Algorithmics D-Side!!

Answer guide by Tunan Shi :)

Section A

Question 1

The 32 nodes should be arranged in the shape of a spine, or a single chain hydrocarbon. This yields a maximum distance of 11.

Answer: A

Question 2

The best thing you can do is make the size of both black and white sets of nodes 50 nodes. Then you add a single edge between every pair of nodes in both sets.

Answer: C

Question 3

To determine of two nodes in a directed graph are reachable from one another in the most efficient way, we should use depth-first search. Note: A linear time algorithm for finding all SCCs at the same time does exist using DFS.

Answer: B

Question 4

A DP algorithm never calculates the same state more than once, so if it has an O(n) recurrence, that must mean that O(n * space complexity) = time complexity.

Answer: A

Question 5

Option C is definitely true because if an HP-Hard problem was solvable in polynomial time, then clearly all NP problems are solvable in polynomial time.

Answer: C

Question 6

Option B is definitely true because Hamiltonian Path Problem can be solved in polynomial time, but problems like the Halting Problem (also in NP-Hard) cannot.

Answer: B

Question 7

A few rough drawings can help you come to a conclusion with this question.

Answer: B

Question 8

$1.328 / 0.488 = ka^{12} / ka^{10}$

$2.72 = a^2$

$a = 1.65$

Answer: A

Question 9

Obviously node 600 would be the deepest node in the tree. Hence, you would want to follow it up the tree to see how deep it is.

$600 \rightarrow 9 \rightarrow 3 \rightarrow 1$

Answer: B

Question 10

Node 1 is parent to nodes 2 and 3. Node 2 is parent to nodes 4 to 7. Node 3 is parent to nodes 8 to 15. If you continue this process, you will realise that node 8 has connections to nodes 256 to 511. This makes 256 nodes. Including the parent node 3, that makes 257 nodes in total.

Answer: D

Question 11

$2T(n-2) < T(n) < 2T(n-1)$

$O(1.414^n) < T(n) < O(2^n)$

Hence, T(n) must grow exponentially.

Answer: C

Question 12

Each edge of the graph can connect 2 connected components together, decrementing the total number. Hence, the answer is 1000 – 967 = 33.

Answer: C

Question 13

If you want all of the edges to bunch up, then you obviously want to sacrifice the smallest number of nodes for the cause. The way to do this is to get a complete subgraph 45 nodes (990 edges), and delete some of the 990 edges so that G has exactly 967 edges. This makes 956 connected components.

Answer: B

Question 14

The non-decision problem TSP cannot be verified in polynomial time as of yet.

Answer: C

Question 15

The Church-Turing thesis is a postulate. This means that it has not been proved. Hence, you cannot prove that you can't build something that's stronger than a Turing machine.

Answer: D

Question 16

They all run infinitely. Smh.

Answer: D

Question 17

What this question is really asking is if O(log(A+B)) is the same as O(log(AB)). Many people would have guessed C, but the proof is as follows:

log(AB) = log(A) + log(B) = O(max(log(A), log(B))) = O(log(max(A, B))) = O(log(A+B))

Answer: C

Question 18

The w loop must form the outer loop in order for the Floyd-Warshall to work. The ordering of the other two is irrelevant.

Answer: B

Question 19

Option II is used to fix a genuine bug in the code. Option IV causes the string reversal to intersperse, as follows:

abcde**f** → fb**cd**ea → f**b**dc**e**a → fedcba

Answer: A

Question 20

Minimum bottleneck path can be solved with Prim’s if we start from the higher weight edges first, because Prim’s gradually builds a perimeter of nodes we can reach given our current minimum edge weight.

Answer: A