)$  for adjacency lists.
- b)  $O(V)$  for adjacency matrix and  $O(1)$  for adjacency lists.**
- c)  $O(E)$  for both adjacency matrix and adjacency lists.
- d)  $O(V)$  for both adjacency matrix and adjacency lists.

Using **Breadth First Search** in an **undirected** graph, how can you recognize a **cycle**?

**Rephrase it.** [2 points]

- a) If you find an adjacent vertex with gray or black colors that has already been visited.
- b) If you find an adjacent vertex with black color that has already been visited.
- c) If you find an adjacent vertex with white color that has already been visited.
- d) If you find an adjacent vertex that has already been visited and it's not your parent.

Which of the following is **NOT** a step in the main loop of Kruskal's algorithm? [2 points]

- a) Selects a remaining edge  $(u, v)$  of lowest cost.
- b) Identifies the component(s) to which  $u$  and  $v$  belong.
- c) Adds  $(u, v)$  if it does not create a cycle.
- d) Updates the costs of edges adjacent from  $u$  or  $v$ .

The solution to the following recurrence:

[2 points]

$$T(n) = 1, \quad n < 2$$

$$T(n) = 2T(n/4) + n^2, \quad n \geq 2$$

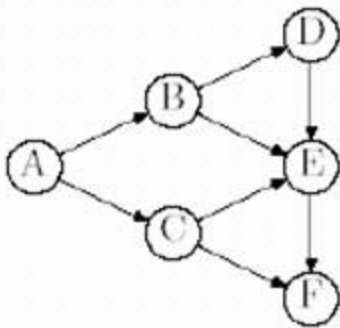
- a) is a simple application of the Master Theorem resulting in  $O(n^2)$
- b) is a simple application of the Master Theorem resulting in  $O(n \log n)$
- c) doesn't fit the Master Theorem but it is  $O(n^2)$
- d) doesn't fit the Master Theorem but it is  $O(n \log n)$

When finding the strongly connected components, the number of components is indicated by:

**[2 points]**

- a) The number of back edges found during the first depth-first search.
- b) The number of cross edges found during the second depth-first search.
- c) The number of restarts for the first depth-first search.
- d) The number of restarts for the second depth-first search.

Which of the orderings below is **NOT** a possible order in which **breadth-first search** would visit the vertices of the graph shown below? [2 points]



a) ACBFED

**b) ABCDEF**

c) ABCEFD

d) None of the above

The worst-case running time of the following code fragment is: [2 points]

```
i = 1
while (i <= n) {
    print i
    i = i * 2 }
```

a)  $O(n)$

b)  $O(1)$

c)  $O(n^3)$

d) None of the above

One of the following statements is true: [2 points]

- a) Insertion Sort algorithm requires memory space more than Merge Sort.
- b) Both Insertion Sort and Merge Sort use Divide-and-Conquer Technique.
- c) Merge Sort uses Divide-and-Conquer Technique with run time  $O(n \log n)$ .
- d) Merge Sort uses Divide-and-Conquer Technique but does not require the combine step.

One of the statements below is **true**:

[2 points]

- a) If the depth-first search of a graph  $G$  yields no cross edges, then the graph  $G$  is acyclic.
- b) Depth-first search of a graph is asymptotically faster than breadth-first search.
- c) At Depth First Search (DFS) algorithm, the DFS-Visit algorithm is called  $N$  times, where  $N$  = the number of trees DFS algorithm generates.
- d) None of the above.

Given the polynomial:  $p(x) = a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \dots + a_0$

Suppose the following algorithm is used:

```
p = a0
xpower = 1
for (i=1; i<=n ; i++)
    xpower = x * xpower
    p = p + ai * xpower
```

How many **multiplications** are done in the worst case?

[2 points]

a)  $O(1)$

b)  $O(n)$

c)  $O(n \log n)$

d)  $O(n^2)$

One of the statements below is **not** true: [2 points]

- a) Let  $G = (V, E)$  be a weighted graph and let  $M$  be a minimum spanning tree of  $G$ . The path in  $M$  between any pair of vertices  $v_1$  and  $v_2$  may be a shortest path in  $G$ .
- b) If  $T$  is a minimum spanning tree for graph  $G$ , then the maximum-weight edge in  $G$  must not be in  $T$ .
- c) A light edge is a cross edge with minimum weight.
- d) None of the above.

One of the following statements is **true**:

ComputerJU.com

[2 points]

- a) For any directed acyclic graph, there is only one topological sort (ordering) of the vertices.
- b) For Topological sort, a graph must be Strongly Connected.
- c) For any directed acyclic graph, there is may be one or more topological sort (ordering) of the vertices.
- d) None of the above.

One of the following statements is **not** true:

[2 points]

- a) Prim's algorithm is a vertex based algorithm.
- b) Kruskal's algorithm is a vertex based algorithm.
- c) The run time of Prim's algorithm is  $O(E \lg V)$ .
- d) Prim's algorithm uses Build Heap function.

A longest path in a directed graph  $G = (V, E)$  can be found in:

[2 points]

- a)  $O(V + E)$  time.
- b)  $O(V)$  time.
- c)  $O(E)$  time.
- d)  $O(E \log V)$  time.

The Breadth-First-Search algorithm makes use of

[2 points]

- a) a stack.
- b) a queue.
- c) an array.
- d) a binary tree.

In Depth-First-Search, what is the structure of gray nodes?

[2 points]

- a) A Star Graph.
- b) A Linear Chain.
- c) A Complete Graph.
- d) A Forest.

One advantage of Quick Sort is: [2 points]

- a) It sorts in place, which means does not require additional memory space other than the input array.
- b) It does preserve the relative order of elements with equal keys.
- c) The worst case is  $O(n \log n)$ .
- d) None of the above.