FA - Quizzes

To implement a circular queue efficiently using sequential allocation, the minimal necessary requirements are to use:

a. An array, a size field and an index field;
b. an array and an index field;
c. an array, a size field and two index fields;
d. an array and a size field.

To implement a queue efficiently using a linked list, the minimal necessary requirements are to use:

a. a doubly linked list, with pointer to first element
b. a singly linked list, with pointer to the first element
c. a DLL, with pointer to first and last elements
d. a SLL, with pointer to the first and the last elements

To implement a stack efficiently using a linked list, the minimal necessary requirements are to use:

e. a doubly linked list, with pointer to first element
f. a singly linked list, with pointer to the first element
g. a DLL, with pointer to first and last elements
h. a SLL, with pointer to the first and the last elements

Consider a binary search tree. Searching for a path between two given keys can be done in (choose the best possible complexity):

a. O(nlogn)
b. O(n)
c. O(h)
d. $O(n^2)$

The number of leaf nodes in a heap of n elements is:

a. 2^n
b. n/2
c. n/6
d. $\log_2 n$

Assume your machine can execute 100.000 MIPS (million instructions per second). What order of magnitude does the size of the problem that a quadratic algorithm can solve in 1 second have?

- 100.000 MIPS → $10^5 * 10^6 = 10^{11}$ instructions per second
- 1 hour = 3600 second → $3.6 * 10^3 * 10^{11} = 3.6 * 10^{14}$
- $n^2 = 3.6 * 10^{14}$ → $n = \text{sqrt}(3.6 * 10^{14})$ → $n \sim 10^8$

Assume your machine can execute 100.000 MIPS (million instructions per second). In which of the following ranges does the size of the problem that an exponential algorithm (assume 10^n) can solve in 1 second belong to?

a. between 50 and 100 elements
b. between 10 and 20 elements
c. between 1 and 10 elements
d. between 20 and 50 elements

For an exponential running time algorithm solving a given problem, the max size of the problem to be solved in a million years is:

a. between 2000 and 3000
b. around 1000
c. between 200 and 300
d. below 100

Assume your machine can execute 100.000 MIPS (million instructions per second). What order of magnitude does the size of the problem that a linear algorithm can solve in 1 hour have?

a. 10^{11}
b. 10^5
c. 10^8
d. 10^{14}

Choose the increasing order for the following Big Oh functions:

a. O(logn) < O(n) < O(log logn)
b. O(log logn) < O(logn) < O(n)
c. O(log logn) < O(n) < O(logn)
d. neither option is true

Choose the increasing order for the following Big Oh functions:

a. $O(n\lg^2 n) < O(n\lg n) < O(n^2 \lg n)$
b. $O(n\lg^2 n) < O(n^2 \lg n) < O(n\lg n)$
c. $O(n\lg n) < O(n\lg^2 n) < O(n^2 \lg n)$
d. neither option is true

Binary insertion sort is stable only if:

a. There are no duplicates
b. The search find the rightmost element equal with the elements to be inserted
c. never. Cannot be made stable.
d. For any implementation. No constraint.

One possible sorting strategy could be generating all possible sequences which result from the input data. It would run in:

a. $O(n^3)$

b. O(n!)
c. neither option is true
d. $O(2^n)$

The upper bound (big Oh) notation is related to:

a. algorithm and implementation
b. algorithm, implementation and data
c. algorithm and data
d. problem and algorithm

For a hash table T having 2000 slots, which stores 25 elements, the value of alpha is:

a. 0.8
b. 1.25
c. 80
d. 0.0125

What complexity has the optimum algorithm for selecting an unordered array's median?

a. O(n)
b. $O(n^2)$
c. O(logn)
d. O(nlogn)

With open addressing:

a. We solve collisions by placing all colliding keys in a linked list
b. we store the elements in the array
c. two keys which collide cannot be both inserted
d. expected (average case) search time is O(n)

Prim's algorithm applies a … strategy:

a. divide and conquer
b. greedy
c. dynamic programming
d. brute force

Linear insertion sort is stable only if:

a. For any implementation. No constraint.
b. choose to insert on the right of the rightmost elements equal with the elements to be inserted.
c. never. Cannot be made stable.
d. There are no duplicates

Given a heap which implements parent <= children relation, the second smallest element in the heap is:

a. the right child of the root
b. either the left or the right child of the root
c. the left child of the root
d. not known

The number of nodes at depth i in a heap (assuming i is not the last level, root is at depth 0) is:

a. $2^{i+1}-1$
b. i
c. i + 1
d. 2^i

What kind of strategy does heapsort apply?

a. none of the above
b. interchange
c. selection
d. insertion

Among the following, which does NOT represent a total order relation in a heap?

a. the root with left child and child's right child
b. the root with its children
c. the leftmost branch
d. the rightmost branch

Randomization is preferred to optimal selection (AklSelection) in practical implementations of QuickSort because:

a. Optimal selection cannot be implemented in practice
b. it is very efficient and easy to implement compared to optimal selection, and reduces the probability of reaching the worst case to almost 0
c. it is easy to implement compared to optimal selection
d. optimal selection does not guarantee finding the median

Given an array of n keys, the best case running time for selection sort if only assignments are considered is:

a. $O(n^2)$
b. O(1)
c. O(logn)
d. O(n)

This is the preorder of a BST (Binary Search Tree): 5, 4, 2, 3, 7. The lowest level where the tree is NOT a PBT is at key 4.

This is the preorder of a BST (Binary Search Tree): 3, 2, 7, 4, 5. The lowest level where the tree is NOT an AVL is at key:

a. 4
b. 7
c. 3
d. 5

Let X be a node with 2 children in a BST. If Y is the predecessor of X, which of the following is true?

a. Y is a leaf
b. Y is the minimum in X's left subtree
c. Y has no right child
d. Y has no left child

Let X be a node with 2 children in a BST. If Y is the successor of X, which of the following is true?

a. Y has no left child
b. Y is the minimum in X's left subtree
c. Y is a leaf
d. Y has no right child

For an exponential running time algorithm solving a given problem of size greater than 100 the approach would be:

a. Parallelization with at least 10 processors
b. Parallelization with at least 100 processors
c. Neither
d. Parallelization with at least 1000 processors

This is the preorder of a BST (Binary Search Tree): 8, 4, 2, 6, 10. The tree is NOT a PBT because of the tree rooted at 8.

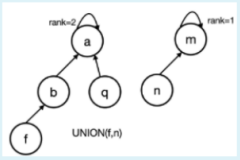
What value could have the missing element (the underscore) such that the following is a max-heap: 9, _, 2, 6, 4, 1

a. 11
b. 5
c. 6
d. 3

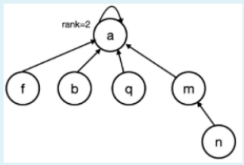
For the QuickSort version where the middle element is selected as a pivot, an array that is already sorted is:

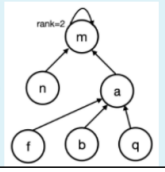
a. worst case
b. average case
c. best case
d. cannot be decided

Given the following disjoint set forest, what will it look like after the operation UNION(f,n) is performed on it? (considering both union by rank and path compression heuristics are implemented)

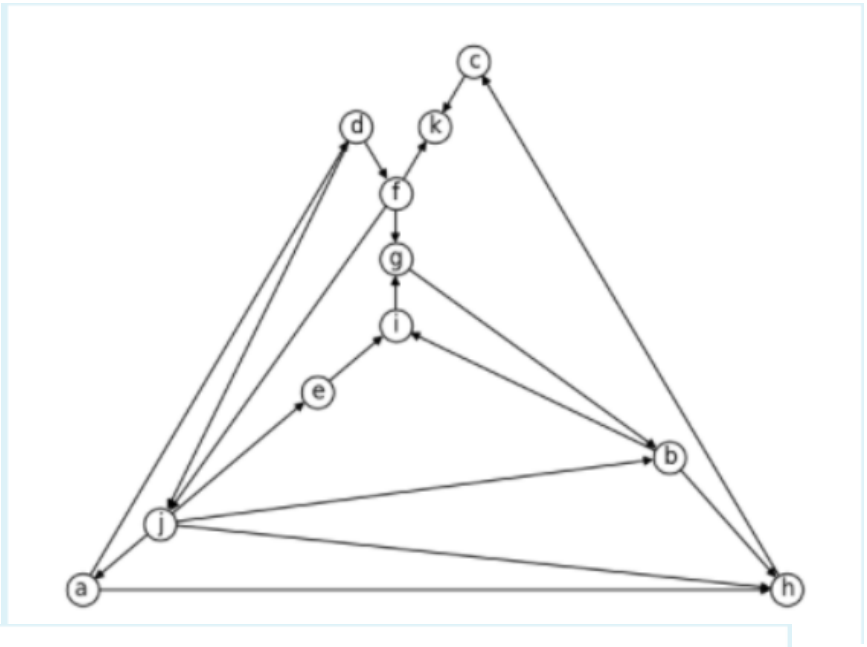


Select one:

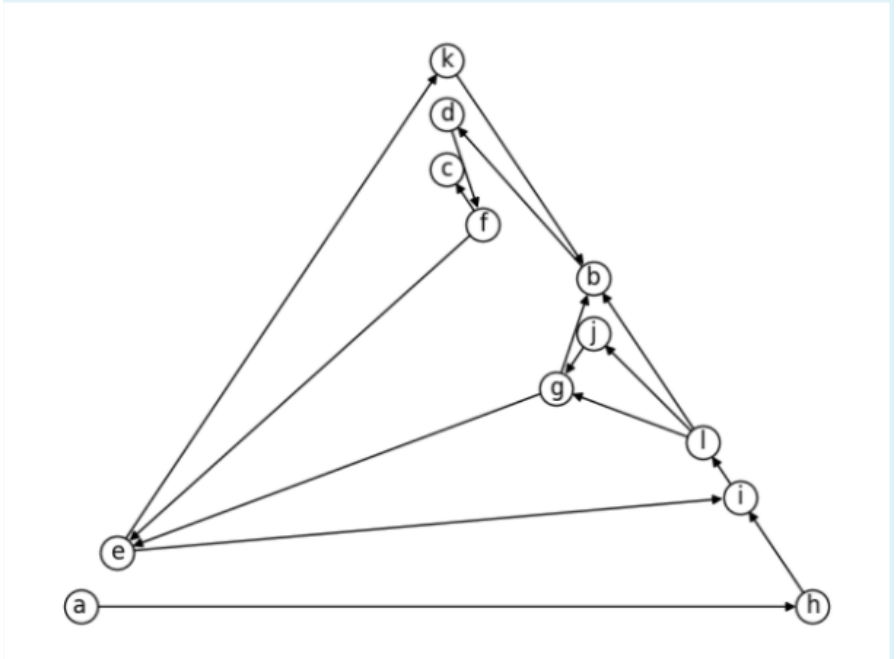
a.


b.


The number of strongly connected components of the graph below is:

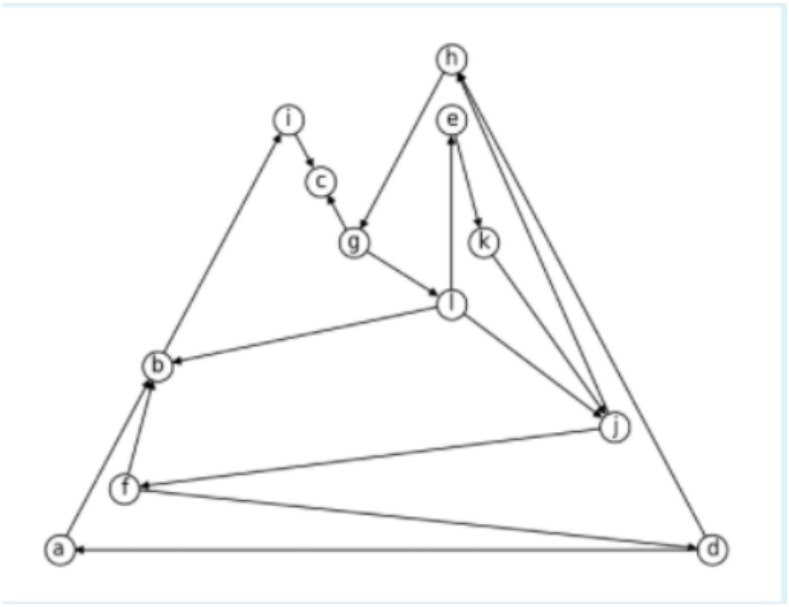


a. 5
b. 4
c. 6
d. 3



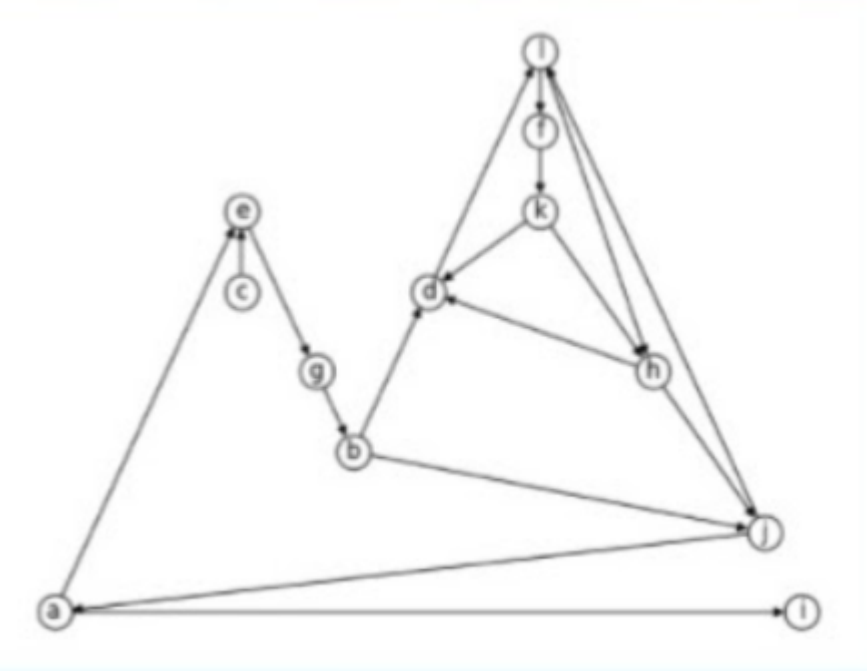
The largest strongly connected component of the graph below contains:

a. 9 nodes
b. 5 nodes
c. 8 nodes
d. 6 nodes



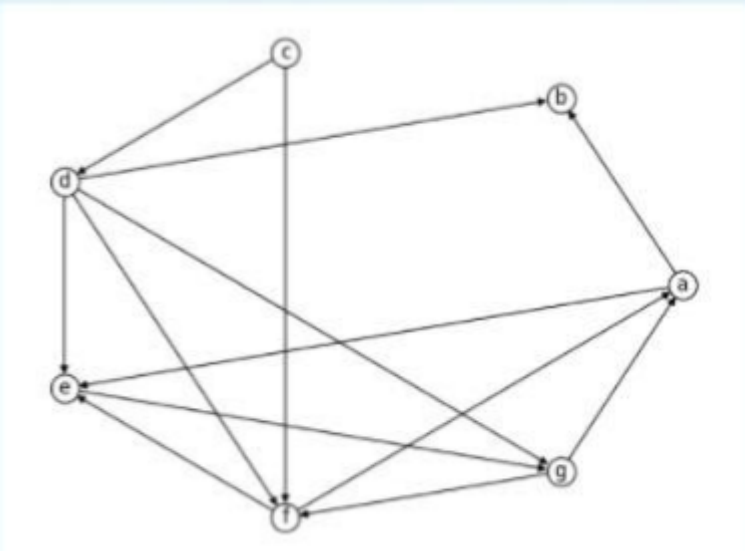
The number of strongly connected components of the graph below is:

a. 4
b. 5
c. 2
d. 3



The largest strongly connected component of the graph below contains:

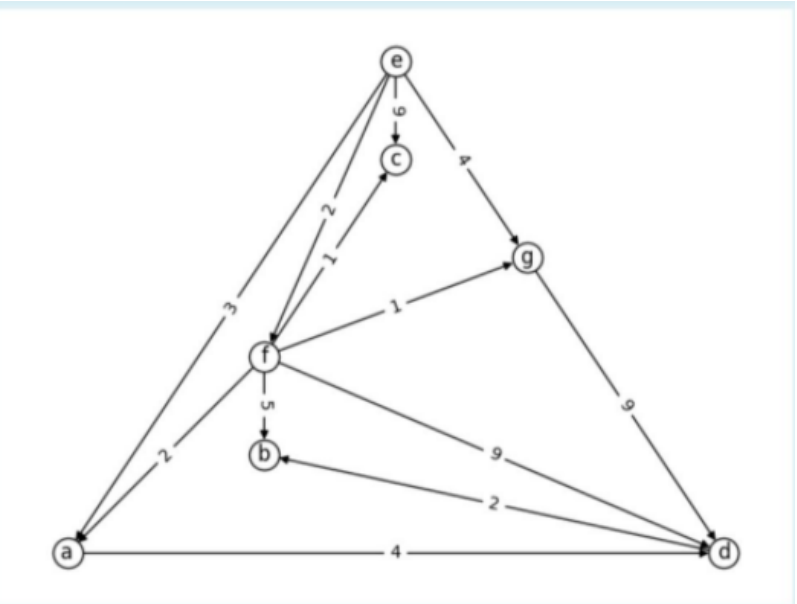
a. 5 nodes
b. 6 nodes
c. 10 nodes
d. 8 nodes



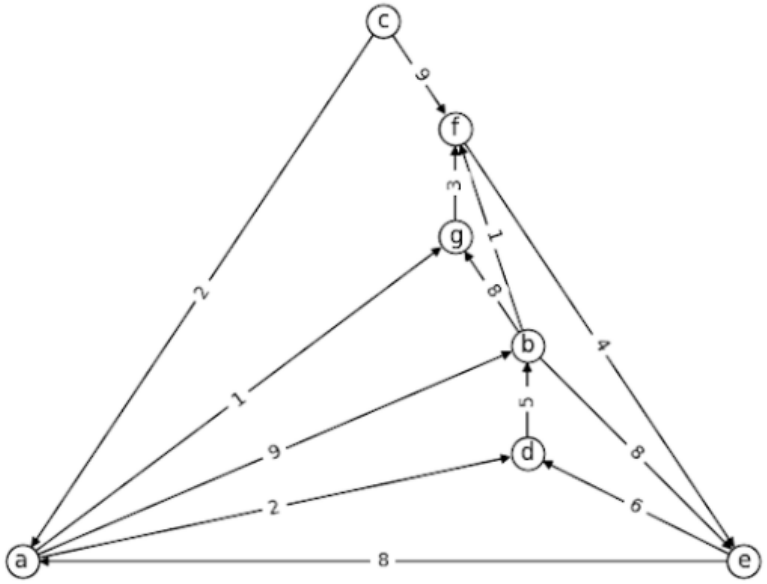
The number of strongly connected components of the graph below is:

a. 2
b. 5
c. 4
d. 3

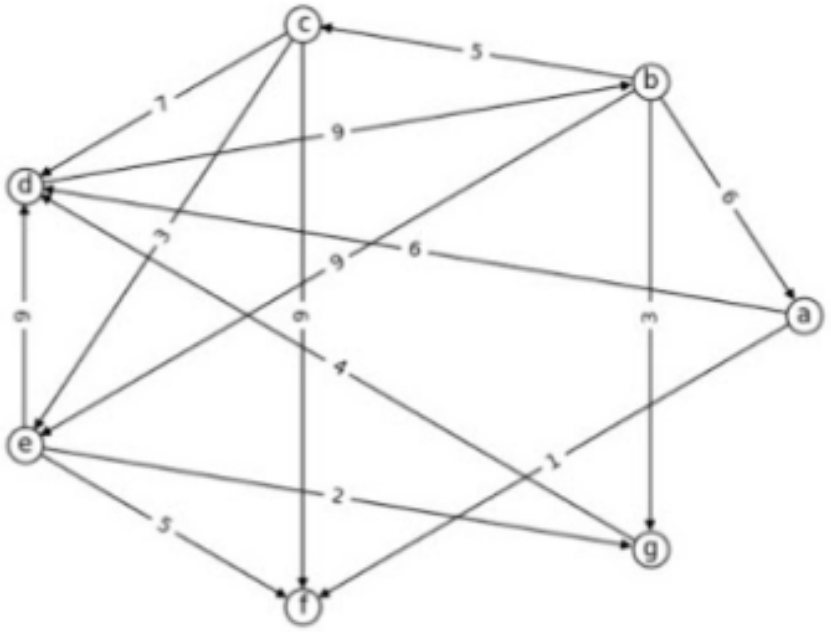
In the graph above, Dijkstra's algorithm is called for the source node e. Fill in the table below the parent (first field) and the distance (second field) for each node in the graph:



node	parent	dist
a	**e**	**3**
b	**f**	**7**
c	**f**	**3**
d	**a**	**7**
e	**nil**	**0**
f	**e**	**2**
g	**f**	**3**



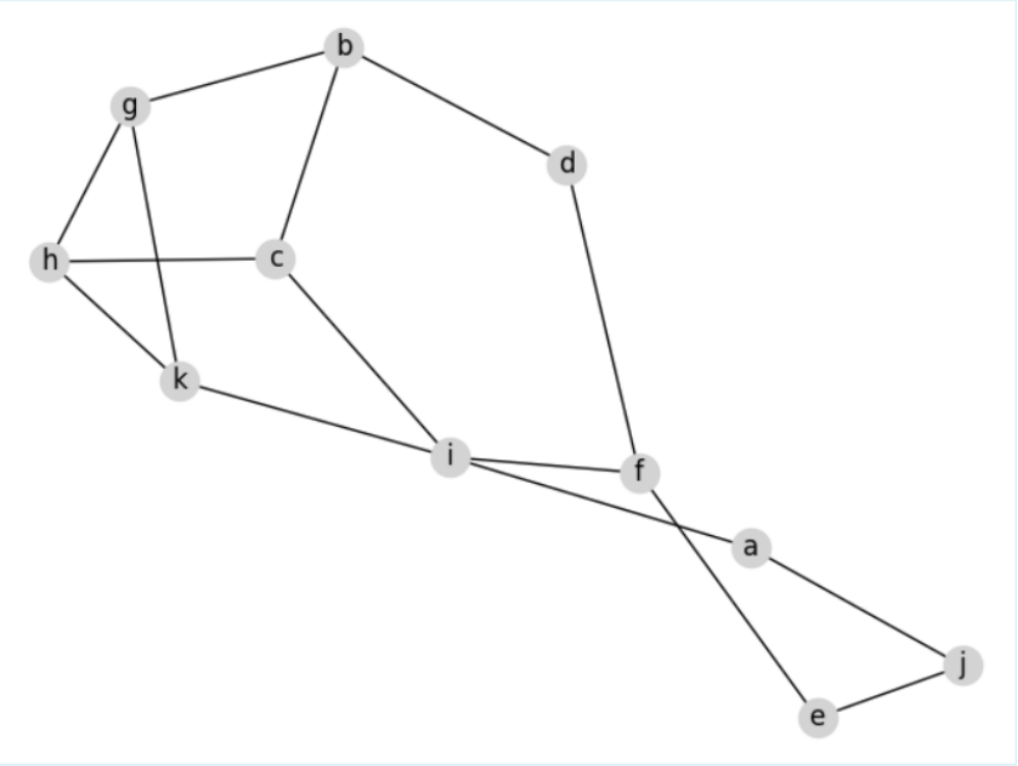
node	parent	dist
a	**c**	**2**
b	**d**	**9**
c	**nil**	**0**
d	**a**	**4**
e	**f**	**10**
f	**g**	**6**
g	**a**	**3**



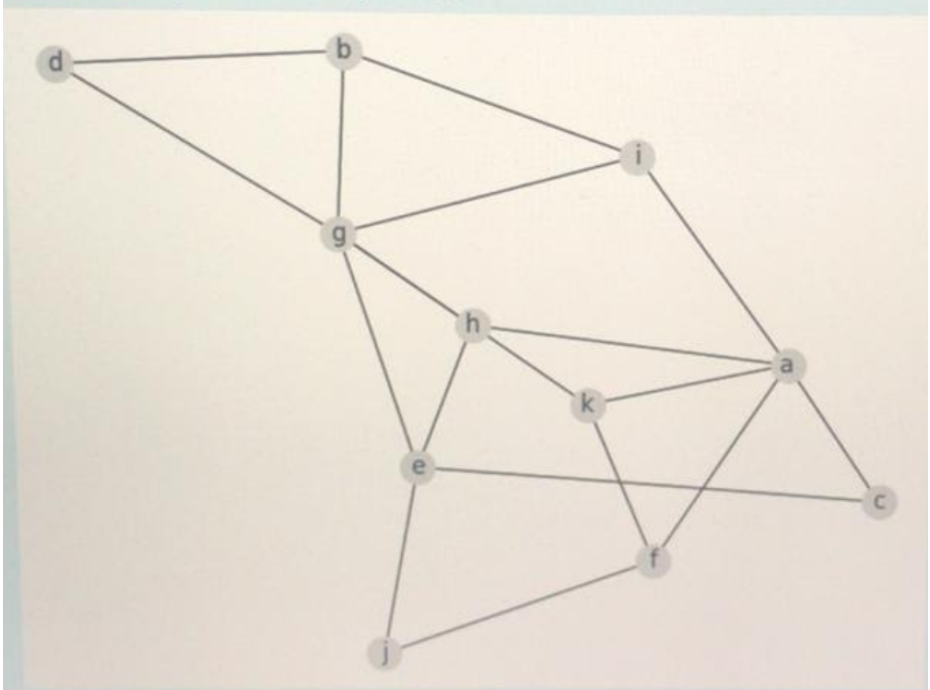
node	parent	dist
a	**b**	**15**
b	**d**	**9**
c	**b**	**14**
d	**nil**	**0**
e	**c**	**17**
f	**a**	**16**
g	**b**	**12**

e.

Considering the graph below, what is the last node that gets colored BLACK in a BFS traversal that starts with node a? The neighbors for each node will be considered in lexicographic order (alphabetically). Please type a single letter indicating the node:



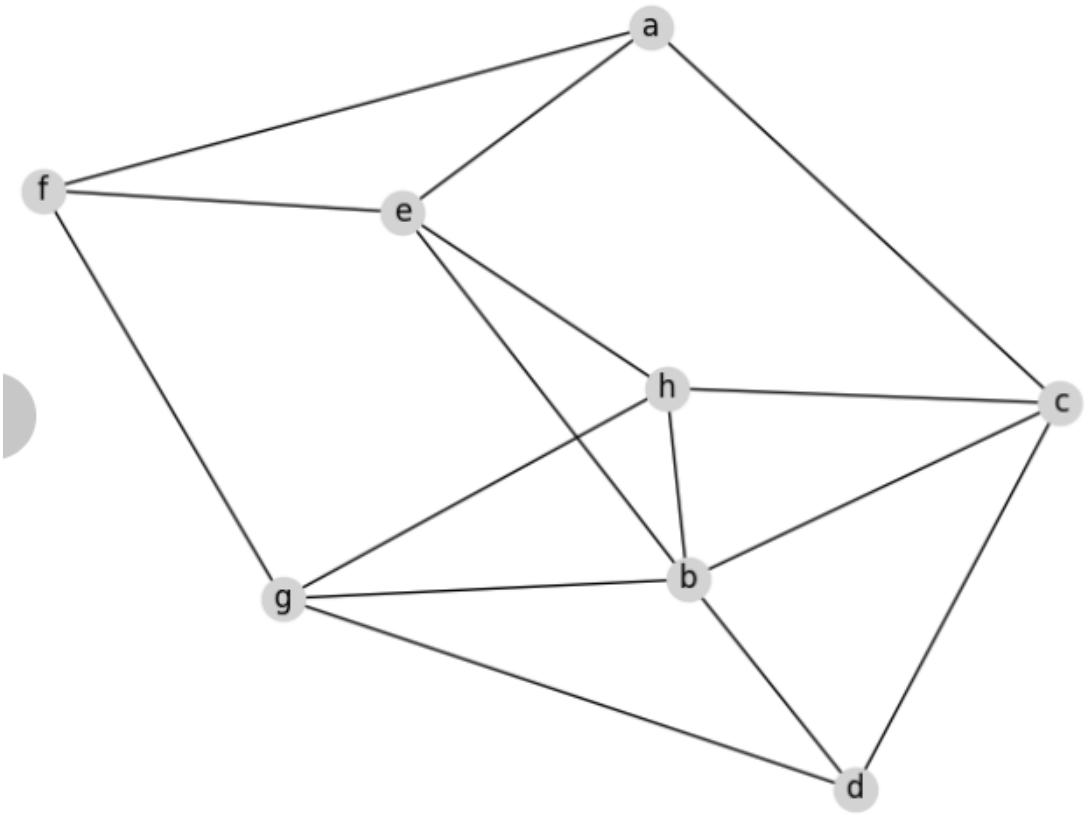
answer: g



answer: f

starting with node g:

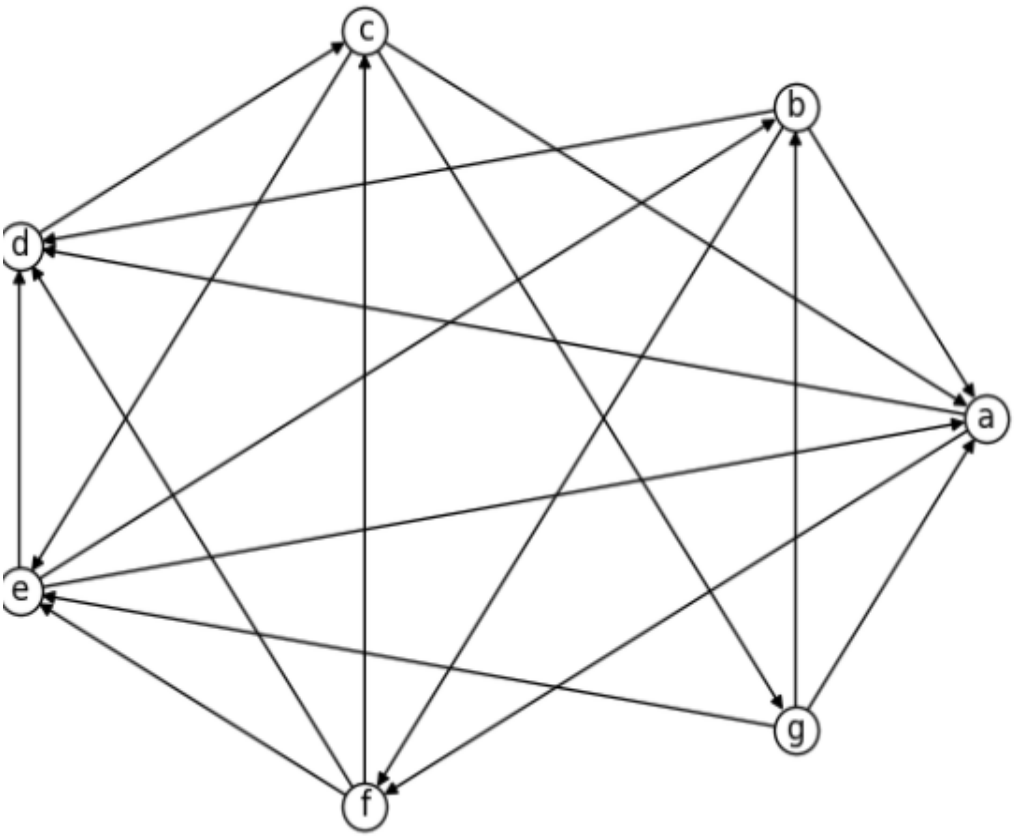
Answer: a

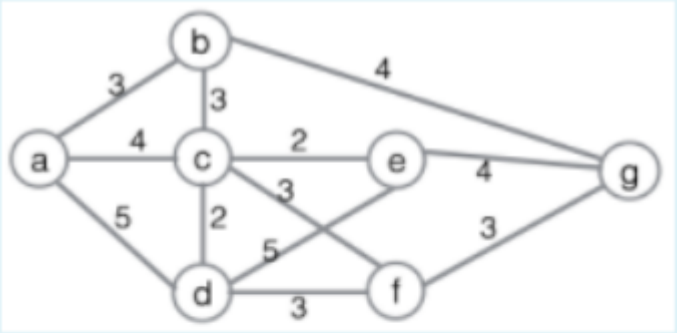


Considering the graph below, what is the last node that gets colored BLACK in a DFS traversal that starts with node e? The neighbors for each node will be considered in lexicographic order (alphabetically). Please type a single letter indicating the node:

Answer: node e

we go through the nodes like this: e, a, d, c, g, b, f





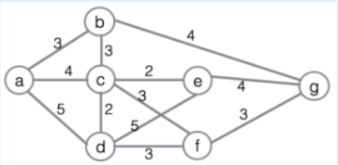
Consider the graph below. Which of the following CANNOT be the sequence of edges added, in that order, to a minimum spanning tree using Kruskal's algorithm?

c. (c, e), (c, d), …

Select one:

- a. (c,d), (c,e), (f,g), (a,b), (c,f), (b,c)
- b. (c,d), (c,e), (c,f), (f,g), (a,b), (b,c)
- c. (c,e), (c,d), (f,g), (b,c), (c,f), (d,f)
- d. (c,e), (c,d), (f,g), (b,c), (a,b), (d,f)

Consider the graph below. Which of the following CANNOT be the sequence of edges added, in that order, to a minimum spanning tree using Prim's algorithm?



Select one:

- a. (f,g), (c,e), (c,d), (d,f), (b,c), (a,b)
- b. (f,c), (c,d), (c,e), (b,c), (a,b), (f,g)
- c. (f,d), (c,d), (c,e), (f,g), (b,c), (a,b)
- d. (f,g), (d,f), (c,d), (c,e), (b,c), (a,b)

a. (f, g), (c, e), (c, d), (d, f), (b, c), (a, b)

What is the best-case complexity of Kruskal's MST algorithm for a graph with |V| vertices and |E| edges, if the edges are already sorted? Consider that union and find operations take amortized O(1) time.

a. O(V+E)
b. O(VE)
c. O(V)
d. O(E)

This is the preorder of a BST (Binary Search Tree): 3, 1, 2, 4, 5. Which of the following statements is true?

a. Neither AVL nor PBT
b. AVL but not PBT
c. Both AVL and PBT
d. Not AVL but PBT

This is the preorder of a BST (Binary Search Tree): 3, 1, 2, 5, 4. Which of the following statements is true?

a. AVL but not PBT
b. Not AVL but PBT
c. Neither AVL nor PBT
d. Both AVL and PBT

This is the preorder of a BST (Binary Search Tree): 4, 2, 3, 5. The tree is NOT a PBT because of the tree rooted at:

a. But it IS a PBT
b. 4
c. 2
d. 5

This is the preorder of a BST (Binary Search Tree): 7, 3, 1, 5, 9. The tree is NOT a PBT because of the tree rooted at:

a. 7
b. 5
c. 9
d. 3

This is the preorder of a BST (Binary Search Tree): 4, 3, 2, 5. The tree is NOT an AVL because of the tree rooted at:

a. 5
b. But it IS an AVL
c. 2
d. 4

This is the preorder of a BST (Binary Search Tree): 3, 1, 2, 4. Which of the following statements is true?

a. Not AVL but PBT
b. AVL but not PBT
c. Neither AVL nor PBT
d. Both AVL and PBT

This is the preorder of a BST (Binary Search Tree): 3, 2, 1, 4. Which of the following statements is true?

a. Both AVL and PBT

b. Neither AVL nor PBT
c. Not AVL but PBT
d. AVL but not PBT

The number of different orders in which we can insert successively the keys 1, 2, 3, 4, 5, 6, 7 in a BST such as to obtain a tree of height 6 is (considering the leaves have height 0):

a. 1
b. 4
c. 2
d. 0

In an OS (Order Statistic) tree which is RB as well, if the dimension of the left subtree is 1, the dimension of the right subtree is in range:

a. [2, 3]
b. [0, 3]
c. [1, 3]
d. [1, 2]

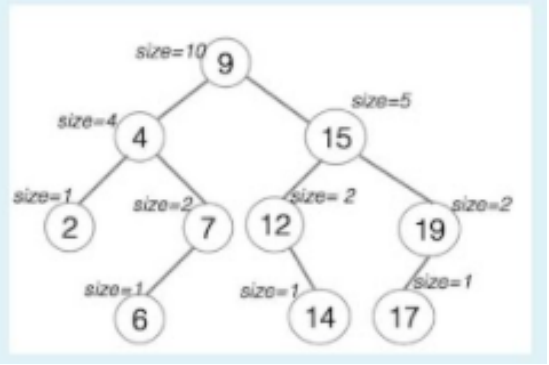
In an OS (Order Statistic) tree which is AVL as well, if the dimension of the left subtree is 2, the dimension of the right subtree is:

a. at least 1
b. at least 3
c. at least 4
d. at least 2

In an OS (Order Statistic) tree which is AVL as well, if the dimension of the left subtree is 3, the dimension of the right subtree is:

a. at most 5
b. at most 4
c. at most 6
d. at most 7

Given the augmented binary search tree below, which can be used to perform searches by rank (using OS_SELECT), the rank of the root is:



a. 1
b. 10
c. 4
d. 5

Assume a graph with 8 vertices and 20 edges. Knowing that Kruskal's algorithm has rejected 7 edges, how many edges were not considered by the algorithm?

- an MST will have V-1 edges, so 7 edges -> 7 were rejected, so 6 weren't considered

a. 5
b. 8
c. 7
d. 6

A spanning tree connects nodes to the network…

a. from a single central node
b. exactly once ??
c. exactly twice
d. multiple ways

Let G be a connected undirected graph having 25 vertices and 100 edges. The weight of a minimum spanning tree of G is 125. When the weight of each edge of G is increased by 10, the weight of a minimum spanning tree becomes…

- 24 edges in the MST -> 125 + 24*10 = 365

a. 225
b. 365
c. 1125
d. 375

Given the graph G = (V, E); V = {a, b, c, d}; E = {(a, b), (b, c), (c, d), (d, a)} to get the largest number of DFS trees, the vertex set should be ordered:

a. Any order would return the same number
b. {a, b, c, d}
c. {d, c, b, a}
d. {c, b, a, d}

Given the graph G = (V, E); V = {a, b, c, d}; E = {(a, b), (c, a), (c, b), (c, d)} to get the smallest number of DFS trees, the vertex set should be ordered:

a. {a, c, b, d} ? - come back to it
b. {a, b, d, c}
c. {a, b, c, d}
d. {c, a, b, d}

In the following graph the dfs which considers the lexicographical order of vertices would classify (c, a) as a back edge. Which order should we consider so that it becomes a tree edge?

G = (V, E), V = {a, b, c}, E = {(a, b), (b, c), (c, a)}

a. any of (c, a, b), (b, c, a) or (c, b, a)
b. c, b, a
c. b, c, a
d. c, a, b

In the following graph the dfs which considers the lexicographical order of vertices would classify (a, b) as a tree edge. Which order should we consider so that it becomes a cross edge?

G = (V, E), V = {a, b, c}, E = {(a, b), (a, c)}

a. b, c, a OR c, a, b
b. b, a, c OR a, c, b
c. b, a, c OR b, c, a
d. c, a, b OR a, c, b

In the following graph, if we add one more edge, that added edge would be either back or tree edge on different dfs. Which edge should we add?

G = (V, E), V = {a, b, c, d}, E = {(a, b), (b, c), (d, a)}

a. (c, d)
b. (d, b)
c. (a, c)
d. any of (c, d), (d, b) or (a, c)

In the following directed graph, the order we should consider for the vertices so that (c, a) is a **back** edge in one dfs and a **tree** edge in the other one is:

G = (V, E), V = {a, b, c}, E = {(a, b), (b, c), (c, a)}

a. a, b, c vs c, a, b
b. b, c, a vs c, a, b
c. a, b, c vs b, c, a
d. either a, b, c vs c, a, b OR a, b, c vs b, c, a

In the following graph, different dfs would classify some edge either as tree or cross edge. Which edge is that one?

G = (V, E), V = {a, b, c}, E = {(b, a), (b, c)}

a. (b, c)
b. neither (b, a) nor (b, c)
c. (b, a)
d. both (b, a) and (b, c)

Consider an undirected graph G having 11 nodes. Its adjacency matrix is given by a 11x11 square matrix whose (i) diagonal elements are 0's and (ii) non-diagonal elements are 5. Then:

a. the graph has a unique MST, of cost 50;
b. the graph has multiple distinct MSTs, of cost 55;
c. the graph has multiple distinct MSTs, each of cost 50;
d. the graph has a unique MST, of cost 55.

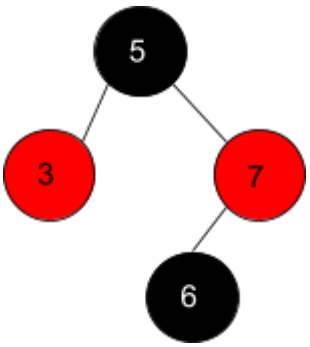
You are given a weighted, undirected graph, whose edges have distinct weights. Considering the minimum edge weight - emin, and the maximum weight edge - emax, which of the following statements might be FALSE?

a. no minimum spanning tree contains emax
b. every minimum spanning tree of G must contain emin
c. if emax is in a minimum spanning tree, then its removal must disconnect G
d. has a unique minimum spanning tree

The maximal subset in a heap representing a total order relation is:

a. any branch
b. the leftmost branch
c. the root with its children
d. the leaves

Which of the following statements is true for the tree below:



a. RB but NOT AVL
b. NOT RB and NOT AVL
c. NOT RB but AVL
d. RB and AVL

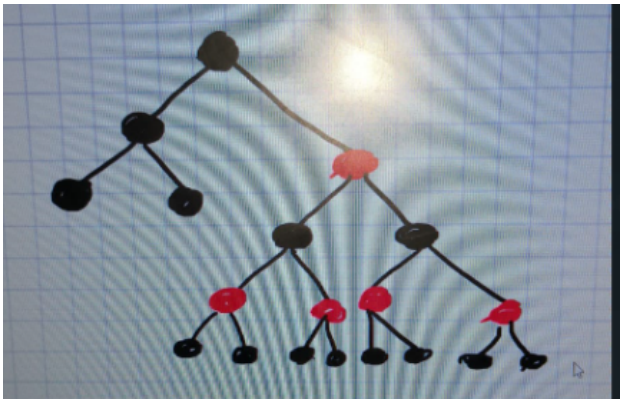
We can build a RB tree which is not an AVL tree with at least:

a. 7 keys
b. CANNOT! Any RB tree is necessary an AVL tree
c. 5 keys
d. 6 keys

This is NOT a RB tree because:

a. 6 is black
b. 10 is black
c. 3 and 5 are black
d. BUT it IS a RB tree

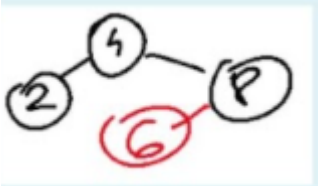
In a RB tree whose left subtree consists of 3 nodes all black, the right subtree has (also counting nil nodes, which are black):



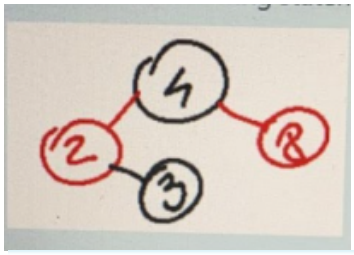
The last row of black nodes are actually the black leaves, which are NIL.

a. at least 11 nodes
b. at least 7 nodes
c. at most 15 nodes
d. at most 7 nodes

Which of the following statements are true for this tree:



a. RB and AVL and PBT
b. no balance property (RB, AVL, PBT)
c. RB and AVL but NOT PBT
d. RB but NOT AVL



a. RB and AVL
b. NOT RB but AVL
c. RB but NOT AVL
d. NOT RB and NOT AVL

To fix the issue of the RB tree below, we need to:

a. Right rotate along edge 9-7
b. turn 7 red
c. turn 3 black
d. left rotate along edge 5-9

Which of the following statements are true for this tree:

a. NOT RB but AVL
b. RB and AVL
c. NOT RB and NOT AVL
d. RB but NOT AVL

This is NOT a RB tree because:

a. It is not PBT
b. It is not BST
c. It is not AVL
d. BUT it IS a RB tree

How much additional memory does heapsort use?

a. O(nlogn)
b. O(logn)
c. O(n)
d. O(1)

If we implemented QuickSort by choosing the pivot to be the first element always, the running time for sorting an already sorted array of size n is (1), and for a random arrays is (2):

a. (1) - $O(n^2)$; (2) - O(nlogn)
b. (1) - O(nlogn); (2) - $O(n^2)$
c. (1) - O(n); (2) - O(nlogn)
d. (1) - O(nlogn); (2) - O(n)

Which of the following represent worst case inputs for quicksort, if the pivot is the element in the middle position, always?

a. a sorted array
b. an array sorted descendingly
c. a random array
d. an array of equal elements

A disadvantage of QuickSelect is that:

a. it doesn't sort the input sequence
b. its running time in the worst case is $O(n^2)$
c. its expected behaviour (in terms of number of operations) on random inputs is poor
d. its running time in the average case is O(n)

Given an array of size n containing equal keys, the running time of HeapSort is…

a. O(n!)
b. $O(n^2)$
c. O(n)
d. O(nlogn)

What is the complexity of the algorithm insertion sort in the worst case?

a. O(logn)
b. O(n)
c. O(nlogn)
d. $O(n^2)$

What data structure is used to implement a heap?

a. queue
b. an array
c. a linked list
d. a binary tree

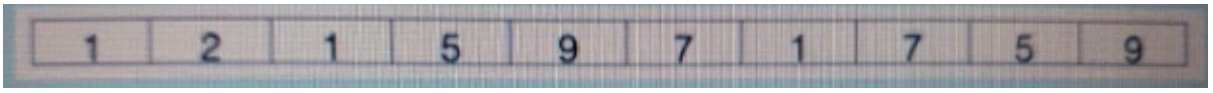
What is the complexity of the algorithm QuickSort in the worst case with regular implementation?

a. O(nlogn)
b. O(logn)
c. $O(n^2)$
d. O(n)

If we use binary search to find the position in the Insertion Sort algorithm, for an array of size n, the algorithm:

a. takes O(nlogn) comparisons and $O(n^2)$ assignments
b. takes $O(n^2)$ comparisons and O(nlogn) assignments
c. takes $O(n^2)$comparisons and $O(n^2)$ assignments
d. takes O(nlogn) comparisons and O(n) assignments

Given the heap, the children of the node with key 2 have keys:



a. 1 and 5
b. 5 and 5
c. 5 and 9
d. 9 and 7

The height of a heap having n nodes is O(...):

a. 2^n
b. logn
c. n
d. n/2

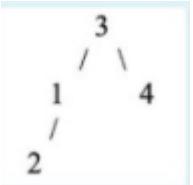
Given the array A = [10, 6, 7, 3, 9, 1, 12], quickSelect(A, 5) returns:

a. 9
b. 6
c. the sorted array
d. 7

Given the array A = [10, 6, 7, 3, 9, 1, 12], quickSelect(A, 4) returns:

a. 3
b. 6
c. The sorted array
d. 7

For the tree below, its postorder is:



a. 2, 1, 3, 4
b. 1, 2, 3, 4
c. 2, 1, 4, 3

d. 2, 3, 1, 4

Hash tables are important because:

a. they have O(n) search time in the worst case
b. They have an expected (average case) O(n) search time
c. They have an expected (average case) O(logn) search time
d. They have an expected (average case) O(1) search time

With open addressing, quadratic probing:

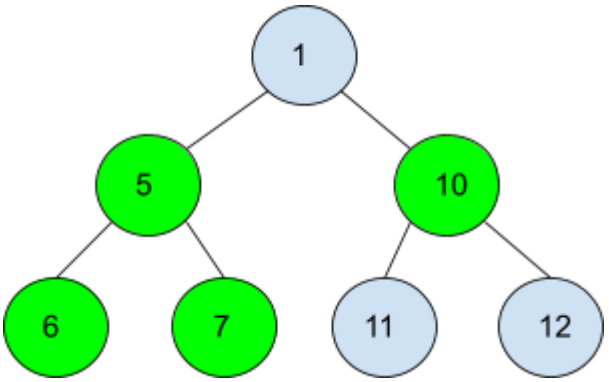
a. as long as the array has empty locations, ANY key could be inserted
b. expected (average case) search time is O(n)
c. the hash function probes a sequence of potential addresses
d. the hash function can generate all possible addresses, for ANY key

A hash function:

a. if used with open addressing, must be surjective
b. takes a fixed length key and transforms it into an arbitrary length code
c. can produce the same values for 2 different keys
d. takes an arbitrary length key and transforms it into an arbitrary length code

You are given a min-heap. How many array cells do you have to access to find the 3rd smallest element (3rd minimum)? (in the worst case)

a. 3
b. $\log_2 n$
c. 4
d. 7



Remarks:

- the root is not verified, because it cannot be the 3rd smallest element (since it is the smallest);
- so we verify the root's children, and then we also verify the children of the smallest child -> we get the 3rd minimum
- green - checked cells

Being given a singly linked list with first and last, which of the following operations is dependent on the list length?

a. insertFirst
b. insertLast
c. deleteFirst
d. deleteLast

In terms of algorithm features we can trade (that is, relax the first in favour of the second):

a. efficiency for stability only if correct
b. correctness for stability only if efficient
c. correctness for stability
d. correctness for efficiency

We want to implement a queue using a circular singly linked list, like in the figure below. To which node should p point such that both enqueue and dequeue are executed in constant time?

a. it cannot be done using a single pointer
b. it should point to the next node after front
c. it should point to the front node
d. it should point to the rear node

Given an array of n keys, the worst case running time for selection sort if only assignments are considered is:

a. O(logn)
b. O(1)
c. $O(n^2)$
d. O(n)

If O(n) additional memory is used, sorting can be solved in O(n) only if:

a. never
b. always
c. cannot be decided in a generic case
d. additional constraints are added on data