CSE 101 - Winter 2015
Quiz 2 Solutions

January 27, 2015

1. True or False: For any DAG \( G = (V, E) \) with at least one vertex \( v \in V \), there must exist at least one topological ordering. (Answer: True)

Solution

Fact (from class) If \( G \) is a DAG, then \( G \) has at least one source (a vertex of indegree 0) and at least one sink (a vertex of outdegree 0).

Base Case: When \( G \) has one vertex, the only topological ordering is this single vertex, so the Claim is true for \( n = 1 \).

Induction Hypothesis: The Claim is true for all DAGs with \( n \leq k \) vertices.

Induction Step: Let \( G \) be any DAG with \( k + 1 \) vertices. Since \( G \) is a DAG, by the Fact, there is at least one sink vertex \( v \) in \( G \). Remove \( v \) (and any edges incident to \( v \)) from \( G \) to obtain graph \( G' \) with \( k \) vertices. By the Induction Hypothesis, \( G' \) has a topological ordering. We can obtain a topological ordering for \( G \) by placing vertex \( v \) at the end of \( G' \)'s topological ordering. Thus the Claim is true for \( n = k + 1 \).

2. Which of the following are possible topological orderings of the vertices of this graph? Mark ALL answers that give valid topological orderings. You will get credit ONLY for marking all valid topological orderings (and no other orderings). (Answer: ADBFEC, ADFBEC, AFDBEC)
Solution
By inspection: A must be first and C must be last in any topological ordering. Either D or F can follow A in a topological ordering; both of these must be ordered before E. If the ordering starts with AD, then either B or F can be next. There are three possible topological orderings: ADBFEC, ADFBEC and AFDBEC.

3. How many distinct topological orderings does the following directed graph have? (Answer: 12)

![Directed Graph]

a) 0  
b) 4  
c) 24  
d) 12  
e) undefined

Solution
In any topological ordering, C must precede D (because there is a directed edge from C to D in the graph). The isolated vertices A and B can occur in any positions in a topological ordering. Notice that of the 4! = 24 permutations of A, B, C, D, half have C before D, and half have D before C. So, there are 12 topological orderings of the given graph.

4. Execute DFS on the following directed graph. How many tree, forward, back and cross edges will there be? Remember to break ties correctly! (Answer: 1 tree, 0 forward, 0 back, 4 cross)

![Directed Graph]
a) 1 tree, 1 forward, 1 back, 2 cross

b) 2 tree, 1 forward, 1 back, 1 cross

c) 1 tree, 0 forward, 1 back, 2 cross

d) 1 tree, 0 forward, 0 back, 4 cross

Solution
Ties are broken by lexicographic order. Thus, \((\text{pre}, \text{post})\) of \(A = (1,4)\), of \(B = (2,3)\), of \(C = (5,6)\), of \(D = (7,8)\). The directed edge \((A,B)\) is a tree edge. Edges \((C,A)\), \((C,B)\), \((D,B)\) and \((D,C)\) are all cross edges.

5. Consider the following pseudo-code:

```python
function func(n):
    if (n > 1):
        return func(n/2) + func(n/2)
    else:
        return 1
endif
```

Which of the following is the correct recurrence relation for the running time \(T(n)\) of the function \(func(n)\)? (Answer: \(T(n) = 2T(n/2) + 1\))

a) \(T(n) = 2T(n/2) + n\)  

b) \(T(n) = 3T(n/2) + 1\)  

c) \(T(n) = 3T(n/2) + n\)  

d) \(T(n) = 2T(n/2) + 1\)

Solution
It can be observed that in each recursive step, two subproblems are generated, each half the size of the original problem. The single addition to compose the subsolutions takes unit time.
6. The runtime of a divide-and-conquer algorithm is described by the following recurrence: \( T(n) = 3T(n/2) + O(1) \). Assume that \( n \) is a sufficiently large integer >> (i.e., much greater than) 3^5. How many subproblems will we have at the 5th level of recursion if the top level is considered to be the 0th level? (Answer: 243)

a) 15  
b) 27  
c) 81  
d) 243

**Solution**

From the recurrence relation, we note that the branching factor \( a = 3 \). At level 0, there is one problem. At each subsequent level, the number of subproblems is multiplied by \( a \). Therefore, at the 5th level of recursion, the total number of subproblems is given by \( a^5 = 3^5 = 243 \).

7. The runtime of a divide-and-conquer algorithm is described by the following recurrence: \( T(n) = 4T(n/2) + n \). What is the total contribution to the runtime of the “\(+ n\)” term over all subproblems at the third level of the recursion, if the top level is considered to be the 0th level? Assume that \( n \) is a sufficiently large integer such that more than 4 levels of recursion will occur, and hence non-zero work will be done at the 3rd level. (Answer: 8n)

a) \( n \)  
b) \( 2n \)  
c) \( 8n \)  
d) \( 16n \)  
e) \( 32n \)

**Solution**

The “\(+ n\)” term indicates that the amount of work charged to a given subproblem is \( 1 \) times the subproblem’s size. The total amount of work done at a particular level is given by the number of subproblems at the level, times the problem size of each subproblem at the level.
From the recurrence, at the third level of recursion there will be \(4^3 = 64\) subproblems of size \(n/2^3 = n/8\) each. Hence, the total amount of work done at the third level of recursion is \(64 \times n/8 = 8n\).

8. True or False: You are given an array \(A\) of size \(n\), whose elements are in sorted order (ascending or descending), and a number \(k\). Every algorithm that takes \(A\) and \(k\) as inputs and returns true if \(k\) is an element in \(A\), and false otherwise, requires \(\Omega(n \log n)\) time in the worst case.
(Answer: False)

Solution
Binary search allows us to search in \(O(\log n)\) time in the worst case.

9. Consider an algorithm whose running time \(T(n)\) is given by the following recurrence: \(T(n) = 27T(n/3) + O(n^d)\). What is the largest possible positive integer value of \(d\) such that the time complexity of the algorithm is \(O(n^3)\)? (Answer: 2)

   a) 1
   
   b) 2
   
   c) 3
   
   d) 4

Solution
In the given recurrence, \(a = 27\) and \(b = 3\). By the Master Theorem, we will have a running time of \(O(n^3)\) until \(a = b^d\). This happens when \(d = 3\), and the running time with \(d = 3\) is \(O(n^3 \log n)\). So the largest integer value of \(d\) that we seek is 2.

10. A divide-and-conquer algorithm for finding the largest and smallest elements in a given array works by dividing the array into two halves, recursively finding the min and max element in each of the two smaller arrays, and then comparing the two max elements and the two min elements to obtain the overall max and min elements. What is the correct recurrence relation for the running time \(T(n)\) of this algorithm?
(Answer: \(T(n) = 2T(n/2) + 2\))

   a) \(T(n) = 2T(n/2) + 2\)
   
   b) \(T(n) = T(3n/4) + T(n/4) + 2\)
   
   c) \(T(n) = T(2n/3) + T(n/3) + 2\)
d) None of the above

Solution
At the recursive step, two subproblems are generated, each with half
the problem size. We use exactly two comparisons to compose the
subsolutions – i.e., one comparison of the two max elements and one
comparison of the two min elements.