What is the major disadvantage of using dynamic data structure?

- a. the overhead to grow it when the structure becomes full
- b. may result in under-estimate of memory usage
- c. need to provide a good estimate of memory usage in advance
- d. may over-estimate memory usage
- e. waste of memory space

What is the average cost of insertion in an ArrayList with N items inside?

- a. O(1)
- b. O(NLogN)
- c. O(N*N)
- d. O(LogN)
- e. O(N)

ArrayList核心扩容

```
private void grow(int minCapacity) {
        // oldCapacity为旧容量，newCapacity为新容量
        int oldCapacity = elementData.length;
        //将oldCapacity 右移一位，其效果相当于oldCapacity /2，
        //我们知道位运算的速度远远快于整除运算，整句运算式的结果就是将新容量更新为
        旧容量的1.5倍，
        int newCapacity = oldCapacity + (oldCapacity >> 1);
        //然后检查新容量是否大于最小需要容量，若还是小于最小需要容量，那么就把最小
        需要容量当作数组的新容量，
        if (newCapacity - minCapacity < 0)
            newCapacity = minCapacity;

        if (newCapacity - MAX_ARRAY_SIZE > 0)
            newCapacity = hugeCapacity(minCapacity);

        elementData = Arrays.copyOf(elementData, newCapacity);
    }
```

Which of the following is not in-place sorting?

- a. selection sort
- b. bubble sort
- c. insertion sort
- d. merge sort
- e. quick sort

这几种种排序方法简要说明:

In the process of converting "3 + k / t" into postfix expression, how many stack push operations need to be performed?

- a. 2
- b. 0
- c. 1
- d. 3
- e. 5

Question 8

Not yet answered | Marked out of 2.50

In the process of evaluating postfix expression "7 b - 8 ^", how many stack pop operations need to be performed?

- a. 3
- b. 4
- c. 2
- d. 1
- e. 0

1.利用栈数据结构进行infix转postfix
2.利用栈数据结构计算postfix

Which of the following is not correct in Java?

- a. Interfaces can extend other interfaces
- b. Classes can extend other classes
- c. Java can have "type parameter"
- d. Classes can implement interfaces
- e. Abstract classes can extend other classes

Clear my choice

Not yet answered

Marked out of 2.50

Flag question

Which of the following data structures does not allow duplicates?

- a. Tree
- b. queue
- c. bag
- d. list
- e. set

Clear my choice

Which of the following is not correct in Java?

- a. An Iterator interface must define a remove() method.
- b. A Comparator interface must define a compareTo() method.
- c. An Iterator interface must define a next() method.
- d. A Scanner() is an iterator
- e. An Iterable class must implement inside it an iterator.

Clear my choice

迭代器接口和比较器接口的正确实现

```java
1 usage
private class ARSetIterator implements Iterator<T> {
    4 usages
    private int wizPos;

    1 usage
    @Contract(pure = true)
    public ARSetIterator() { wizPos = 0; }

    @Contract(pure = true)
    @Override
    public boolean hasNext() { return wizPos < size; }

    @Contract(mutates = "this")
    @Override
    public T next() {
        T returnItem = items[wizPos];
        wizPos += 1;
        return returnItem;
    }
}
```

```java
public class Dog implements Comparable<Dog> {

    4 usages
    private String name;
    5 usages
    private int weight;

    3 usages
    @Contract(pure = true)
    public Dog(String n, int w) {
        name = n;
        weight = w;
    }

    /**
     * @param other another Dog
     * @return negative number if this is smaller than other
     *         0 if this equals other
     *         positive number if this is larger than other
     */
    @Override
    public int compareTo( @NotNull Dog other) {
        return this.weight - other.weight;

    }
```

```java
@Override
public boolean equals(Object that ) {
    if (this == that) return true;
    if (that == null) return false;
    if (!(that instanceof Dog)) return false;
    Dog thatDog = (Dog) that;
    return this.weight==thatDog.weight&& this.name.equals(thatDog.name);


}
```

```java
private class SizeComparator implements Comparator<HASet<T>> {


    @Contract(pure = true)
    @Override
    public int compare( @NotNull HASet<T> h1,  @NotNull HASet<T> h2) {
        return h1.size-h2.size;
    }

}

/** Comparator of set based on size */
public Comparator<HASet<T>> getSizeComparator() {
    return new SizeComparator();
}
```

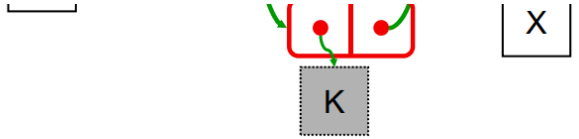
Question 4

Not yet answered | Marked out of 2.50 | Flag question

Data insertion in a linked list is achieved by?

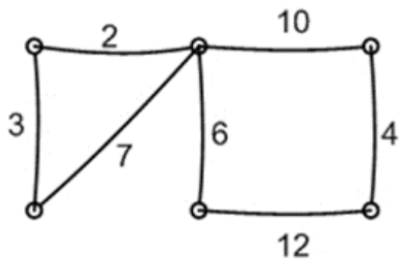
- a. changing of elements
- b. None of the others are true
- c. changing of stacks
- d. changing of links
- e. changing of indices

链表的插入操作



- Traverse the list by following the links
- Insert by changing links
- Remove by changing links

Derive a **maximum** spanning tree (MST) for the following graph. What is the total cost for the MST derived?



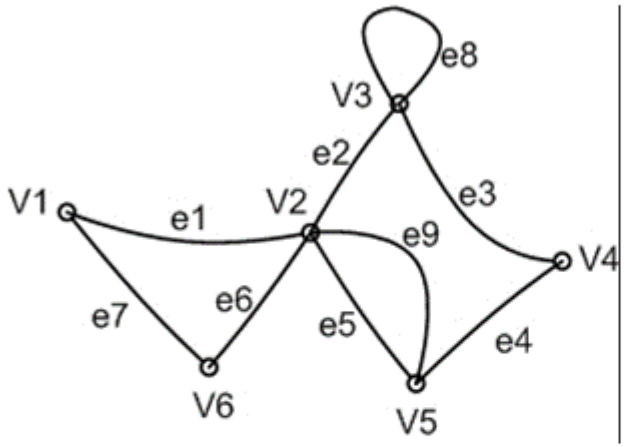
- a. 25
- b. 38
- c. 21
- d. 32
- e. 15

找最小生成树和最大生成树:
1. prim算法
2. Kruskal算法

Which of the following is not a linear data structure?

- a. list
- b. priority queue
- c. queue
- d. tree
- e. stack

What is the degree of vertex V3 in the following graph?



- a. 3
- b. 1
- c. 0
- d. 4
- e. 2

Graph的度
分为有向图和无向图
拓扑排序

Which of the following data structures is not ordered?

- a. map
- b. stack
- c. queue
- d. array
- e. list

If a hash table is well designed with few collisions, what is the average cost of each hash table lookup?

- a. $O(N^3)$
- b. O(1)
- c. None of the others are true
- d. $O(N^2)$
- e. O(N)

Hash Tables

- another kind of Table
- **O(1)** in average for insert, lookup, and remove
- use an array named T of *capacity* N
- define a hash function that returns an integer int **H(string key)**
- must return an integer between 0 and N-1
- store the key and info at T[H(key)]
- H() must always return the same integer for a given key

HashMap的实现

Question 7

Not yet answered | Marked out of 2.50 | Flag question

What is the average performance of searching an item within a non-sorted array with N items inside?

- a. O(N*N)
- b. O(1)
- c. O(LogN)
- d. O(N)
- e. O(NLogN)

Which of the following operations does not belong to the stack interface?

- a. min
- b. pop
- c. push
- d. new
- e. top

What is returned from hasNext() when the queue is empty?

- a. a boolean value FALSE
- b. None of the others are true
- c. the null value
- d. an exception
- e. a boolean value TRUE

```
1 usage
private class ARSetIterator implements Iterator<T> {
    4 usages
    private int wizPos;

    1 usage
    @Contract(pure = true)
    public ARSetIterator() {
        wizPos = 0;
    }

    @Contract(pure = true)
    @Override
    public boolean hasNext() {
        return wizPos < size;
    }

    @Contract(mutates = "this")
    @Override
    public T next() {
        T returnItem = items[wizPos];
        wizPos += 1;
        return returnItem;
    }
}
```

Assume a reasonable and efficient unordered ArraySet implementation, which of the following does not have the same cost as the others?

- a. all the others have the same cost
- b. add
- c. isEmpty
- d. contains
- e. remove

```java
public void add(T item) {
    if (item == null) {
        throw new IllegalArgumentException("IllegalArgumentException");
    }
    if (!contains(item)) {
        items[size] = item;
        size++;
    }


}
```

```java
public boolean contains(T item) {
    for (T i : this) {
        if (i.equals(item)) {
            return true;
        }
    }
    return false;
}
```

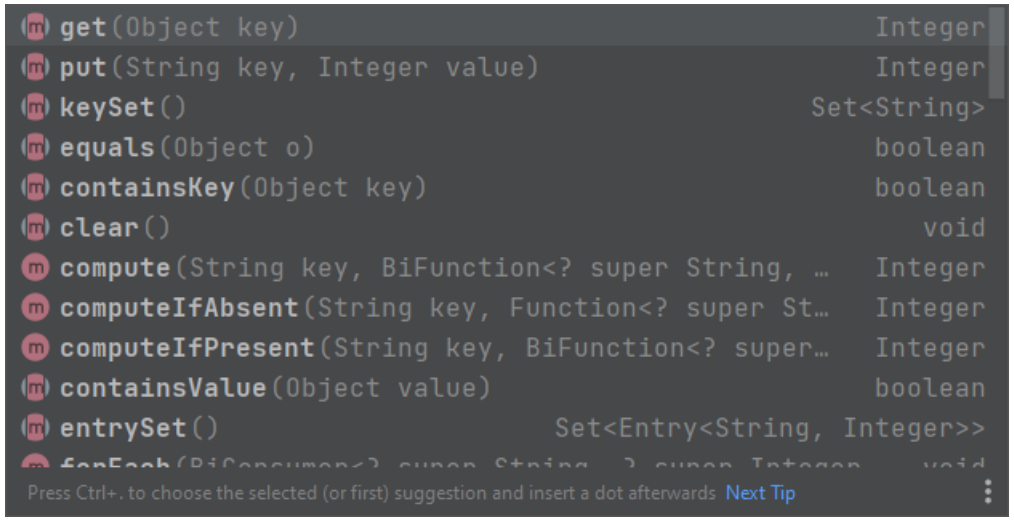
```java
public void remove(T item) {
    if (item == null) {
        throw new IllegalArgumentException("IllegalArgumentException");
    }
    if (contains(item)) {
        int index = 0;
        for (int i = 0; i < size; i++) {
            if (items[i].equals(item)) {
                index = i;
                break;
            }
        }
        for (int i = index; i < size - 1; i++) {
            items[i] = items[i + 1];
        }
        size--;
    }
}
```

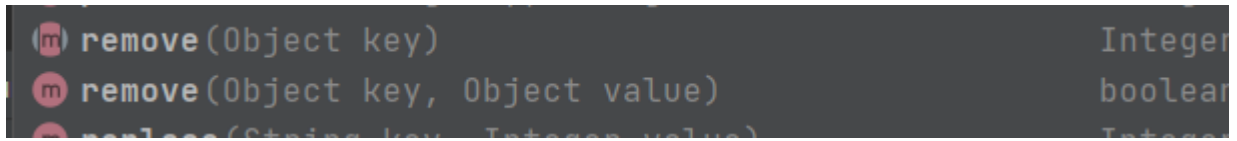
Which of the following specifies the number of vertices in a graph G?

- a. order of G
- b. map of G
- c. hash of G
- d. index of G
- e. incidence of G

Which of the following operations does not exist under a map data structure?

- a. get
- b. size
- c. set
- d. remove
- e. put





Which of the following operations is used for rebalancing an AVL tree?

- a. rotation
- b. none of the others are correct
- c. mirroring
- d. pruning
- e. shifting

AVL Time complexity in big O notation

Algorithm	Average	Worst case
Search	$O(\log\ n)$	$O(\log\ n)$
Insert	$O(\log\ n)$	$O(\log\ n)$
Delete	$O(\log\ n)$	$O(\log\ n)$
Space	$O(n)$	$O(n)$

What is the average performance of searching an item within an AVL tree with M items inside?

- a. O(1)
- b. O(LogM)
- c. O(N*N)
- d. O(MLogM)
- e. O(N)

What is the worst case performance for deletion of one value in an AVL tree with N nodes?

- a. O(log N)
- b. O(1)
- c. O(N log N)
- d. $O(N^2)$
- e. O(N)

11
16
21
26
31

Using abstraction and encapsulation in designing programs, which of the following is not necessarily an outcome?

- a. modular programs
- b. shorter code
- c. program easier to change
- d. localisation of errors
- e. elimination of side effects

Which of the following is not correct?

- a. items in a set may not be duplicated
- b. items in a bag may be duplicated
- c. items in a sorted list may be duplicated
- d. items in a linked list may be duplicated
- e. items in a priority queue may not be ordered

Clear my choice

Which of the following is related to the "information hiding" design principle?

- a. decoding
- b. dynamic data type
- c. privacy
- d. watermarking
- e. encoding

What is the best language to study data structures?

- a. C
- b. Assembly
- c. Java
- d. C++
- e. None of the others are true. Language is not an issue.

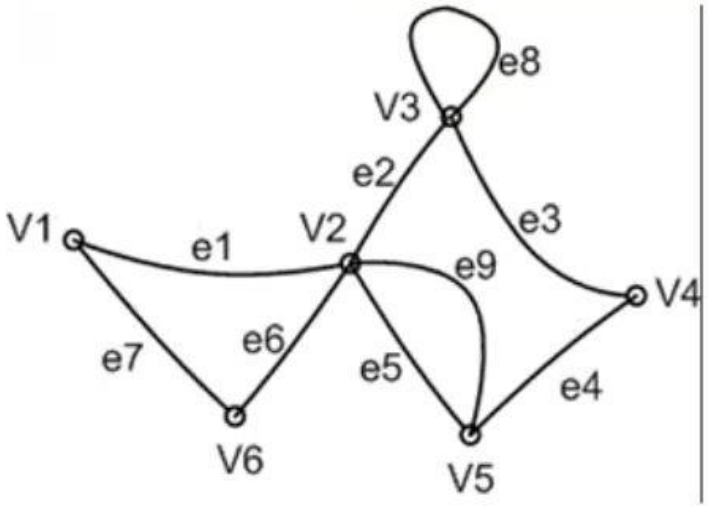
Clear my choice

Which of the following is TRUE for hash table design?

- a. Table size is usually prime to avoid bias
- b. A hash function is needed to generate random numbers
- c. Information hiding is use
- d. None of the others
- e. Huffman coding is applied in the design

- Table size is usually *prime* to avoid bias

What is the degree of vertex V3 in the following graph?

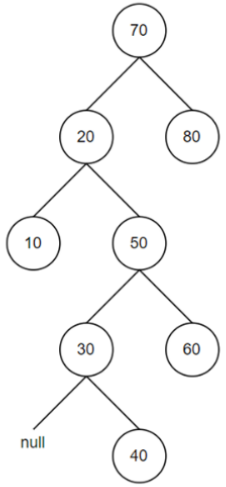


- a. 3
- b. 1
- c. 0
- d. 4
- e. 2

Not yet answered | Marked out of 10.00 | Flag question

A Binary Search Tree (BST) was created by inserting these integers in the following 50, 80, 30, 10, 40, 60 (i.e. "70" gets inserted first and "60" inserted last).

Time left 1:34:41

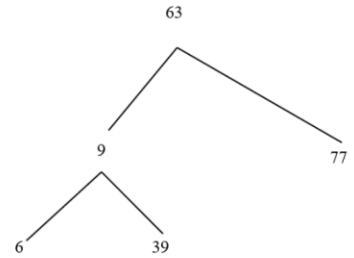


Drag-and-drop the correct sequence of integers when traversing the tree using **Post-order Depth First Traversal**. Note that your sequence must absolutely match the index numbers to the left-most column of the table otherwise 2 marks will be deducted for each incorrect match. The answers for the first 3 indices have been provided. Complete the rest.

	Correct Integer Sequence
Index 0	10
Index 1	40
Index 2	30
Index 3	
Index 4	
Index 5	
Index 6	
Index 7	

20 70 60 40 80 50 10 30

What is the outcome of postOrder traversal of the following tree?



- a. 63, 9, 6, 39, 77
- b. 6, 9, 39, 63, 77
- c. 77, 63, 39, 9, 6
- d. 6, 39, 9, 77, 63
- e. 77, 6, 39, 9, 63

Drag-and-drop the correct sequence in implementing the *delete* operation of a *binary min-heap* abstract data type, assuming the element to be deleted is never in the last level. Note that your sequence must absolutely match the step numbers to the left-most column of the table below otherwise 3 marks will be deducted for each incorrect match.

Step 1	Drag answer here
Step 2	Drag answer here
Step 3	Drag answer here
Step 4	Drag answer here
Step 5	Drag answer here

- Replaced the root element in the heap with the found indexed element.
- Add the indexed element to the bottom leaf of the heap.
- Output updated binary heap.
- Take out the last element from the last level of the heap and replace the index with this element.
- If the replaced element is greater than any of its child node, swap the element with its smallest child.
- If the replaced element is smaller than any of its

Drag-and-drop the correct sequence in implementing the *delete* operation of a *binary min-heap* abstract data type, assuming the element to be deleted is never in the l
level. Note that your sequence must absolutely match the step numbers to the left-most column of the table below otherwise 3 marks will be deducted for each incor
match.

Step 1	Drag answer here
Step 2	Drag answer here
Step 3	Drag answer here
Step 4	Drag answer here
Step 5	Drag answer here

- Take out the last element from the last level of the heap and replace the index with this element.
- If the replaced element is greater than any of its child node, swap the element with its smallest child.
- If the replaced element is smaller than any of its child node, swap the element with its greatest child.
- Repeat steps 3 to 4 until the node reaches its correct position.
- Find the index for the element to be deleted.

```java
public static <T> void printArray( @NotNull T[] array) {
    for (T element : array) {
        System.out.println(element);
    }
}

public static void main(String[] args) {
    Integer[] intArray = {1, 2, 3, 4, 5};
    String[] stringArray = {"a", "b"};


    printArray(intArray);


    printArray(stringArray);
}
```