Unit 5 - Week 1 Quiz

Week 1 Quiz

The due date for submitting this assignment has passed. **Due on 2019-02-06, 23:59 IST.**

Assignment submitted on 2019-01-31, 11:37 IST

All questions carry equal weightage. You may submit as many times as you like within the deadline. Your final submission will be graded.

1) An image processing application involves multiplying two n×n matrices A and B to yield a new matrix C in time $O(n^3)$ followed by an edge detection phase that takes times $O(n^2)$ for each of the matrices A, B, and C. What is the most accurate and concise description of the complexity of the overall algorithm? ***2 points***

- $O(n^2)$
- $O(n^3)$
- $O(n^2+n^3)$
- $O(n^5)$

Yes, the answer is correct.
Score: 2

Feedback:
When there are multiple phases in sequence, the largest of the phases dominates the overall complexity. Here the first phase is $O(n^3)$ and the second phase is $O(n^2)$.

Accepted Answers:
$O(n^3)$

2) We are trying to determine the worst case time complexity of a library function that is provided to us, whose code we cannot read. We test the function by feeding large numbers of random inputs of different sizes. We find that for inputs of size 40 and 400, the function always returns within one second, but for inputs of size 4,000 it sometimes takes about a minute and for inputs of size 40,000 it sometimes takes more than 10 minutes. What is a reasonable conclusion we can draw about the worst case time complexity of the library function? (You can assume, as usual, that a typical desktop PC performs 10^9 basic operations per second.) ***2 points***

- $O(n^2)$
- $O(n^2 \log n)$
- $O(n^3)$
- $O(n^4)$

Yes, the answer is correct.
Score: 2

Feedback:
Assuming a CPU with 10^9 operations per second, 400^3 would be well under 10^9 whereas 4000^3 is 64×10^9 (about 1 minute) and 40000^3 is 64000×10^9 (about 15 minutes).

Accepted Answers:

$O(n^3)$

3) Suppose g(n) is $3n^4+2n+5$ and h(n) is $4n^3 + 5n^2 + 12$. Let f(n) be a third, unknown function. Which of the following is not possible. **2 points**

- f(n) is O(g(n)) and f(n) is also O(h(n))
- f(n) is not O(g(n)) and f(n) is also not O(h(n))
- f(n) is O(g(n)) but f(n) is not O(h(n))
- f(n) is O(h(n)) but f(n) is not O(g(n))

Yes, the answer is correct.
Score: 2

Feedback:
Since h(n) is O(g(n)), if f(n) is O(h(n)) it must also be O(g(n)). All other combinations are possible.

Accepted Answers:
f(n) is O(h(n)) but f(n) is not O(g(n))

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

```
int i = 100, n = 40;
main(){
  while (i >= n){
    i = i-1;
    n = n+1;
  }
}
```

- 30
- 31
- 32
- 33

Yes, the answer is correct.
Score: 2

Feedback:
After 30 iterations, `i` is 70 and `n` is 70. At this point, `i==n` so the 31st iteration succeeds. The next test of the `while` condition fails and exits the loop. Hence, overall the while condition is checked 32 times.

Accepted Answers:
32

5) If T(n) is O(n $\sqrt{}$ n) which of the following is false? **2 points**

- T(n) is O(n log n)
- T(n) is $O(n^2)$
- T(n) is $O(n^2 \log n)$
- T(n) is $O(n^3)$

Yes, the answer is correct.
Score: 2

Feedback:
$\sqrt{}$ *n is not O(log n)*

Accepted Answers:
T(n) is O(n log n)

End

X

Unit 7 - Week 2 Quiz

Register for Certification exam

Course outline

How to access the portal

Week 1: Introduction

Week 1: Analysis of algorithms

Week 1 Quiz

Week 2: Searching and sorting

Week 2 Quiz

Quiz : Week 2 Quiz

Week 2 Programming Assignment

Week 3: Graphs

Week 3 Quiz

Week 3 Programming Assignment

Week 4: Weighted graphs

Week 4 Quiz

Week 4 Programming Assignment

Week 5: Data Structures: Union-Find and Heaps

Week 2 Quiz

The due date for submitting this assignment has passed. **Due on 2019-02-13, 23:59 IST.**

Assignment submitted on 2019-02-06, 08:44 IST

All questions carry equal weightage. You may submit as many times as you like within the deadline. Your final submission will be graded.

1) An array A contains N integers arranged in a random sequence. We want to check if all entries in A are distinct. Which of the following would be the most efficient algorithm, asymptotically. ***2 points***

- For each pair of positions i and j, check if A[i] is equal to A[j].
- Sort the array using insertion sort. When inserting each element into the sorted prefix, check if there is already a value equal to it in the prefix.
- Sort the array using quick sort. Use binary search on the sorted array to find duplicated values.
- Sort the array using merge sort. Scan the sorted array from beginning to end looking for adjacent equal elements.

Yes, the answer is correct.
Score: 2

Feedback:
Merge sort is O(n log n) and doing a linear scan is O(n), so asymptotically this can be done in O(n log n). Comparing all pairs of indices is $O(n^2)$, as is using insertion sort to do the identification or using quicksort. Also, for the third option it is not clear why binary search is useful.

Accepted Answers:
Sort the array using merge sort. Scan the sorted array from beginning to end looking for adjacent equal elements.

2) Suppose our aim is to sort an array in ascending order. Which of the following statements is true? ***2 points***

- Input in descending order is worst case for selection sort but not for insertion sort.
- Input in ascending order is worst case for insertion sort but not for selection sort.
- 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.

Yes, the answer is correct.
Score: 2

Feedback:
Input in descending order and ascending order are both worst case for selection sort. Input in descending order is worst case for insertion sort.

Accepted Answers:
Input in descending order is worst case for both selection sort and insertion sort.

3) Suppose we want to sort an array in ascending 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 ***2 points***

permutation of {1, 2, …, n}. Which of the following would ***definitely*** be a worst case permutation of this input for this implementation of quicksort?

- ○ {1, 2, …, n} with all even numbers in random order followed by all odd numbers in random order.
- ◉ {1, 2, …, n} in ascending 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

Yes, the answer is correct.
Score: 2

Feedback:
Choosing the last element as pivot for an already sorted array will split the array into size (n-1) and 0 at every time.

Accepted Answers:
{1, 2, …, n} in ascending order.

4) Which of the following statements is ***not*** true? ***2 poi***

- ○ Quicksort and merge sort are both examples of divide and conquer algorithms.
- ○ If we could find the median in time O(n), quicksort would have worst case complexity O(n log n).
- ◉ If we randomly choose a pivot element each time, quicksort will always terminate in time O(n log n).
- ○ For every fixed strategy to choose a pivot for quicksort, we can construct a worst case input that requires time $O(n^2)$.

Yes, the answer is correct.
Score: 2

Feedback:
Even with a random choice of pivot in each iteration, we can only guarantee that the expected time is O(n log n), not the worst-case.

Accepted Answers:
If we randomly choose a pivot element each time, quicksort will always terminate in time O(n log n).

5) We have a list of pairs [("Tariq",71),("Brinda",85),("Shweta",71),("Sunita",85), ("Salma",72), ("Uday",60)], where each pair consists of a student's name and his/her marks in a course. We sort these pairs in ascending order of marks. Which of the folllowing corresponds to a stable sort of this input? ***2 points***

- ○ [("Uday",60),("Tariq",71),("Shweta",71),("Salma",72).("Sunita",85),("Brinda",85)]
- ○ [("Uday",60),("Shweta",71),("Tariq",71),("Salma",72),("Sunita",85),("Brinda",85)]
- ○ [("Uday",60),("Shweta",71),("Tariq",71),("Salma",72),("Brinda",85),("Sunita",85)]
- ◉ [("Uday",60),("Tariq",71),("Shweta",71),("Salma",72),("Brinda",85),("Sunita",85)]

Yes, the answer is correct.
Score: 2

Feedback:
The repeated marks values are 71 and 85. The students with these marks should appear in the same ord in the output as in the input.

Accepted Answers:
[("Uday",60),("Tariq",71),("Shweta",71),("Salma",72),("Brinda",85),("Sunita",85)]

End

X

Unit 10 - Week 3 Quiz

Register for Certification exam

Course outline

How to access the portal

Week 1: Introduction

Week 1: Analysis of algorithms

Week 1 Quiz

Week 2: Searching and sorting

Week 2 Quiz

Week 2 Programming Assignment

Week 3: Graphs

Week 3 Quiz

Quiz : Week 3 Quiz

Week 3 Programming Assignment

Week 4: Weighted graphs

Week 4 Quiz

Week 4 Programming Assignment

Week 5: Data Structures:

Week 3 Quiz

The due date for submitting this assignment has passed. Due on 2019-02-20, 23:59 IST.

Assignment submitted on 2019-02-20, 20:26 IST

All questions carry equal weightage. You may submit as many times as you like within the deadline. Your final submission will be graded.

1) An undirected graph G on 25 vertices has 5 connected components. What is the minimum number of edges in G? ***2 points***

- 24
- 20
- 19
- Depends on the sizes of the five connected components.

Yes, the answer is correct.
Score: 2

Feedback:
A minimal connected graph on n vertices is a tree, with n-1 edges. If each of the five components is a tree and we join them with edges to form a single component, we would have to add 4 edges. The resulting graph would be a tree with 24 edges on 25 vertices. Hence, the five components originally had 20 edges. This is the same, regardless of how the graph is decomposed

Accepted Answers:
20

2) Suppose we have a directed graph G = (V,E) with V = {1,2,…,n} and E is presented as an adjacency list. For each vertex u in V, out(u) is a list $[v_1,v_2,\ldots,v_k]$ such that (u,v_i) in E for each i in {1,2,…,k. ***2 points***

For each u in V, we wish to compute a corresponding list in(u) = $[v_1,v_2,\ldots,v_{k'}]$ such that (v_i,u) in E for each i in {1,2,…,k'.

Let n be the number of vertices in V and m be the number of edges in E. How long would it take to construct the lists in(u), u in V, from the lists out(u), u in V?

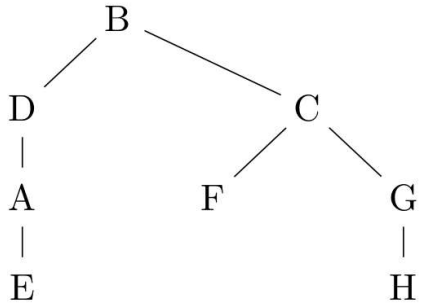
- $O(n^2)$
- $O(n^2 + m)$
- $O(n + m)$
- $O(m)$

No, the answer is incorrect.
Score: 0

Feedback:
We can do it in O(n+m) as follows. Initialize in(u) to empty for each u in V (time O(n)). For each v in V, sca out(v) and for each j in out(v), add v to in(j) (time O(m) across all V).

Accepted Answers:
O(n + m)

3) Suppose we obtain the following DFS tree rooted at node B for an undirected graph with vertices {A,B,C,D,E,F,G,H}. ***2 points***



Which of the following ***cannot*** be an edge in G?

- (B,E)
- (C,H)
- (D,E)
- (F,H)

Yes, the answer is correct.
Score: 2

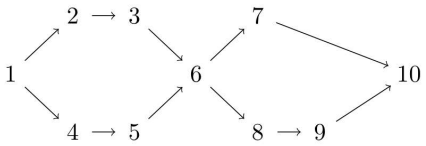
Feedback:
In an undirected graph, all non-tree edges must be along a path from an ancestor to a descendant. (F,H) i an edge across different paths and hence not possible.

Accepted Answers:
(F,H)

4) We are interested in topological orderings of the following DAG that satisfy one or both of the following constraints: ***2 points***

- 3 appears before 5
- 7 appears before 9

How many such orderings are there?



- 21
- 18
- 15
- 3

Yes, the answer is correct.
Score: 2

Feedback:
Any topological sort is of the form 1, followed by a topological ordering of {2,3,4,5}, followed by 6, followed by a topological ordering {7,8,9}.

Nodes {2,3,4,5} have (4 choose 2) = 6 topological orderings of which 3 have 3 before 5 and 3 have 5 before 3.

Nodes {7,8.9} have 3 topological orderings, of which 2 have 7 before 9 and 1 has 7 after 9.

There are 6 × 3 = 18 total topological orderings of which 3 × 1 = 3 violate both conditions, so 15 satisfy on condition or both.

Accepted Answers:
15

5) Assembling a modern car consists of many steps, such as fitting the engine block, connecting the brakes, attaching the doors, etc. Suppose there are 10 steps, labelled A, B, C, D, E, F, G, ***2 points***

H, I, J. Each step takes a day to complete and we have the following dependencies between steps.

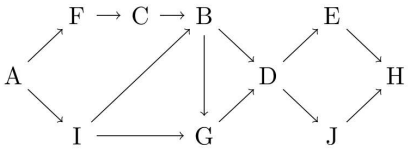
- A must happen before F
- A must happen before I
- B must happen before G
- B must happen before D
- C must happen before B
- D must happen before E
- D must happen before J
- E must happen before H
- F must happen before C
- G must happen before D
- I must happen before B
- I must happen before G
- J must happen before H

What is the minimun number of days required to assemble a car?

- 7
- 8
- 9
- 10

Yes, the answer is correct.
Score: 2

Feedback:
Here is the corresponding dag, whose longest path, for instance A-F-C-B-G-D-E-H, has length 8.



Accepted Answers:
8

End

X

Unit 13 - Week 4 Quiz

Week 4 Quiz

The due date for submitting this assignment has passed. **Due on 2019-02-27, 23:59 IST.**
Score: 6/10=60%

Assignment submitted on 2019-02-27, 18:20 IST

All questions carry equal weightage. You may submit as many times as you like within the deadline. Your final submission will be graded.

1) A new regional airline has adopted a hub-and-spoke model for its operations. Its main hub is at Nagpur. Fron Nagpur, it operates hopping flights to its other destinations. These may visit multiple cities, but the return flights follows the reverse sequence of the outward flights. For instance, a hopping flight from Nagpur to Nasik to Solapur would return back from Solapur to Nasik to Nagpur. ***2 points***

Each direct connection has a fixed ticket cost, that is the same in both directions. The total cost of a hopping flight is the sum of the costs of the individual segments.

We want to compute the cheapest route from a given city to every other city in the airline's network. Which of the following is true for this specific situation?

- We can use BFS to compute the cheapest routes and the answers will be the same as the ones obtained by Dijkstra's algorithm.
- BFS will compute the same cheapest routes as Dijkstra's algorithm if the source node is the hub, Nagpur, but may not give the same answer otherwise.
- BFS will compute the same cheapest routes as Dijkstra's algorithm if the source node is ***not*** the hub, Nagpur, but may not give the same answer otherwise.
- BFS will compute the same cheapest routes as Dijkstra's algorithm for routes that do not pass through the hub, Nagpur, but may not give the same answer otherwise.

No, the answer is incorrect.
Score: 0

Feedback:
The hub-and-spoke network is a weighted tree, so there is only one path between any pair of nodes. Hen BFS will work as well as Dijkstra's algorithm from any starting point.

Accepted Answers:
We can use BFS to compute the cheapest routes and the answers will be the same as the ones obtained by Dijkstra's algorithm.

2) An airline charges a fixed price for each direct flight. For each sequence of hopping flights, the ticket price is the sum of the fares of the individual sectors. TripGuru has precalculated the cheapest routes between all pairs of cities so that it can offer an optimum choice instantly to customers visiting its website. Overnight, the government has added a 13% luxury service surcharge to the cost of each individual flight. Which of the following describes the impact of this surcharge on TripGuru's computation? ***2 points***

- The surcharge favours hopping flights with fewer sectors. TripGuru should recompute any cheapest route where there is a shorter route in terms of number of flights.

- The surcharge favours hopping flights with more sectors. TripGuru should recompute any cheapest route where there is a longer route in terms of number of flights.
- (•) There is no impact. Cheapest routes between all pairs of cities remains unchanged.
- The impact is unpredictable. TripGuru should recompute all cheapest routes.

Yes, the answer is correct.
Score: 2

Feedback:
Each edge weight w goes to 1.13w. Across a path, the path weight p becomes 1.13p. This happens to all paths, so the old shortest paths remain the new shortest paths.

Accepted Answers:
There is no impact. Cheapest routes between all pairs of cities remains unchanged.

3) In the pseudocode below, `W` is the adjacency matrix of a weighted undirected graph over n nodes {1,2,…,n}. More precisely, `W[u,v]` is the weight of the edge `(u,v)` if such an edge exists, and is infinity if there is no edge `(u,v)`. ***2 points***

```
for s,t in 1,2,…,n {
  SP[s,t,0] = 0 if s = t, infinity otherwise
}

for (i = 1; i < n; i++){
  for (t = 1; t <= n; t++){
    SP[s,t,i] = SP[s,t,i-1];
    for (u = 1; u <= n; u++){
      SP[s,t,i] = min(SP[s,t,i],SP[s,u,i-i]+W[u,t]);
    }
  }
}
```

Which of the following is ***not*** a valid statement about this algorithm.

- This pseudocode implements the Floyd-Warshall algorithm.
- (•) At the end of the loop, the values `SP[s,t,n]` represent the shortest paths from `s` to `t`.
- If we extend the loop to more than n iterations, we can detect negative cycles.
- The worst case complexity of this algorithm is $O(n^3)$.

No, the answer is incorrect.
Score: 0

Feedback:
This computes shortest paths of lengths 1,2,…,n between all pairs of vertices.

- *shortestpath(s,t,0) = 0, if s = t, infinity, otherwise*
- *shortestpath(s,t,i+1) = min (shortestpath (s,t,i), min_{k} shortestpath(s,k,i) + W[k,t])*

This is not the same as the Floyd-Warshall calculation. If we continue this for path lengths greater than n and the value of a shortest path decreasee, we would have identified a negative cycle.

Accepted Answers:
This pseudocode implements the Floyd-Warshall algorithm.

4) Suppose we have a weighted undirected graph with negative edge weights. Which of the following is correct? ***2 points***

- Neither Krusksl's algorithm nor Prim's algorithm can be used to compute an MCST.
- Kruskal's algorithm will compute a valid MCST but Prim's algorithm will not.
- Prim's algorithm will compute a valid MCST but Kruskal's algorithm will not.
- (•) Both Krusksl's algorithm and Prim's algorithm can be used to compute an MCST.

Yes, the answer is correct.
Score: 2

Feedback:
Both Kruskal's algorithm and Prim's algorithm are unaffected by negative edge weights.

Accepted Answers:

Both Krusksl's algorithm and Prim's algorithm can be used to compute an MCST.

5) Suppose we run Prim's algorithm and Kruskal's algorithm on a graph G and the two algorithms produce minimum-cost spanning trees T_P and T_K, respectively, and T_P is different from T_K. Which of the following must be true? ***2 points***

- All edges in G have the same weight.
- Some pair of edges in G have the same weight.
- If e is a minimum cost edge in G, e belongs to both T_P and T_K.
- If e is a maximum cost edge in G, e belongs to neither T_P nor T_K.

Yes, the answer is correct.
Score: 2

Feedback:
If edge weights are distinct, the spanning tree is unique. As a counterexample for the first two options, consider a triangle with all edge weights equal. For the third option, consider a square with opposite edge having equal weights, but adjacent edges not equal.

Accepted Answers:
Some pair of edges in G have the same weight.

End

Unit 17 - Week 5 Quiz

Week 5 Quiz

The due date for submitting this assignment has passed. **Due on 2019-03-06, 23:59 IST.**

Assignment submitted on 2019-03-06, 23:11 IST

All questions carry equal weightage. You may submit as many times as you like within the deadline. Your final submission will be graded.

1) In which of the following scenarios is amortized complexity likely to be a more accurate measure of algorithmic efficiency? ***2 points***

- ( ) Worst case cost of each operation is independent of the other operations.
- (•) Worst case cost of an operation depends on the preceding sequence of operations.
- ( ) Average cost of an operation across random inputs can be computed.
- ( ) Inputs are constrained to come in a sequence that reduces the overall worst case complexity.

Yes, the answer is correct.
Score: 2

Feedback:
Amortized complexity finds the average cost of an operation by computing the total cost across a sequenc and dividing by the length of the sequence. This makes sense when the costs of individual operations var in a predictable way based on the history of the operations.

Accepted Answers:
Worst case cost of an operation depends on the preceding sequence of operations.

2) In the Union-Find data structure, suppose we want to add an operation `reassign(j,k)` that moves item `j` to the same component as item `k`. What would be the complexity of this operation for a set with n elements? ***2 points***

- ( ) O(n) for both the array representation and the pointer representation.
- (•) O(1) for the array representation and O(log n) for the pointer representation.
- ( ) O(n) for the array representation and O(log n) for the pointer representation.
- ( ) O(1) for both the array representation and the pointer representation.

No, the answer is incorrect.
Score: 0

Feedback:
In the array representation, we have to assign Component[j] the same value as Component[k]. In the pointer representation, we look up Node[j] and Node[k] and copy the parent pointer from Node[k] to Node So both updates take time O(1).

Accepted Answers:
O(1) for both the array representation and the pointer representation.

3) In a **min-heap**, what is the most accurate description of the worst-case complexity of the operation `find_max` that reports the value of the largest element in the heap, without removing it? ***2 points***

- O(n log n)
- O(n)
- O(log n)
- O(1)

Yes, the answer is correct.
Score: 2

Feedback:
The min-heap structure does not help us find the maximum value in any way, so the simplest optio scan the heap as an unsorted array and return the maximum value in O(n) time.

Accepted Answers:
O(n)

4) After inserting 57 into a max-heap, the structure of the heap is [82,57,27,42,25,18,25,27,32]. What was the structure of the heap before inserting 57? ***2 poi***

- [82,42,32,27,25,18,25,27]
- [82,27,42,32,25,18,25,27]
- [82,42,27,25,32,18,25,27]
- [82,42,27,32,25,18,25,27]

Yes, the answer is correct.
Score: 2

Feedback:
The leaf node where 57 was inserted was the last one, which is currently 32. The path from the current position of 57 to this leaf goes via the node with value 42. So we have to undo the swap made along this path when inserting 57.

Accepted Answers:
[82,42,27,32,25,18,25,27]

5) Consider an alternative to binary search called ternary search that divides the input array into three equal parts and recursively searches one of these three segments. What can we say about the asymptotic worst case complexity of ternary search versus binary search? ***2 points***

- The complexity of ternary search is strictly better than that of binary search.
- The complexity of ternary search is the same as that of binary search.
- The complexity of binary search is strictly better than that of ternary search.
- The relative complexity of ternary and binary search depends on the distribution of values across the array.

Yes, the answer is correct.
Score: 2

Feedback:
The recurrence for ternary search is

- *T(0) = T(1) = 1*
- *T(n) = O(n) + T(n/3)*

which resolves as $T(n) = O(\log_3 n)$. However, since $\log_3 n = \log_2 n / \log_2 3$, the asymptotic complexity is the same as $O(\log_2 n)$ for binary search.

Accepted Answers:
The complexity of ternary search is the same as that of binary search.

End

Unit 20 - Week 6 Quiz

Week 6 Quiz

The due date for submitting this assignment has passed. **Due on 2019-03-13, 23:59 IST.**

Assignment submitted on 2019-03-13, 22:11 IST

All questions carry equal weightage. You may submit as many times as you like within the deadline. Your final submission will be graded.

1) Suppose we implement a binary search tree without a parent pointer in each node. So each node has a value and pointers to the left and right children. What would be the complexity of computing the successor for a node in such an implementation for a balanced search tree on n nodes? ***2 points***

- O(n) for all nodes
- O(log n) for all nodes
- O(log n) for a node with a right child, O(n) otherwise
- O(log n) for a node with a right child, O(n log n) otherwise

No, the answer is incorrect.
Score: 0

Feedback:
If we have to follow parent pointers to find succ, we can instead start a second scan from the root and ma the point where we turn left to enter the subtree containing the current node, so it takes O(log n) in all cases.

Accepted Answers:
O(log n) for all nodes

2) We have n distinct values stored in a binary search tree. Define the height of a tree to be the number of nodes in the longest path from root to leaf. Which of the following statements is ***not*** true? ***2 points***

- The height of the tree is at least log n.
- The height of the tree is at most n.
- If the root is the median value, the height of the tree is at most log n.
- If the root is the median value, the height of the tree is at most n/2.

Yes, the answer is correct.
Score: 2

Feedback:
An upper bound of log n on the height requires the tree to be balanced.

Accepted Answers:
If the root is the median value, the height of the tree is at most log n.

3) Preorder traversal prints a tree by first listing the value at the root and then recursively listing the values of the left and right subtrees. ***2 points***

```
function preOrder(t) {
  if (t != NIL) {
```

```
    print(t.value);
    preOrder(t.left);
    preOrder(t.right);
  }
}
```

What is the complexity of preorder traversal for a binary search tree with n nodes?

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

Yes, the answer is correct.
Score: 2

Feedback:
Preorder traversal visits and lists each element once, so it is O(n).

Accepted Answers:
O(n) whether the tree is balanced or unbalanced.

4) The preorder traversal of a binary search tree with integer values produces the following sequence: 35, 23, 26, 46, 40, 39, 41, 52. What is the value of the right child of the root of the tree? ***2 points***

- ( ) 39
- ( ) 40
- ( ) 41
- (•) 46

Yes, the answer is correct.
Score: 2

Feedback:
The inorder traversal of a search tree is always the sorted sequence. In this case: 23, 26, 35, 39, 40, 41, 46, 52. From the preorder traversal, we know that 35 is the root of the tree, so the segment 23, 26 corresponds to the left subtree and the segment 39, 40, 41, 46, 52 corresponds to the right subtree. The preorder traversal of the right subtree starts with 46, so this is the right child of the root node.

Accepted Answers:
46

5) Suppose we build a Huffman code over the four letter alphabet {a,b,c,d}, where f(a), f(b), f(c) and f(d) denote the frequencies (probabilities) of the letters and f(a) > f(b) > f(c) > f(d). Which of the following is a valid conclusion? ***2 points***

- ( ) The Huffman code will assign two bits to every letter always.
- ( ) The Huffman code will assign variable numbers of bits to the letters if f(a) > f(b)+f(c)+f(d) and will assign two bits to every letter otherwise.
- (•) The Huffman code will assign variable numbers of bits to the letters if f(a) > max(f(b),f(c)+f(d)) and will assign two bits to every letter otherwise.
- ( ) The Huffman code will assign variable numbers of bits to the letters if f(a) > min(f(b),f(c)+f(d)) and will assign two bits to every letter otherwise.

Yes, the answer is correct.
Score: 2

Feedback:
The first iteration will group {c,d} as {cd}. To assign a variable number of bits, the second iteration should combine {b,cd} to form {bcd}. For this, f(b) and f(c)+f(d) should be less than f(a), which means f(a) > max(f(b),f(c)+f(d)).

Accepted Answers:
The Huffman code will assign variable numbers of bits to the letters if f(a) > max(f(b),f(c)+f(d)) and will assign two bits to every letter otherwise.

End

Unit 23 - Week 7 Quiz

Week 7 Quiz

The due date for submitting this assignment has passed. **Due on 2019-03-20, 23:59 IST.**

Assignment submitted on 2019-03-20, 22:07 IST

All questions carry equal weightage. You may submit as many times as you like within the deadline. Your final submission will be graded.

You are playing an old-style video game in which you have to shoot down alien spaceships as they fly across the screen from left to right. Each spaceship flies across the screen at a specified height. You have an antiaircraft gun set to shoot down all spaceships at a certain height. Spaceships fly one at a time, so if your gun is set to fire at the correct height, it will shoot down the spaceship currently flying across the screen.

You can set the initial height at which the gun fires. As the game progresses, you can reset the height, but only to a lower value. You are given in advance the height at which each spaceship flies. There are n spaceships numbered 1, 2, … , n in the order in which they fly across the screen. For $1 \le i \le n$, h[i] denotes the height at which spaceship i flies.

1) Let V[i] denote the maximum number of spaceships from i, i+1, … , n that you can shoot down with a single gun, assuming you start by shooting V[i]. Which of the following is a valid recursive formulation of V[i]? (Note that the maximum of an empty set is defined to be 0.) ***2 points***

- V[1] = 1
  For $1 \le i \le n$, V[i] = 1 + max{V[j] | j < i and h[j] ≤ h[i]}
- V[1] = 1
  For $1 \le i \le n$, V[i] = 1 + max{V[j] | j < i and h[i] ≤ h[j]}
- V[n] = 1
  For $1 \le i \le n$, V[i] = 1 + max{V[j] | j > i and h[i] ≥ h[j]}
- V[n] = 1
  For $1 \le i \le n$, V[i] = 1 + max{V[j] | j > i and h[j] ≥ h[i]}

Yes, the answer is correct.
Score: 2

Feedback:
The problem is the same as finding the longest descending subsequence.

Accepted Answers:
V[n] = 1
For $1 \le i \le n$, V[i] = 1 + max{V[j] | j > i and h[i] ≥ h[j]}

2) What is the size of the memo table for this problem? ***2 points***

- n-1
- n
- n+1

- n^2

Yes, the answer is correct.
Score: 2

Feedback:
One entry per spaceship, n spaceships

Accepted Answers:
n

3) What is a good order to compute the values of V[i] using dynamic programming? ***2 poi***

- (•) From V[n] to V[1]
- ( ) From V[1] to V[n]
- ( ) Either from V[n] to V[1] or from V[1] to V[n]
- ( ) None of these

Yes, the answer is correct.
Score: 2

Feedback:
V[i] depends on V[j], i < j ≤ n

Accepted Answers:
From V[n] to V[1]

4) How much time will it take to compute V[1] using standard dynamic programming? ***2 points***

- (•) O(n)
- ( ) O(n log n)
- ( ) $O(n^2)$
- ( ) $O(n^2 \log n)$

No, the answer is incorrect.
Score: 0

Feedback:
Computing V[i] requires scanning V[i+1] to V[n] so it takes time O(n-i). Hence, the overall time is 1+2+...+n-1 or $O(n^2)$

Accepted Answers:
$O(n^2)$

5) Suppose we have 30 spaceships whose heights are as follows. ***2 points***

[71, 68, 98, 46, 33, 48, 51, 5, 54, 28, 28, 92, 65, 15, 41, 63, 21, 34, 84, 98, 70, 44, 58, 91, 12, 67, 61, 27, 91, 6]

What is the maximum number of spaceships we can shoot down starting from the first one?

- ( ) 10
- (•) 9
- ( ) 8
- ( ) 7

Yes, the answer is correct.
Score: 2

Feedback:
9. For instance, 1-2-4-5-10-11-14-25-30

Accepted Answers:
9

End

Name:

Advanced Programming, II Semester, 2014–2015

Quiz 7, 15 April 2015

Answer all questions in the space provided. Use the reverse for rough work, if any.

At the end of its fifth successful season, the Siruseri Premier League is planning to give an award to the Most Improved Batsman over the five years. For this, an Improvement Index will be computed for each batsman. This is defined as the longest subsequence of strictly increasing scores by the batsman among all his scores over the five seasons. For example, if the scores for a batsman over the five seasons are $[20, 23, 6, 34, 22, 52, 42, 67, 89, 5, 100]$, his improvement index is 7 based on the subsequence $[20, 23, 34, 52, 67, 89, 100]$.

1. let $S[1..n]$ denote a sequence of n scores for which the improvement index is to be calculated. For $1 \leq j \leq n$, let $I(j)$ denote the improvement index for the prefix of scores $S[1..j]$ ending at $S[j]$.

   Which of the following is a correct recursive formulation of $I(j)$?

   (a) $I(1) = 1$
   For $j \in 2, 3, \ldots, n$, $I(j) = 1 + \max\{I(k) \mid 1 \leq k < j, S[k] > S[j]\}$

   (b) $I(1) = 1$
   For $j \in 2, 3, \ldots, n$, $I(j) = 1 + \max\{I(k) \mid 1 \leq k < j, S[k] < S[j]\}$

   (c) $I(1) = 1$
   For $j \in 2, 3, \ldots, n$, $I(j) = \begin{cases} 1 + S[j-1], & \text{if } S[j-1] < S[j] \\ 1, & \text{otherwise} \end{cases}$

   (d) $I(1) = 1$
   For $j \in 2, 3, \ldots, n$, $I(j) = \begin{cases} 1 + S[j-1], & \text{if } S[j-1] > S[j] \\ 1, & \text{otherwise} \end{cases}$

   Answer: (b) Look for the longest sequence that can be extended by $S[j]$.

   (4 marks)

2. How would we evaluate this recursive definition using dynamic programming?

   (a) A one dimensional table t of size n, to be filled from $t[1]$ to $t[n]$

   (b) A one dimensional table t of size n, to be filled from $t[n]$ to $t[1]$

   (c) A two dimensional table t of size $n \times n$, to be filled row-wise from $t[1][1]$ to $t[n][n]$.

   (d) A two dimensional table t of size $n \times n$, to be filled row-wise from $t[n][n]$ to $t[1][1]$.

   Answer: (a) The recursive function has a single argument, so we need only a one dimensional table. The base case is $I(1)$, so start with $T[1]$ and work up to $T[n]$.

   (3 marks)

3. How much time will it take to evaluate this recursive definition using dynamic programming?

   (a) $O(n)$

   (b) $O(n \log n)$

   (c) $O(n^2)$

   (d) $O(n^3)$

   Answer: (c) Each entry $I(j)$ requires scanning $I(1)$ to $I(j-1)$, so the time taken is $1 + 2 + \cdots + n - 1$ which is $O(n^2)$.

   (3 marks)

Name:

Advanced Programming, II Semester, 2014–2015

Quiz 6, 6 April 2015

Answer all questions in the space provided. Use the reverse for rough work, if any.

1. Consider the following strategy to solve the single source shortest path problem with positive integer edge weights from source s.

   - Replace each edge with weight w by w edges of weight 1 connected by new intermediate nodes.
   - 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?

   (a) This strategy will not solve the problem correctly.
   (b) This strategy will solve the problem correctly and is as efficient as Dijkstra's algorithm.
   (c) This strategy will solve the problem correctly but is not as efficient as Dijkstra's algorithm.
   (d) This strategy will only work if the graph is connected.

   (5 marks)

   Answer: (c) The size of the graph blows up according to the edge weights, but the strategy is otherwise correct.

2. 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. 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+list and tree implementations of Union-Find?

   (a) Array+list representation: $O(n)$, Tree representation: $O(n)$
   (b) Array+list representation: $O(size(c))$, Tree representation: $O(n)$
   (c) Array+list representation: $O(n)$, Tree representation: $O(size(c))$
   (d) Array+list representation: $O(size(c))$, Tree representation: $O(size(c))$

   (5 marks)

   Answer: (b) In the array+list representation we have the list of members of c which allows us to update the contents of c in time $O(size(c))$. In the tree representation there is no easy way to identify all elements that belong to component c without scanning the entire set, so it takes time $O(n)$.

Name:

Advanced Programming, II Semester, 2014–2015

Quiz 5, 30 March 2015

Answer all questions in the space provided. Use the reverse for rough work, if any.

---

1. Recall that Dijkstra's algorithm to compute the shortest path from a source vertex s to every other vertex keeps track of two quantities: the time at which each already visited vertex was "burnt" and the expected burning time for all as yet unburnt vertices. The main loop of the algorithm repeats the following steps till all the vertices are burnt:

   - Burn the unburnt vertex with minimum expected burning time.
   - Update the expected burning time of all neighbours of the vertext just burnt.

   Let $G = (V, E)$ with $|V| = n$ and $|E| = m$. What is the complexity of Dijkstra's algorithm in each of the following implementations, in terms of n and m?

Representation of G	*Data structure for expected burning time*	*Worst-case Complexity*
Adjacency Matrix	Array	$O(n^2)$
Adjacency List	Array	$O(n^2)$
Adjacency Matrix	Heap	$O(n^2 + m \log n)$ or $O(n^2 \log n)$
Adjacency List	Heap	$O((m+n) \log n)$

(5 marks)

2. We have a connected graph with edge weights with n vertices and m edges. We execute the following algorithm. Start with the original graph. Consider each edge e_j in decreasing order of cost. If this edge is part of a cycle delete it.

   (a) What does the algorithm achieve?

   Constructs a minimum-cost spanning tree.

   (b) How many edges will the algorithm delete?

   The final tree has $n-1$ edges, so $m-(n-1)$ edges will be deleted.

(5 marks)

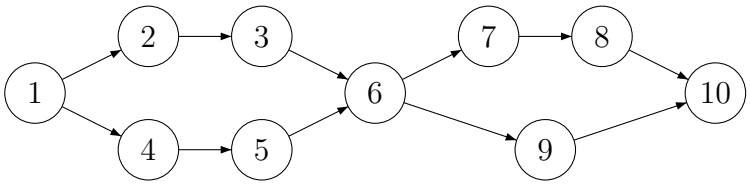
Name:

Advanced Programming, II Semester, 2014–2015

Quiz 4, 18 March 2015

Answer all questions in the space provided. Use the reverse for rough work, if any.
Don't forget to fill your name!

1. How many topological orderings does the following directed acyclic graph have?

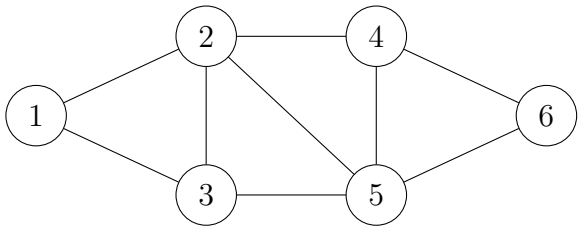


In any valid linearization, the vertices {2,3,4,5} must come between 1 and 6 and the vertices {7,8,9} must come between 6 and 10. If we fix the positions of {2,3}, the positions of {4,5} are decided. Likewise if we fix the positions of {7,8}, the position of 9 is decided. This gives us the following calculation for the total number of orderings.

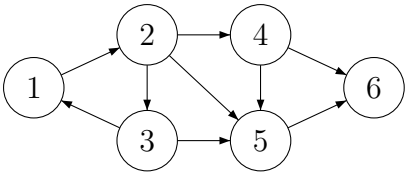
$$\binom{4}{2} \times \binom{3}{2} = 6 \times 3 = 18$$

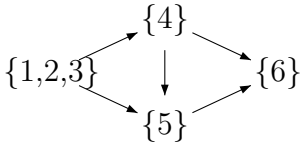
(5 marks)

2. Orient the edges of the following undirected graph so that the resulting directed graph has *four* strongly connected components. Draw the corresponding scc dag—label each vertex of the dag with the corresponding scc.



The smallest nontrivial scc one can generate is a triangle. This means that the other three scc's must be singletons. You can orient any triangle as an scc and set up the other edges appropriately. For instance:





(5 marks)

Name:

Advanced Programming, II Semester, 2014–2015

Quiz 3, 11 March 2015

Answer all questions in the space provided. Use the reverse for rough work, if any.
Don't forget to fill your name!

1. Let $G = (V, E)$ with $|V| = n$ and $|E| = m$. What is the worst-case complexity of:

   (a) BFS using an adjacency matrix representation of G. $O(n^2)$

   (b) BFS using an adjacency list representation of G. $O(n + m)$

   (c) DFS using an adjacency matrix representation of G. $O(n^2)$

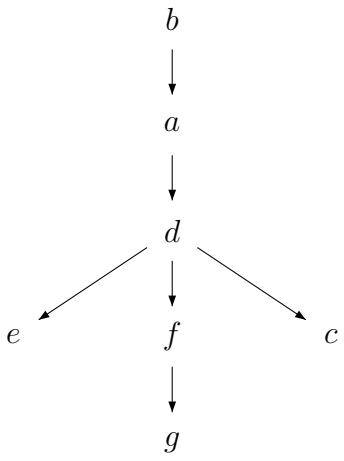
   (d) DFS using an adjacency list representation of G. $O(n + m)$

   (2 marks)

2. Given the following data about DFS on a directed graph, reconstruct the DFS tree.

Vertex	a	b	c	d	e	f	g
Entry(v)	2	1	10	3	4	6	7
Exit(v)	13	14	11	12	5	9	8

(5 marks)



3. Here are three non-tree edges in the graph of the previous question. Classify them as forward/backward/cross.

   (a) (c, b) — backward edge
   (b) (f, e) — cross edge
   (c) (d, g) — forward edge

   (3 marks)

Name:

Advanced Programming, II Semester, 2014–2015

Quiz 2, 2 March 2015

Answer all questions in the space provided. Use the designated space for rough work, if any.
Don't forget to fill your name!

1. Complete the following function definition so that it behaves as described below—that is, fill in the parameters for `f()` in the correct order, with default values, as appropriate.

```
def f(.........):
  print("a",a,"b",b,"c",c,"d",d)
```

Expected behaviour:

```
>>> f(b=4,a=3)
a 3 b 4 c 10 d 15

>>> f(3,5,7)
a 3 b 5 c 7 d 15

>>> f(3,c=7)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: f() takes at least 2 arguments (2 given)
```

(4 marks)

Solution:

```
def f(a,b,c=10,d=15):
```

- The first line gives us the default values for `c` and `d`.
- The second line tells us the positions for `a` and `b`.
- The third line tells us that neither `a` nor `b` has default values.

Rough work

2. The function `minout(inl)` takes a list `inl` of distinct natural numbers (i.e., integers from the set {0,1,2,...}) and returns the smallest natural number not present in `inl`. In other words, if 0 is not in `inl`, then `minout(inl)` returns 0, else if 0 is in `inl` but 1 is not in `inl`, them `minout(inl)` returns 1, etc.

   For example, `minout([1,3,2,4,17])` is `0` and `minout([1,3,0,2,4])` is `5`.

   Here is a recursive algorithm to compute `minout`. (Note that `inl` is *not* assumed to be sorted, nor do we make any assumptions about the range of values in `inl`.)

   - Suppose the length of `inl` is n. Construct two lists `lower` and `upper`, where `lower` contains elements smaller than $\lfloor \frac{n}{2} \rfloor$ and `upper` contains the rest.
   - If the size of `lower` is strictly smaller than $\frac{n}{2}$, the missing number is smaller than $\frac{n}{2}$, so recursively invoke `minout` on `lower`.
   - Otherwise, invoke `minout` on `upper`. (All numbers in `upper` are bigger than $\frac{n}{2}$, so some offset is required.)

   Analyze the running time of this algorithm. (Write a recurrence and solve it.) *(6 marks)*

   Solution:

   The recurrence is:

   - $T(1) = 1$
   - $T(n) = T\left(\frac{n}{2}\right) + n$

   Unwinding this we get $n + \frac{n}{2} + \frac{n}{4} + \ldots 1 = O(n)$.

Rough work