X

NPTEL

Courses » Design and Analysis of Algorithms

Week 1 Quiz

1) In the code fragment below, `start` and `end` are integer values and `prime(x)` is a function that returns true if `x` is a prime number and false otherwise. *2 points*

```
i := 0; j := 0; k := 0;
for (m := start; m <= end; m := m+1){
  k := k + m;
  if (prime(m)){
    i := i + m;
  }else{
    j := j + m;
  }
}
```

At the end of the loop:

◯ `k < i+j`
◯ `k = i+j`
◯ `k > i+j`
◯ Depends on `start` and `end`

2) How many times is the comparison i `<= n` performed in the following program? *2 points*

```
int i = 10, n = 100;
main(){
  while (i <= n){
    i = i+1;
    n = n-1;
  }
}
```

◯ 45
◯ 46
◯ 47
◯ 90

3) An algorithm has two phases. The first phase, initialization, takes time $O(n^2)$. The second phase, which is the main computation, takes time O(n log n). What is the most accurate description of the complexity of the overall algorithm? *2 points*

◯ O(n log n)
◯ $O(n^2)$

- ◯ $O(n^2 \log n)$
- ◯ $O(n^2 + \log n)$

4) Let $f(n) = 560n^3 + 3n + 107$ and $g(n) = 3n^3 + 5000n^2$. Which of the following is true? *2 points*
- ◯ f(n) is O(g(n)), but g(n) is not O(f(n))
- ◯ g(n) is O(f(n)), but f(n) is not O(g(n))
- ◯ f(n) is not O(g(n)) and g(n) is not O(f(n))
- ◯ f(n) is O(g(n)) and g(n) is O(f(n))

5) if $T(n) = n \sqrt{n}$ then *2 points*
- ◯ T(n) = O(n)
- ◯ T(n) = O(n log n)
- ◯ $T(n) = O(n^2)$
- ◯ None of the above

Week 2 Quiz

6) Assume that a CPU can process 10^8 operations per second. Suppose you have to sort an array with 10^6 elements. Which of the following is true? *2 points*
- ◯ Insertion sort will always take more than 2.5 hours while merge sort will always take less than 1 second.
- ◯ Insertion sort will always take more than 2.5 hours while quicksort will always take less than 1 second
- ◯ Insertion sort could take more than 2.5 hours while merge sort will always take less than 1 second.
- ◯ Insertion sort could take more than 2.5 hours while quicksort will always take less than 1 second.

7) Suppose our aim is to sort an array in descending order. Which of the following statements is true? *2 points*
- ◯ Input in descending order is worst case for both selection sort and insertion sort.
- ◯ Input in ascending order is worst case for both selection sort and insertion sort.
- ◯ Input in descending order is worst case for insertion sort but not for selection sort.
- ◯ Input in ascending order is worst case for selection sort but not for insertion sort.

8) Suppose we want to sort an array in descending order and we implement quicksort so that we always choose the last element in the array as the pivot element. Assume that the input is a permutation of {1, 2, …, n}. Which of the following would be a worst case permutation of this input for this implementation of quicksort? *2 points*
- ◯ {1, 2,…, n} with all even numbers in random order followed by all odd numbers in descending order.
- ◯ {1, 2, . . . , n} in descending order.
- ◯ {1,2,...,n} in some random order.
- ◯ {1, 2, . . . , n} with all odd numbers in ascending order followed by all even numbers in random order.

9) Which of the following statements is *not* true? *2 points*
- ◯ For every fixed strategy to choose a pivot for quicksort, we can construct a worst case input that requires time $O(n^2)$.
- ◯ If we randomly choose a pivot element each time, quicksort will always terminate in time O(n log n).
- ◯ If we could find the median in time O(n), quicksort would have worst case complexity O(n log n).
- ◯ Quicksort and merge sort are both examples of divide and conquer algorithms.

10) We have a list of points [(7,1),(3,5),(6,1),(6,5),(0,2),(9,0)]. We sort these in ascending order by the second coordinate. Which of the folllowing corresponds to a stable sort of this input? *2 points*

- ◯ [(9,0),(7,1),(6,1),(0,2),(3,5),(6,5)]
- ◯ [(0,2),(3,5),(6,1),(6,5),(7,1),(9,0)]
- ◯ [(9,0),(6,1),(7,1),(0,2),(3,5),(6,5)]
- ◯ [(9,0),(7,1),(6,1),(0,2),(6,5),(3,5)]

Week 3 Quiz

11) An undirected graph G has 100 nodes and there is a path from every vertex to every other vertex in G. Suppose v_1 and v_2 are two vertices in G. Then, what can we say about the shortest path from v_1 to v_2? *2 points*

- ◯ The path has at most 2 edges.
- ◯ The path has at most 50 edges.
- ◯ The path has at most 99 edges.
- ◯ None of the above.

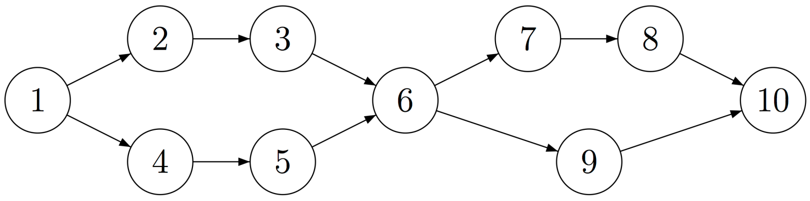
12) An undirected graph G has 300 edges. The degree of a vertex v, deg(v), is the number of edges connected to v. Suppose $V = \{v_1, v_2, ..., v_{30}\}$. What is the value of $\deg(v_1) + \deg(v_2) + \cdots + \deg(v_{30})$? *2 points*

- ◯ 300
- ◯ 600
- ◯ 900
- ◯ None of these

13) Which of the following is not true of depth-first search (DFS) starting at a vertex v? *2 points*

- ◯ DFS identifies all vertices reachable from v.
- ◯ Using an adjacency list instead of an adjacency matrix can improve the worst case complexity.
- ◯ DFS will identify the shortest paths from v in any graph without edge weights.
- ◯ DFS numbering can be used to identify cycles in the graph.

14) How many topological orderings does the following directed acyclic graph have? *2 points*



- ◯ 6
- ◯ 12
- ◯ 18
- ◯ 24

15) Anjali has enrolled for a part-time Masters programme where she has to complete 8 courses, numbered 1, 2, . . . , 8. Each course takes a full semester to complete. She can take as many or as few courses as she wants in each semester. *2 points*
Some courses are prerequisites for other courses. If course A is a prerequisite for course B, she can take course B the semester after she finishes course A, or any time after that, but not before.
Given the following information about prerequisites, compute the minimum number of semesters she needs to complete these courses

- Prerequisites for course 1: course 2,4,5,7
- Prerequisites for course 2: course 3
- Prerequisites for course 3: course 5,6

- Prerequisites for course 4: course 8
- Prerequisites for course 5: course 8
- Prerequisites for course 6: course 7,8

○ 3
○ 4
○ 5
○ 6

Week 4 Quiz

16) Consider the following fragment of code for a graph algorithm on an undirected graph. *2 points*

```
for each vertex i in V
  mark i as visited
  for each edge (j,i) pointing into i
     update weight(j,i) to weight(j,i) + k
```

Which of the following is the most accurate description of the complexity of this fragment. (Recall that n is the number of vertices, m is the number of edges.)

○ O(n)
○ O(nm)
○ O(n+m)
○ O(m)

17) Consider the following strategy to solve the single source shortest path problem with edge weights from source s. *2 points*

1. Replace each edge with weight w by w edges of weight 1 connected by new intermediate nodes
2. Run BFS(s) on the modified graph to find the shortest path to each of the original vertices in the graph.

Which of the following statements is correct?

○ This strategy will solve the problem correctly but is not as efficient as Dijkstra's algorithm.
○ This strategy will solve the problem correctly and is as efficient as Dijkstra's algorithm.s st
○ This strategy will not solve the problem correctly.
○ This strategy will only work if the graph is acyclic.

18) Consider the following strategy to convert a graph with negative edge weights to one that does not have negative edge weights. Let the maximum magnitude negative edge weight in the graph be -k. Then, for each edge in the graph with weight w, update the weight to w+k+1. Consider the following claim: *2 points*

- To solve the shortest path problem in the original graph, we can run Dijkstra's algorithm on the modified graph and subtract the added weights to get the original distances.

Which of the following is *not* correct.

○ The claim is not true in general.
○ The claim is true for all graphs.
○ The claim is true for connected acyclic graphs.
○ The claim is not true in general for connected graphs with cycles.

19) Consider the following algorithm on a graph with edge weights. *2*

points

- Sort the edges as [e_1,e_2,...,e_m] in decreasing order of cost.
- Start with the original graph. Consider each edge e_j. If this edge is part of a cycle delete it.

Which of the following is *not* true.
◯ After processing all m edges, the resulting graph is connected.
◯ What remains at the end is a minimum cost spanning tree.
◯ Exactly m-n+1 edges will be deleted.
◯ At most n-1 edges will be deleted.

20) Suppose we have a graph with negative edge weights. We take the largest magnitude negative edge weight -k and reset each edge weight w to w+k+1. Which of the following is true? *2 points*
◯ Kruskal's algorithm will identify the same spanning tree on the new graph as on the old graph.
◯ Minimum cost spanning trees in the original graph will not correspond to minimum cost spanning trees in the new graph.
◯ Prim's algorithm will not work on the original graph but will work on the modified graph.
◯ There are more minimum cost spanning trees in the modified graph than in the original graph.

Week 5 Quiz

21) Suppose we want to extend the union-find data structure to support the operation Reset(c), which takes as input the name of a component c and then breaks up c into singleton components, like MakeUnionFind(). For instance if c = 3 and c currently consists of {1,3,7}, then Reset(c) will produce three components called 1, 3 and 7 consisting of {1}, {3} and {7}, respectively.
Which of the following is correct about the cost of adding Reset(c) to the array and pointer implementations of union-find discussed in the lecture? *2 points*
◯ Array representation: O(n), Pointer representation: O(n)
◯ Array representation: O(size(c)), Pointer representation: O(n)
◯ Array representation: O(n), Pointer representation: O(size(c))
◯ Array representation: O(size(c)), Pointer representation: O(size(c))

22) Suppose we want to delete an arbitrary element from a max heap. (Assume we have auxiliary arrays NodeToHeap[] and HeapToNode[] required for the update() operation.)
Consider the following strategies. *2 points*

- Strategy 1: Remove the element from the array, compress the array and reheapify.
- Strategy 2: Update the value of this node to the current maximum value in the heap + 1, then delete_max.

◯ Strategy 1 takes time O(n log n) and Strategy 2 takes time O(n)
◯ Strategy 1 takes time O(n) and Strategy 2 takes time O(n log n)
◯ Strategy 1 takes time O(log n) and Strategy 2 takes time O(n)
◯ Strategy 1 takes time O(n) and Strategy 2 takes time O(log n)

23) Suppose we want to support the operations predecessor and successor in a heap. Given a value v in the heap, pred(v) tells us the next smaller value currently in the heap and succ(v) tells us the next larger value currently in the heap. *2 points*
◯ In both min heaps and max heaps, both operations take time O(n).
◯ In both min heaps and max heaps, both operations take time O(log n).
◯ In a min heap, pred(v) takes time O(log n) and succ(v) takes O(n) whereas in a max heap pred(v) takes time O(n) and succ(v) takes O(log n).
◯ In a min heap, pred(v) takes time O(n) and succ(v) takes O(log n) whereas in a max heap pred(v)

takes time O(log n) and succ(v) takes O(n).

24) Suppose we do merge sort with a three-way split: divide the array into 3 equal parts, sort each part and do a 3 way merge. What would the worst-case complexity of this version be? *2 points*

◯ $O(n^2)$
◯ $O(n^2 \log_3 n)$
◯ $O(n \log_2 n)$
◯ $O(n (\log_2 n)^2)$

25) In the closest pair of points problem, we have assumed that no two points have the same x or y coordinate. Which of the following steps would become more complicated to justify without this assumption. *2 points*

◯ Constructing Q_Y and R_Y from P_Y in time O(n) in the divide step.
◯ Constructing Q_X and R_X from P_X in time O(n) in the divide step.
◯ Constructing S_Y from Q_Y and R_Y in time O(n) the combine step.
◯ Arguing that every d/2 side square in the d-band around the separator can have at most one point.

Week 6 Quiz

26) What is the complexity of inorder traversal for a binary search tree with n nodes? *2 points*

◯ O(log n) whether the tree is balanced or unbalanced.
◯ O(log n) if the tree is balanced, O(n) otherwise.
◯ O(n) whether the tree is balanced or unbalanced.
◯ O(n) if the tree is balanced, O(n log n) otherwise.

27) Suppose we do not have a parent pointer in the nodes of a search tree, only left-child and right-child. Which of the following operations can be computed in time O(log n) for a balanced search tree? *2 points*

◯ find, insert, delete, but not min, max, pred, succ
◯ find, insert, delete, min, max, but not pred, succ
◯ find, insert, delete, pred, succ but not min, max
◯ All of find, insert, delete, min, max, pred, succ

28) Consider the following algorithm to build a balanced search tree from a sorted sequence. *2 points*

- Make the mid-point of the sequence the root of the tree
- Recursively construct balanced search trees from elements to the left and right of the midpoint and make these the left and right subtrees of the root.

What is the complexity of this procedure where the sorted sequence is a sorted *array*?

◯ O(n)
◯ O(n log n)
◯ $O(n^2)$
◯ Depends on the contents of the original sequence

29) Consider the following algorithm to build a balanced search tree from a sorted sequence. *2 points*

- Make the mid-point of the sequence the root of the tree
- Recursively construct balanced search trees from elements to the left and right of the midpoint and make these the left and right subtrees of the root.

What is the complexity of this procedure where the sorted sequence is a sorted *list*?
- ◯ O(n)
- ◯ O(n log n)
- ◯ O(n^2)
- ◯ Depends on the contents of the original sequence.

30) Which of the following is not a greedy algorithm? *2 points*
- ◯ Dijkstra's algorithm for single source shortest paths
- ◯ Bellman-Ford algorithm for single source shortest paths
- ◯ Prim's algorithm for minimum cost spanning tree
- ◯ Kruskal's algorithm for minimum cost spanning tree

Week 7 Quiz

31) We want to write down a recursive expression for Profit[i]. *2 points*

- Profit[N] = 0
- For 1 ≤ i < N, Profit[i] = ??

◯ max(Profit[i+1], Profit[i+2] + Price[i+1] - Price[i] - 2)
◯ max(Profit[i+2] + Price[i+1] - Price[i] - 2, Profit[i+3] + Price[i+2] - Price[i] - 2, ... Profit[N] + Price[n-1] - Price[i] - 2, Price[n] - Price[i] - 2)
◯ max(Profit[i+1], Profit[i+2] + Price[i+1] - Price[i] - 2, Profit[i+3] + Price[i+2] - Price[i] - 2, ... Profit[N] + Price[n-1] - Price[i] - 2)
◯ max(Profit[i+1], Profit[i+2] + Price[i+1] - Price[i] - 2, Profit[i+3] + Price[i+2] - Price[i] - 2, ... Profit[N] + Price[n-1] - Price[i] - 2, Price[n] - Price[i] - 2)

32) What is the size of the memo table for this problem? *2 points*
- ◯ N
- ◯ N-1
- ◯ N+1
- ◯ N^2

33) What is a good order to compute the values of Profit[i] using dynamic programming? *2 points*
- ◯ From Profit[1] to Profit[N]
- ◯ From Profit[N] to Profit[1]
- ◯ Either from Profit[1] to Profit[N] or from Profit[N] to Profit[1]
- ◯ None of these

34) How much time will dynamic programming take to compute Profit[1]? *2 points*
- ◯ O(N)
- ◯ O(N log N)
- ◯ O(N^2)
- ◯ O(N^2 log N)

35) What is the value of Profit[1] in the example given in the problem statement? *2 points*
- ◯ 6
- ◯ 7
- ◯ 8

- 9

Week 8 Quiz

36) Which of the following is not a linear constraint? *2 points*
- 8x ≤ 3y + 25
- 5x + 6xy ≤ 5
- 9x ≥ 7y + 8z
- 5y + 3x ≥ 33

37) When we model a graph problem using LP, mapping each path to a variable is not a good strategy because: *2 points*
- Paths can be hard to compute.
- We have to be careful to avoid cycles.
- A graph has exponentially many paths.
- Edges may be directed.

38) Suppose we compute the maximum s-t flow F in a network. Then, which of the following is true of s-t cuts? *2 points*
- From F, we can identify a minimum s-t cut in polynomial time.
- From F, we can identify all minimum s-t cuts in polynomial time.
- From F, we know the capacity of the minimum s-t cut, but identifying such a cut can take exponential time.
- F gives a lower bound for the capacity of the minimum s-t cut but not the exact capacity.

39) We have an exponential time algorithm for problem A, and problem A reduces in polynomial time to problem B. From this we can conclude that: *2 points*
- B has an exponential time algorithm.
- B cannot have a polynomial time algorithm.
- A cannot have a polynomial time algorithm.
- None of the above.

40) Suppose SAT reduces to a problem C. To claim that C is NP-complete, we additionally need to show that: *2 points*
- There is a checking algorithm for C.
- Every instance of C maps to an instance of SAT.
- Every instance of SAT maps to an instance of C.
- C does not have an efficient algorithm.