Revised 1.2.3 Part 2 Array Algorithms

Otakat Andrysek

Part VII: Common Array Algorithms – Count Matches

The goal is to have everything completed and turned in by the time we meet on Friday. If you don't get to do the insert algorithm in Part VIII, don't worry about it...turn in what you have. Be sure to have 1.2.2B done by Friday.

I will look at 3, 7 and 9. If you have a lot of time on your hands at the end after #9 sort the array by ranking and/or alphabetically by song title.

Another common array algorithm is to count the number of items in an array that match a certain value. Explore this type of algorithm in the steps below.

1. In your initialization list, change your ratings so that you have a few ties for #2.

   Done.

2. In a new for-each loop:
   a. Count the number of songs that are tied for second.

   b. Using a String variable and string concatenation, keep track of the song titles tied for second.

3. a) When the loop finishes, report the number of #2 songs and their titles. Why was a for-each loop recommended for this step (as opposed to a standard for loop)?

   A for each loop would be more efficient than my code.

   b) Why was it better to store the names of the song matches in a string than in a song array?

   Because arrays cannot be modified.

   c) What would be a problem using a method to do this algorithm?

   I don't know… I used a method.

d) Paste a snip of your code which carries out the task described in Part VII.

No, I'm not proud of how I did this.

```
public static int tiedForSecond(Song[] songs, int bestSongValue)
{
    System.out.println("\n");
    int k = 0;
    int m = 1;
    for (int j = 1; j < 9; j++)
    {
        for (int i = 0; i < songs.length; i++)
        {
            if (songs[i].getRating() == (bestSongValue - j))
            {
                if (j > m)
                {
                    return 0;
                }
                m = j;
                System.out.println("The second place result(s) | Name: " + songs[i].getTitle() + " | Rating " + songs[i].getRating());
            }
        }
    }
    return -1;
}
```

Part VIII: Common Array Algorithms – Delete and Insert

Consider how you would go about deleting and inserting elements in an array. First, remember that when you create an array, it has a *fixed size*; you cannot change it. The values in an array can be null, or the empty string, or zero, but the array elements still exist. This means, an array can be **partially filled**.
For example, you have created your songs array of 10 songs, but later decide to delete the sixth song. Your array might end up looking like:

index	contents
0	`Song("The Twist")`
1	`Song("Smooth")`
2	`Song("Mack the Knife")`
3	`Song("How Do I Live")`
4	`Song("Party Rock Anthem")`
5	`null`
6	`Song("Macarena")`
7	`Song("Physical")`
8	`Song("You Light Up My Life")`
9	`Song("Hey Jude")`

This is considered partially filled; it seems as if the array has a hole or a gap at index 5. If you were to iterate over this array, the null entry could cause a problem. So, to delete elements in an array, you must rearrange them so there are no gaps or holes.

4. Review the presentation on deleting an array element, *1.2.3 Array Deletion Algorithm*.

5. Continuing in the MediaLib Algorithms class, create a for loop to iterate over the length of your songs array. Choose a song title that you will delete from your array, preferably from somewhere in the middle, and use the equals method to find that song title.

6. Copy the next song in the array to the current location. Continue iterating, replacing the next song into the current location until you reach the end of the array.

7. Write a new for loop to display the contents of your new, shorter song list. Remember, you will need to reference the variable that has the index of the last

defined element in your partially filled array. Paste snips of a table similar to what is shown above, both before and after you made the deletion. You don't need any borders on the table.

index	contents (before)	contents (after)
0	`Song("The Twist")`	`Song("The Twist")`
1	`Song("Smooth")`	`Song("Smooth")`
2	`Song("Mack the Knife")`	`Song("Mack the Knife")`
3	`Song("How Do I Live")`	`Song("How Do I Live")`
4	`Song("Party Rock Anthem")`	`Song("Party Rock Anthem")`
5	`Song("I Gotta Feeling”)`	`Song("Macarena")`
6	`Song("Macarena")`	`Song("Physical")`
7	`Song("Physical")`	`Song("You Light Up My Life")`
8	`Song("You Light Up My Life")`	`Song("Hey Jude")`
9	`Song("Hey Jude")`	`null`

Inserting into a partially filled array reverses the deletion process.

8. Review the presentation on inserting an array element, *1.2.3 Array Insertion Algorithm*.

9. Write an algorithm to insert a new song (a title you haven't used before) into a partially filled array of songs.

Paste snips of a table similar to what is shown above, both before and after you made the insertion. You don't need any borders on the table.

index	contents (before)	contents (after)
0	`Song("The Twist")`	`Song("The Twist")`
1	`Song("Smooth")`	`Song("Smooth")`
2	`Song("Mack the Knife")`	`Song("Mack the Knife")`
3	`Song("How Do I Live")`	`Song("How Do I Live")`
4	`Song("Party Rock Anthem")`	`Song("Party Rock Anthem")`
5	`Song("I Gotta Feeling”)`	`Song("I Gotta Feeling”)`
6	`Song("Macarena")`	`Song("Macarena")`
7	`Song("Physical")`	`Song("Physical")`
8	`Song("You Light Up My Life")`	`Song("You Light Up My Life")`
9		`Song(“It Works!”)`