Matthew Ostin
CSCI 1112
Professor Taylor

Homework three analysis

(Linked List) public void add(Song song);

1) Analysis variable *n* (number of songs in the list)
2) Pseudo-code:
```
Create new Node with the song
If the list is empty, set the head to the new node
Else, iterate to the end of the list and set the next node to the new node
```
3) The time complexity is O(*n*)

(Array list)

1) *n* size of the list
2) Pseudo-code
```
    Check if the list needs to be resized
    Add the song to the end of the list
```
3) The time complexity: O(*n*)

(Linked List) public void Song remove();

1) *n* size of the list
2) Pseudo-code
```
If the list is empty return null
Get the song from the head node
Set the head to the next node
Return the song
```
3) The time complexity O(n)

(Array list)

1) *n* size of the list
2) Pseudo-code
```
If the list is empty, return null
Get the last Song in the list
Remove the last element in the list
Return Song
```
3) The time complexity: O(*n*)

(Linked List) public void remove (Song song);

1) *n* size of the list
2) Pseudo-code
```
If the list is empty, return null
If the head node has the song, set the head to the next node and return song
Iterate through the list and find the node with the song
If the song is not found, return null
```

Set the previous node next to the node after the current node and return song

3) Time complexity O(*n*)

(Array List)

1) *n* size of the list
2) Pseudo-code

```
If the list is empty, return null
Iterate through the list and find the index of the song
If the song is not found, return null
Remove the element at the index and return the Song
```

3) Time complexity O(*n*)

(Linked List) public void clear();

1) *n* size of the list
2) Pseudo-code

```
Set the size of the list to 0
```

3) Time complexity: O(*n*)

(Array List)

1) *n* size of the list
2) Pseudo-code

```
Set the size of the list to 0
```

3) Time complexity O(*n*)

(Linked List) public void isEmpty();

1) *n* size of the list
2) Pseudo-code

```
Return true if the head is null, false otherwise
```

3) Time complexity is O(n)

(Array list)

1) *n* size of the list
2) Pseudo-code

```
Return true if the size of the list is 0, false if not
```

3) Time complexity O(*n*)

(Linked List) public void count();

1) *n* size of the list
2) Pseudo-code

```
Set a count var to 0
Iterate through the list and increment the count for each node
Return the count
```

3) Time complexity O(*n*)

(Array List)

1) *n* size of the list
2) Pseudo-code
   Return the size of the list
3) Time complexity O(*n*)

(Linked List) public void get(int i);

1) *n* size of the list, *i* the index of the desired song
2) Pseudo-code
   Set a current variable to the head of the list
   Iterate through the list *i* times, updating the current variable to the next node at each iteration
   Return the song at the node that the current variable is pointing to
3) Time complexity O(n)

(Array List)

1) *n* size of the list, *i* index of the desired song
2) Pseudo-code:
   Return the song at the index i in the list
3) Time complexity O(n)

(Linked List) public boolean contain (Song song)

1) *n* size of the list
2) Pseudo-code
   Set a current variable to the head of the list
   Iterate through the list
   If the node contains the song, return true
   If the end of the list is reached without finding the song, return false
3) Time complexity is O(*n*)

(Array List)

1) *n* size of the list
2) Pseudo-code
   Iterate through the list
   If the current element equals the song, return true
   If the end of the list is reached without finding the song, return false
3) Time complexity is O(*n*)