Stacks and Queues (Part 2)

CS112 Recitation 3

Note: No more tilde notation, we are moving back to big O notation :)

Q1 Valid Parentheses

Write a stack client Parentheses that reads in a text stream from standard input and uses a stack to determine whether its parentheses are properly balanced. For example, your program should print true for [()]{}{[()()]()} and false for [(]).

Let's think about it first…

-how can you differentiate between the kinds of parenthesis?

-when should you add an item to the stack?

-when do you know the string is invalid?

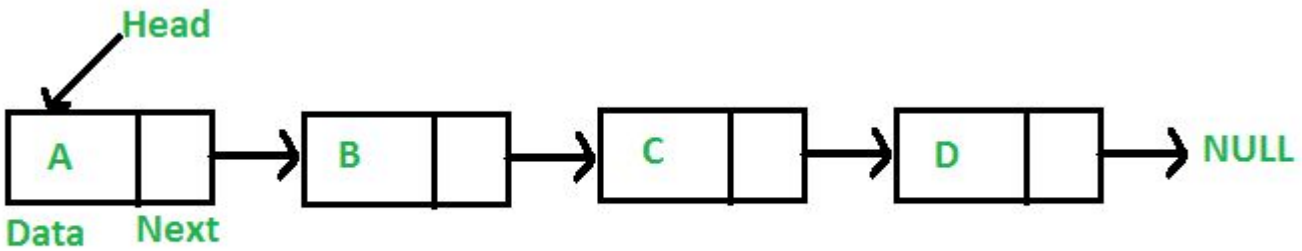
```
public boolean isBalanced(String input) {
        char[] parentheses = input.toCharArray();
        Stack<Character> stack = new Stack<>();

        for (_____) { // loop through all the chars
                if (_____) {
                        stack.push(_____) // when do you push
                } else {
                        if (_____) {
                                return false; // when do we automatically know that our string is not balanced?
                        }

                        char firstItem = stack.pop();
                        if (_____ == ')' && firstItem != _____ // code here
                                ||
                                || ) {
                                        return false;
                        }
                }

                return true;
        }
```

Q1 - Valid Parentheses - Solution

```
private boolean isBalanced(String input) {

        char[] parentheses = input.toCharArray();
        Stack<Character> stack = new Stack<>();

        for (char parenthesis : parentheses) {
                if (parenthesis == '('
                                || parenthesis == '['
                                || parenthesis == '{') {
                        stack.push(parenthesis);
                } else {
                        if (stack.isEmpty()) {
                                return false;
                        }

                        char firstItem = stack.pop();

                        if (parenthesis == ')' && firstItem != '('
                                        || parenthesis == ']' && firstItem != '['
                                        || parenthesis == '}' && firstItem != '{') {
                                return false;
                        }
                }
        }

        return true;
}
```

Q2 Max in a Linked List

Write a method max() that takes a reference to the first node in a linked list as argument and returns the value of the maximum key in the list. Assume that all keys are positive integers, and return 0 if the list is empty. Try to write a recursive solution.



Assume: private Node first refers to the head of the linked list

```
public int max() {
        if (_____) // what is the first thing you should check?
                return _____;

        int maxValue = first.item;

        Node current;

        for (_____) { // fill conditions here
                int currentValue = _____ // get the value of the node we are at the in the list

                if (_____) { // compare to see if we get a new max value


                }
        }

        return maxValue;
}
```

Q2 - Max in a Linked List - Iterative Solution

```
private class Node {
        Item item;
        Node next;
}
private int size;
private Node first;
```

```
public int max() {
        if (isEmpty()) {
                return 0;
        }

        int maxValue = (Integer) first.item;

        Node current;

        for (current = first.next; current != null; current = current.next) {
                int currentValue = (Integer) current.item;

                if (currentValue > maxValue) {
                        maxValue = currentValue;
                }
        }

        return maxValue;
}
```

Q2 - Max in a Linked List - Recursive Solution

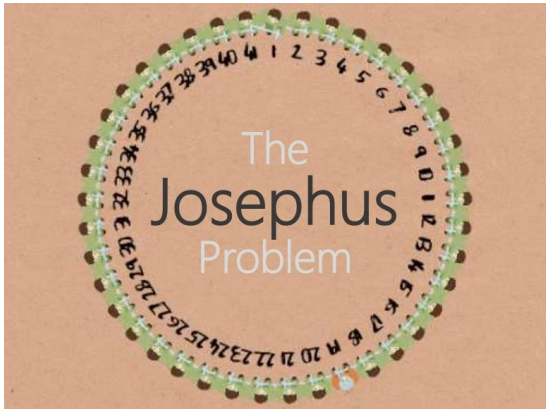
```
private class Node {
        Item item;
        Node next;
}
private int size;
private Node first;
```

```
public int max() {
        if (isEmpty()) {
                return 0;
        }

        int currentMaxValue = (Integer) first.item;
        return getMax(first.next, currentMaxValue);
}

private int getMax(Node node, int currentMaxValue) {
        if (node == null) {
                return currentMaxValue;
        }

        int currentValue = (Integer) node.item;

        if (currentValue > currentMaxValue) {
                currentMaxValue = currentValue;
        }

        return getMax(node.next, currentMaxValue);
}
```

Q3 - Josephus problem

Watch the first 2-3 mins of this video to get a feel of it: https://www.youtube.com/watch?v=uCsD3ZGzMgE

In the Josephus problem from antiquity, N people are in dire straits and agree to the following strategy to reduce the population. They arrange themselves in a circle (at positions numbered from 0 to N−1) and proceed around the circle, eliminating every Mth person until only one person is left. Legend has it that Josephus figured out where to sit to avoid being eliminated. Write a Queue client Josephus that takes N and M from the command line and prints out the order in which people are eliminated (and thus would show Josephus where to sit in the circle).



Q3 - Josephus problem Solution

```java
package chapter1.section3;

import edu.princeton.cs.algs4.Queue;
import edu.princeton.cs.algs4.StdOut;

/**
 * Created by Rene Argento on 8/21/16.
 */
public class Exercise37_JosephusProblem {

    private static void josephusProblem(int personOrder, int numberOfPeople) {

        Queue<Integer> queue = new Queue<>();

        for(int i = 0; i < numberOfPeople; i++) {
            queue.enqueue(i);
        }

        while (numberOfPeople > 0) {

            for (int i = 1; i < personOrder; i++) {
                queue.enqueue(queue.dequeue());
            }

            StdOut.print(queue.dequeue() + " ");

            numberOfPeople--;
        }
    }

    // Parametes example: 5 10
    public static void main(String[] args) {
        int personOrder = Integer.parseInt(args[0]);
        int numberOfPeople = Integer.parseInt(args[1]);

        StdOut.println("Order in which people are eliminated:");
        josephusProblem(personOrder, numberOfPeople);
    }

}
```

Note: don't worry if there are errors saying Std.Out or Queue not resolved, just compile and run it as long as you have all the Std files in the same folder.

Q4 - Task Scheduler (Optional) https://leetcode.com/problems/task-scheduler/

Given a characters array tasks, representing the tasks a CPU needs to do, where each letter represents a different task. Tasks could be done in any order. Each task is done in one unit of time. For each unit of time, the CPU could complete either one task or be idle. However, there is a non-negative integer n that represents the cooldown period between two same tasks (the same letter in the array). That is, there must be at least n units of time between any two same tasks. Return the least number of units of times that the CPU will take to finish all the given tasks.

No need to code, let's just think about it first!

Example 1:

Input: tasks = ["A","A","A","B","B","B"], n = 2

Output: 8

Explanation:

A -> B -> idle -> A -> B -> idle -> A -> B

There is at least 2 units of time between any two same tasks.

Method Definition: public int leastInterval(char[] tasks, int n)

Task Scheduler

What are some things we can do…

1. Execute tasks in the order they come in (making the machine idle when we need to)
    a. What are the issues with this approach?
2. Look at the relative frequencies of each task, and execute the one with the most frequency first
    a. Any issues? How would we go about implementing this?
3. Remember, we only need to get the total time units, we don't really need to arrange the tasks ourselves!

Task Scheduler Intuition

https://leetcode.com/problems/task-scheduler/discuss/760131/Java-Concise-Solution-Intuition-Explained-in-Detail

How to find the idle time ?

Assume, there 3 A tasks. Tasks = `["A","A","A","B"]` . cooling time n = 2 .
If A Is run at a particular time, we cannot run it for 2 intervals after that.
So we have to try filling up those 2 spaces with some other task.

```
What are the maximum number of idle spaces that we could have?
```

It would be (max Frequency task - 1 * n)
In this example, A has maximum frequency of 3, so there are 2 set of intervals(each of size n=2) that needs to be filled with some other task.

A			A			A

Once we know the maximum number of idle spaces, we have to just reduce the the count of spaces every time we find a task that can fill up that idle space.

Steps -
1) Create count array to keep track of frequency of each task. Size 26 as given in input.
2) Sort the frequency is ascending order, the task at last position (`arr[25]`) would be the one with maximum frequency.
Example -
Tasks `["A","A","A","B","B","C"]`
A has max occurrence = 3, n = 2. Hence we can place A as follows

A	idle	idle	A	idle	idle	A

`Max idle spaces = (3-1) * 2 = 4` . We could see there 4 idle spaces above.

3)Now we have to just find that there are sufficient tasks to fill these 4 idle spaces.
We iterate over rest of array in descending order and substract it's count from idle spaces.
Task B has count 2 and C has count 1. Hence 1 space remains idle.

A	B	C	A	B	Idle	A

Answer , `task length + idle time = 6 + 1 = 7`

Task Scheduler Code

```
class Solution {
    public int leastInterval(char[] tasks, int n) {
        if (tasks == null || tasks.length == 0) {
            return 0;
        }
        int m = tasks.length;
        int[] cnt = new int[26];
        /* Build the count array with frequency of each task */
        for (char c: tasks) {
            cnt[c - 'A']++;
        }
        Arrays.sort(cnt);
        /* Get maximum frequency task  and calculate max idle spaces*/
        int max = cnt[25]-1;
        int spaces = max * n;

        /* Iterate over rest of the array and reduce the idle space count */
        for (int i = 24; i >= 0; i--) {
            spaces -= Math.min(max, cnt[i]);
        }
        /* Handle cases when spaces become negative */
        spaces = Math.max(0, spaces);
        return tasks.length + spaces;
    }
}
```

Good Work!

Go to https://dynrec.cs.rutgers.edu/live/

Enter the Quiz Code: