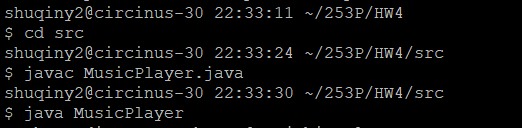
Compile

javac MusicPlayler.java

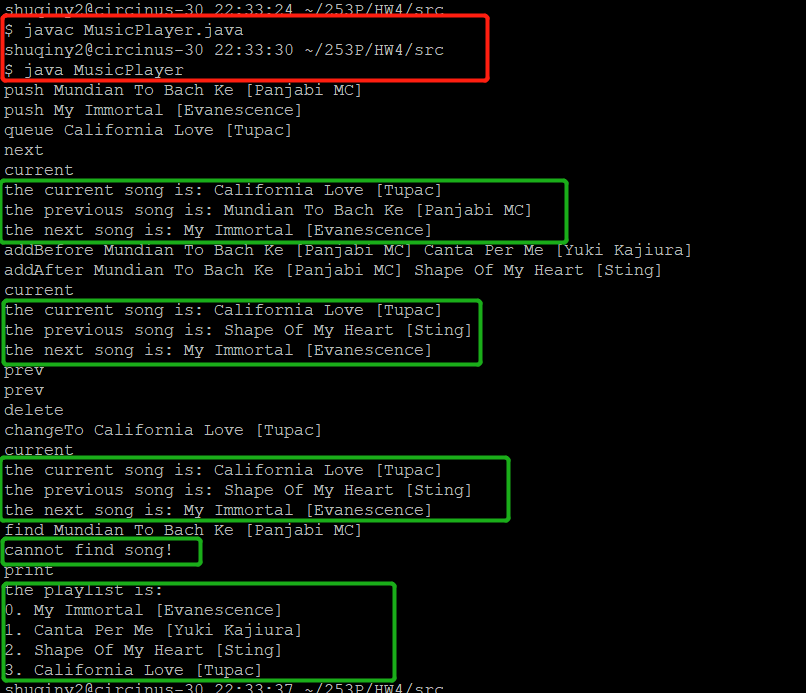
java MusicPlayer



Run

// red part: compile and run

// green part: output



Q & A

1. What data structure did you implement ***SimplePlayList*** as?

I implement SimplyPlayList as a linked list, each song can be seen as a node.

1. List all the attributes (aka fields) you needed in order to implement ***SimplePlayList*** (also include the attributes for any other auxiliary data structures it uses)?  (Do not list functions (aka methods)).

**Class** SimplePlayList {

**protected** Node **head**; *// The first song of playlist***protected** Node **cur**; *// The current song of playlist***protected int size**; *// The size of playlist*

}

**class** Node {  
 **protected** Song **song**;  
 **protected** Node **next**;

}

**class** Song {  
 **protected** String **title**;  
 **protected** String **artist**;

}

1. How does ***SimplePlayList*** retrieve a random song in *O(n)* time?  Explain in detail using a few sentences.

Get a random n ( n is between 0 and the length of playlist ), move the pointer from current position to its next n times.

1. If ***prev*** is processed in *O(n)* time, then how is ***find*** able to print the previous song of the found song in *O(1)* time?

The cur pointer can only move to the next node, so if you want to visit the previous node, you have to start from the beginning. That’s how ***prev*** works in O(n). But in ***find***, since we have to traverse the whole list***,*** we can keep the location of the previous song with an extra pointer. By using this pointer we can achieve O(1) in ***find.***

Two Leetcode problems are substitutions, but I’d like to briefly talk about some ideas about 4.2 and 4.3

1. Very briefly explain how ***GeneralPlayList*** differs from ***SimplePlayList*** as? (Just one or two short sentences that gets the main point(s) across).

***SimplePlayList*** is a linked list while ***GeneralPlayList*** is a doubleLinkedList. Also, ***GeneralPlayList*** keeps a pointer pointing to its tail.

1. List all the attributes (aka fields) you needed in order to implement ***GeneralPlayList*** (also include the attributes for any other auxiliary data structures it uses)? (Do not list functions (aka methods)).

**class** DoubleLinkedNode {  
 protected Song song;  
 protected DoubleLinkedNode next;  
 protected DoubleLinkedNode pre;

protect boolean visited; // if this node has been visited

}

**class** GeneralPlayList **implements** MusicPlayerImpl {  
  
 protected DoubleLinkedNode head; // The first song of playlist  
 protected DoubleLinkedNode tail; // The first song of playlist  
 protected DoubleLinkedNode cur; // The current song of playlist

protected int size; *// The size of playlist*  
 static int randomCnt;

}

1. What allows queue to be able to add to the end of the playlist in O(1) time now?

We keep a tail pointer. Every time queue an element, just append to the tail and move the pointer.

1. How does ***GeneralPlayList*** retrieve a random song without iterating over a song twice? Explain in detail using a few sentences.

Every node has a param visited, which will be initialized to false.

Add a param randomCnt to GeneralPlaylist, counting the number of songs that was randomly retrieved.

When random() is called, we firstly do the same traversal in SimplePlaylist, then check if this node has been visited. If hasn’t been visited, update visited, increment randomCnt and retrieve this song; else go next until an unvisited node is found.

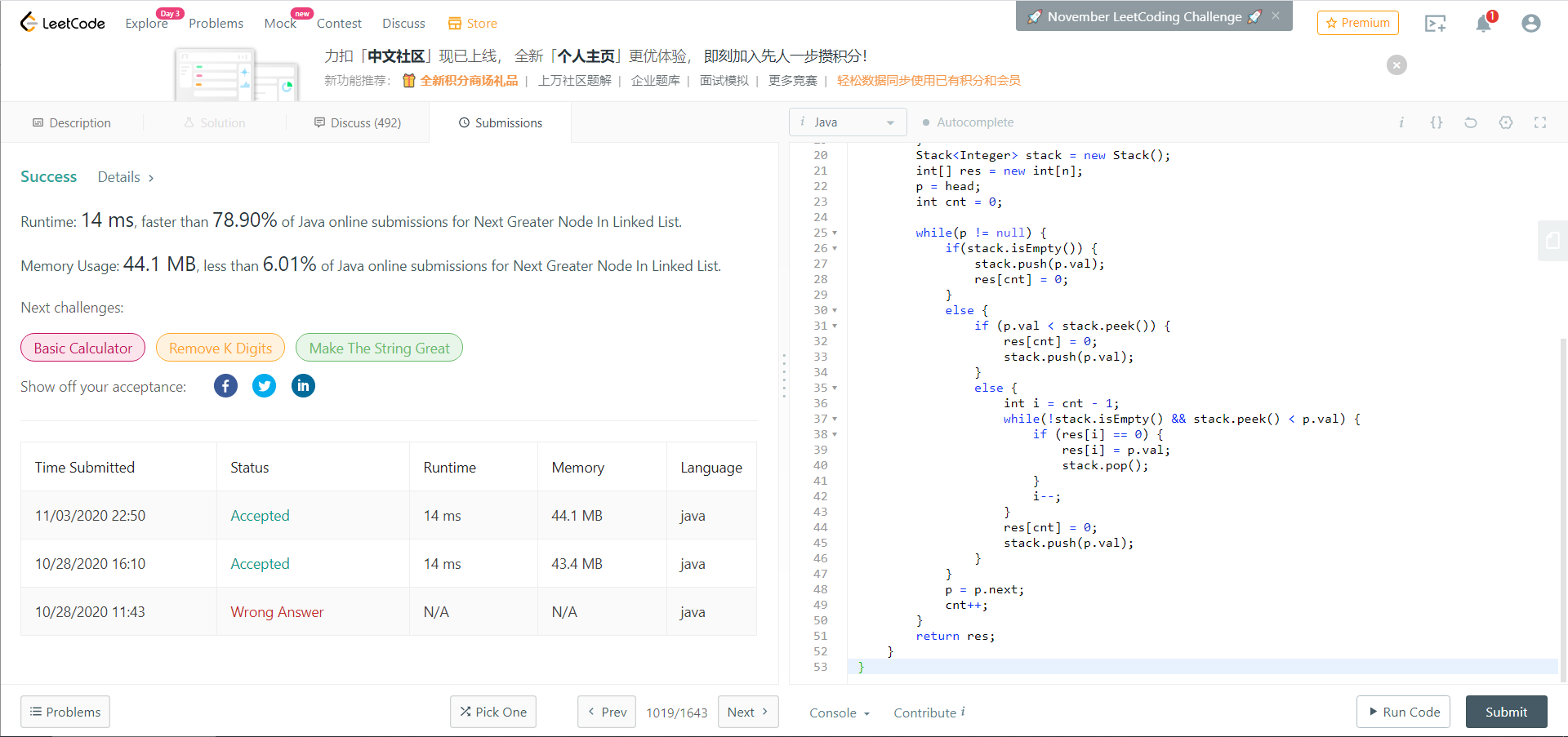
When randomCnt equals to the size of the playlist, reset randomCnt to 0 and every node to unvisited.

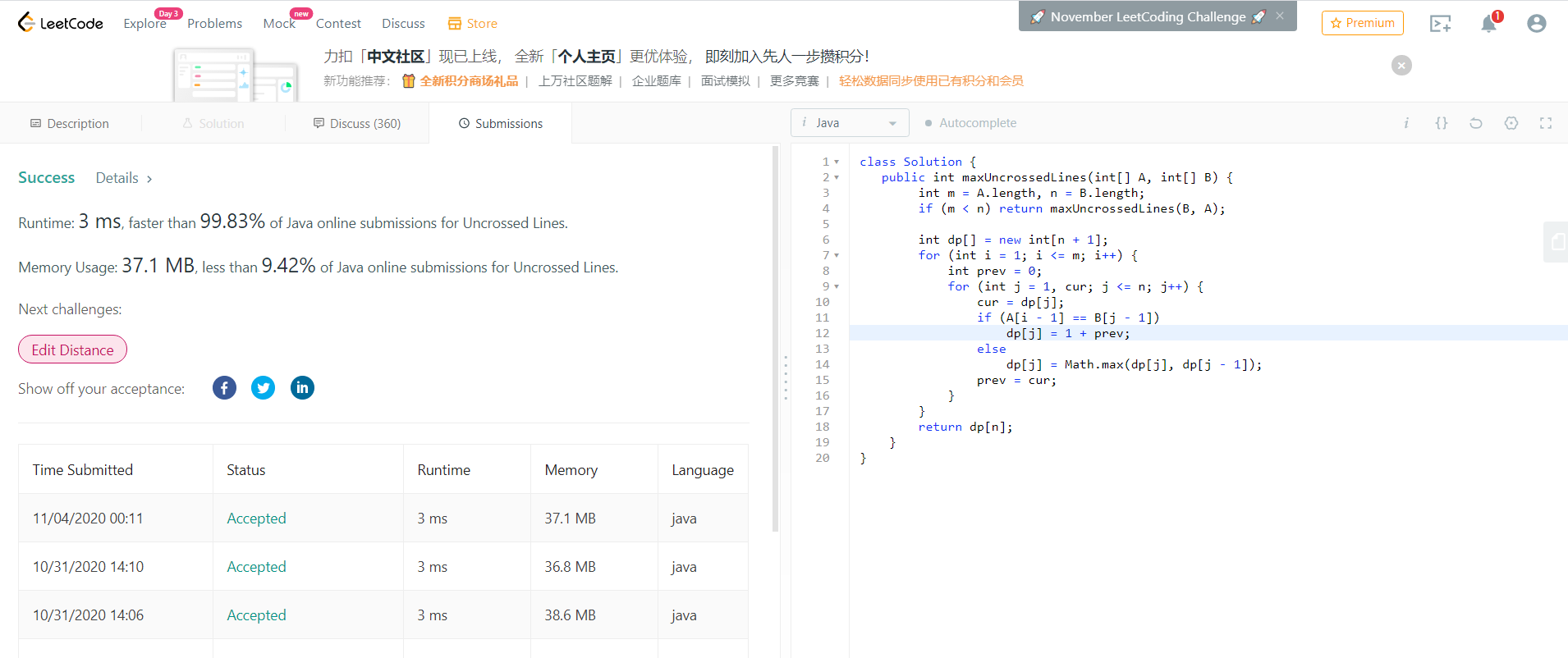
1. Very briefly explain how ***AdvancedPlayList*** differs from ***GeneralPlayList*** as? (Just one or two short sentences that gets the main point(s) across).

***AdvancedPlayList*** has one more HashMap<String, DoubleLinkedNode>, whose key is the title and artist of a song and the value is the node in playlist.

1. What additional data structure did you use to help you achieve the desired improvements?

A hashMap.

Leetcode 1019

Leetcode 1035