Sample Questions for practice

1) How to draw a weighted undirected graph with exactly 3 nodes that has exactly 0 minimum spanning trees?
2) How to draw a weighted undirected graph with exactly 3 nodes that has exactly 2 minimum spanning trees?
3) How to draw a weighted undirected graph with exactly 3 nodes that has exactly 3 minimum spanning trees?
4) What is the worst-case asymptotic running time of heap-sort?
5) What is the worst-case asymptotic running time of merge-sort?
6) What is the worst-case asymptotic running time of quick-sort?
7) What is the asymptotic running time of quick-sort if the array is already sorted (or almost sorted) and the pivot-selection strategy picks the leftmost element in the range-to-be-sorted?
8) What is the asymptotic running time of quick-sort if the array is already sorted (or almost sorted) and the pivot-selection strategy picks the rightmost element in the range-to-be-sorted?
9) What is the asymptotic running time of quick-sort if the array is already sorted (or almost sorted) and the pivot-selection strategy picks the middle element in the range-to-be-sorted?
10) For each function on the left, give the best matching order of growth of the running time on the right.

__B__

```
public static int f1(int N) {
    int x = 0;
    for (int i = 0; i < N; i++)
        x++;
    return x;
}
```

_____

```
public static int f2(int N) {
   int x = 0;
   for (int i = 0; i < N; i++)
      for (int j = 0; j < i; j++)
         x++;
   return x;
}
```

_____

```
public static int f3(int N) {
   if (N == 0) return 1;
   int x = 0;
   for (int i = 0; i < N; i++)
      x += f3(N-1);
   return x;
}
```

_____

```
public static int f4(int N) {
   if (N == 0) return 0;
   return f4(N/2) + f1(N) + f4(N/2);
}
```

_____

```
public static int f5(int N) {
   int x = 0;
   for (int i = N; i > 0; i = i/2)
      x += f1(i);
   return x;
}
```

_____

```
public static int f6(int N) {
   if (N == 0) return 1;
   return f6(N-1) + f6(N-1);
}
```

_____

```
public static int f7(int N) {
   if (N == 1) return 0;
   return 1 + f7(N/2);
}
```

A. $\log N$

B. N

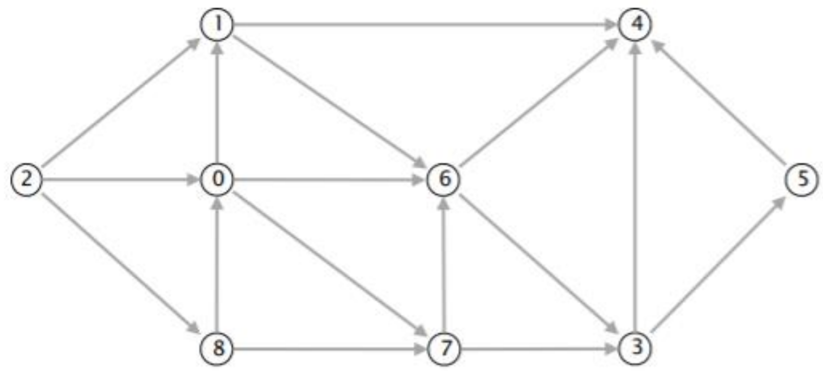
C. $N \log N$

D. N^2

E. 2^N

F. $N!$

11) Consider the following acyclic digraph. Assume the adjacency lists are in sorted order: for example, when iterating through the edges pointing from 0, consider the edge 0 → 1 before 0 → 6 or 0 → 7.



(a) Compute the topological order by running the DFS-based algorithm and listing the vertices in *reverse postorder*.

2

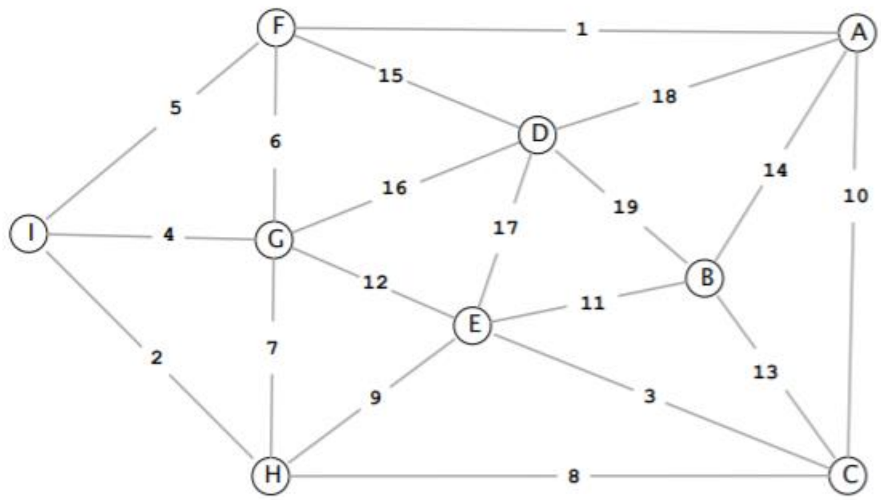
--- --- --- --- --- --- --- --- ---

(b) Run breadth-first search on the digraph, starting from vertex 2. List the vertices in the order in which they are dequeued from the FIFO queue.

2

--- --- --- --- --- --- --- --- ---

12) Consider the following edge-weighted graph with 9 vertices and 19 edges. Note that the edge weights are distinct integers between 1 and 19.



(a) Complete the sequence of edges in the MST in the order that *Kruskal's algorithm* includes them (by specifying their edge weights).

1
---- ---- ---- ---- ---- ---- ---- ----

(b) Complete the sequence of edges in the MST in the order that *Prim's algorithm* includes them (by specifying their edge weights).

1
---- ---- ---- ---- ---- ---- ---- ----

13)

Consider the *first* call to key-indexed counting when running LSD string sort on the input array `a[]` of 20 strings. Recall that key-indexed counting is comprised of four loops. Give the contents of the integer array `count[]` after each of the first three loops (for indices between `'a'` and `'g'`); then, give the contents of the string array (for the indices 0–5 and 18–19) after the fourth loop.

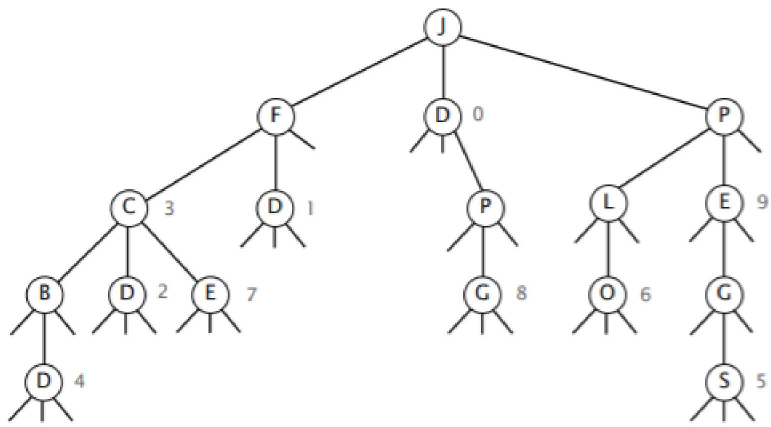
i	a[i]
0	badge
1	freed
2	blurb
3	embed
4	basic
5	field
6	bluff
7	dwarf
8	fudge
9	climb
10	cycle
11	bleed
12	budge
13	crumb
14	cubic
15	cable
16	blend
17	cliff
18	bread
19	cache

c	count[] (*first*)	count[] (*second*)	count[] (*third*)
⋮	⋮	⋮	⋮
'a'			
'b'			
'c'			
'd'			
'e'			
'f'			
'g'			
⋮	⋮	⋮	⋮

i	a[i] (*fourth*)
0	
1	
2	
3	
4	
5	
6	not required
7	not required
8	not required
9	not required
10	not required
11	not required
12	not required
13	not required
14	not required
15	not required
16	not required
17	not required
18	
19	

14)

Consider the following ternary search trie, with string keys and integer values.



Circle which one or more of the following strings are keys in the TST.

`B    BD    C    CD    D    E    FD    JLO    JP    JPEG    JPEGS    JPG    PEG    PEGS`

Solutions:

1) Any unconnected, weighted, undirected graph
2) Graph needs 2 non-self edges, with exactly 2 having the same weight and the third edge having a lower weight
3) Graph needs 3 non-self edges, all with the same weight
4) O(n log n)
5) O(n log n)
6) O(n^2)
7) O(n^2)
8) O(n^2)
9) O(n log n)
10) B D F C B E A
11) (a) 2 8 0 7 1 6 3 5 4 (b) 2 0 1 8 6 7 4 3 5
12) (a) 1 2 3 4 5 8 11 15 (b) 1 5 2 4 8 3 11 15 The starting vertex must be either A or F (but it doesn't matter which).
13)

c	count[]	count[]	count[]
⋮	⋮	⋮	⋮
'a'	0	0	0
'b'	0	0	3
'c'	3	3	5
'd'	2	5	11
'e'	6	11	17
'f'	6	17	20
'g'	3	20	20
⋮	⋮	⋮	⋮

i	a[i]
0	blurb
1	climb
2	crumb
3	basic
4	cubic
5	freed
⋮	⋮
18	dwarf
19	cliff

14) BD C CD E FD JPG PEGS