CSE12 - Lecture 26 - Answer Key

Wednesday, November 30, 2022 11:00 AM

Generics

Which of the following AList declarations will result in a compile error? Check all that apply:

A. AList< int > myList= new AList< int >();
B. List< Integer > myList = new AList< Integer >();
C. AList< AList< String >> myList = new AList< AList< String >>();
D. AList myList< Integer > = new AList< Integer >();
E. AList< E > myList = new AList< String >();
F. AList< Object> myList = new AList< Object >();

A, D, E -> cause compiler errors

Queue / Stack

```
ALQueue<String> myQ = new ALQueue<>();
myQ.enqueue("A");
myQ.enqueue("A");
myQ.dequeue();
myQ.enqueue("C");
myQ.enqueue("B");
myQ.enqueue(myQ.dequeue());
myQ.enqueue("D");
myQ.enqueue(myQ.dequeue());
System.out.println(myQ.toString());
```

What is printed?

front -> [B, A, D, C] <- back

```
ALStack<String> myS = new ALStack<>();
myS.push("A");
myS.push("A");
myS.pop();
myS.push("C");
myS.push("B");
myS.push(myS.pop());
myS.push("D");
myS.push(myS.pop());
System.out.println(myS.toString());
```

What is printed?

[A, C, B, D] <- top

Run-Time

```
// This method returns whether or not a pair of numbers, num1 and num2, are between 1-m and 1-n, respectively
boolean findPair(int num1, int num2, int m, int n) {
   for (int i = 1; i <= m; i++) {
      if (num1 == i) {
         for (int j = 1; j <= n; j++) {
            if (num2 == j) {
               return true;
            }
         }
      }
   }
   return false;
}
```

What is the worst case runtime of findPair? O(m + n)

What is the best case runtime of findPair given it returns false? O(m)

Time Complexity Review

Check which of the following are true:

A. $n + 5n^3 + 8n^4 = O(n)$
B. $n! + n^2 = O(n\log(n))$
C. $2^n + n\log(n) = O(n!)$
D. $1/(n^2) + 5 = O(1/n)$

C -> true

Which of the following relationships hold? [Extra practice: come up with values for n0 and C for those that do]

A. $n^2 + n^3$ is $\Omega(n^3)$
B. $n * \log(n) + n^2$ is $\Omega(\log(n) * n^2)$
C. $1/n + \log(n) * n^2$ is $O(n^2)$
D. $n + \log(n)$ is $O(\log(n))$
E. $1/(n^{10}) + 100$ is $\Theta(1)$
F. $(n^4)/\log(n)$ is $\Theta(n^4)$

A, E -> hold

Refer to the following methods:

```
public static void f1(int n) {
    int a = 0;
    for (int i = 0; i < n; i++) {
        for (int j = i; j < n; j++) {
            a = i;
        }
    }
}

public static void f2(int n) {
    for (int i = 0; i < n; i += 1) {
        n = n / 2;
    }
}

public static void f3(int n) {
    int a = 0;
    int x = Math.abs(100 - n) * n;
    for (int i = 0; i < x; i++) {
        a = i;
    }
}
```

Which of the following big-theta statements are true:

A. f1 is $\Theta(1)$
B. f1 is $\Theta(n)$
C. f1 is $\Theta(n^2)$
D. f2 is $\Theta(1)$
E. f2 is $\Theta(\log(n))$
F. f2 is $\Theta(n)$
G. f3 is $\Theta(1)$
H. f3 is $\Theta(n)$
I. f3 is $\Theta(n^2)$

C, E, I -> true

Partition

Consider the following code and the implementation of partition() discussed in lecture.

```
String[] b = {"b", "f", "a", "e", "c", "d" };
System.out.println(partition(b, 0, 6));
System.out.println(Arrays.deepToString(b));
```

What return value would partition() method print out for the above array, low and high? 3

What would the array look like after the above call to partition()?

b, c, a, d, e, f

MergeSort

Consider the merge sort from class. How many times will the element at index 0 be copied when sorting an array of length n over the entire run of the algorithm?

$2 * \log_2(n)$

Which of the following statements about sorting are true?

A. The best case time of all sorts is O(1) because of the case when an array is length 1
B. Merge sort has best and worst cases of O(nlg(n))
C. If arrays are split into thirds instead of halves in merge sort, the best case would still be O(nlg(n)) {HINT: look up the rules of logs!}
D. Quicksort is O(n^2) only when an array is in reversed order
E. The worst cases for selection sort and insertion sort occur when an array is in reversed order

b, c, e -> true

Hash Table (using separate chaining)

```
int hash(String key) {
   return key.length;
}
```

Hash table just before expandCapacity is called:

0. - null
1. - {"greetings" : 6}
2. - {"hi" : 5}
3. - {"bye" : 9}
4. - {"happy week 7" : 3}
5. - {"hello" : 2}
6. - null
7. - null

After expandCapacity is called, which of the following elements will have a different index in the new array after rehashing?

A. {"greetings" : 6}
B. {"hi" : 5}
C. {"bye" : 9}
D. {"happy week 7" : 3}
E. {"hello" : 2}

A, D -> will have different index

Hash Table - Separate Chaining

```
int hash(char key) {
   return (int) key;
}
```

Which of the following sequences of insertions would cause the most collisions for a hash table with four buckets and assuming expandCapacity is not called during the adds?

A. add('A', 56); add('B', 5); add('C', 65); add('D', 2);

B. add('E', 43); add('F', 7); add('K', 6); add('L', 160);

C. add('M', 58); add('Q', 14); add('U', 20); add('W', 37);

D. add('N', 7); add('R', 24); add('V', 92); add('Z', 100);

E. add('Z', 91); add('R', 604); add('P', 9); add('L', 5);

D

65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O
80	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	W
88	X
89	Y
90	Z

Hash Table - Linear Probing

```
int hash(char key) {
   return (int) key;
}
```

Also refer to the following sequence of insertions:

```
add('N', 7);
add('R', 24);
add('V', 92);
add('Z', 100);
```

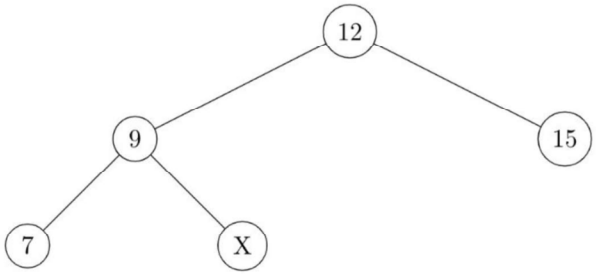
What is the contents of the bucket array right before calling expandCapacity()?

[{V: 92}, null, {N: 7}, {R: 24}]

What is the contents of the bucket array after the sequence has ended?

[null, null, {R: 24}, {Z: 100}, null, null, {V: 92}, {N: 7}]

BST

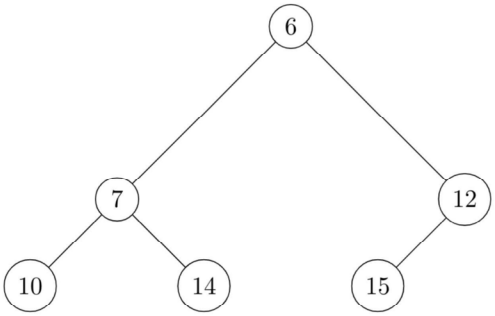


If X is the fifth value added to the BST below, which of the following are possible values of X? Select all that apply.

A. 6
B. 8
C. 10
D. 11
E. 13

C, D

Min Heap Add



If the value 4 is added to the min heap below, what number will end up in the new bottom right leaf node?

12

Iterator

Which interfaces are required by Java to use a data structure in an enhanced for loop?

Iterable, Iterator

Which is the proper way to implement next() for an Iterator:

A. return value
B. save value, update to next element, return saved value
C. update to next element, return value
D. save value, return saved value
E. return value, update to next element

B