


Algorithmics	Student information	Date	Number of session
	UO: 276824	21/04/2021	6
	Surname: García Fernández	 Escuela de Ingeniería Informática Universidad de Oviedo	
	Name: Pablo		



Activity 1. Validation Results

The complexity of my algorithm is kind of a subtraction where $a = 3$ since it is being called three times per iterative call and $b = 1$ since we move the level from 1 to 1 and then the complexity is $O(3^{n/1})$ that is $O(3^n)$.

My idea to approach this solution is to have two special methods, the first one is used to check if the added song can stay there because if it surpasses the minutage limit it is rejected, the code for this method is:

```
private static boolean minuteLimit(List<Song> list) {
    boolean res = true;
    int seconds = 0;
    for(Song so : list) {
        seconds += so.getDuration().getTotSec();
    }
    if(seconds <= (LenghtOfBlocks*60)) {
        res = false;
    }
    return res;
}
```

The second method is a method that checks if the song that you tried to add has a better score than another one inside and it tries to fit inside considering the minutes limit and the code is:

Algorithmics	Student information	Date	Number of session
	UO: 276824	21/04/2021	6
	Surname: García Fernández		
	Name: Pablo		

```

private static int compareScore(List<Song> listN, Song song) {
    List<Song> list = listN;
    List<Song> listD = new ArrayList<Song>();
    int initialScore = song.getScore();

    for(int i=0 ; i<list.size() ;i++) {
        if(initialScore > list.get(i).getScore()) {
            listD.add(list.get(i));
        }
    }

    for(int i=0 ; i<listD.size() ;i++) {
        if(minuteLimit(list)) {
            if(song.getScore() > listD.get(i).getScore()) {
                list.remove(listD.get(i));
                if(minuteLimit(list)) {
                    return -1;
                }
            }
            else {
                return i;
            }
        }
    }

    return -1;
}

```

The backtracking method that I used was inspired in the pseudocode that you gave us but it look a little bit different:

Algorithmics	Student information	Date	Number of session
	UO: 276824	21/04/2021	6
	Surname: García Fernández		
	Name: Pablo		

```

private static void backtracking(int level) {
    if(level == numSongs) {
    }
    else {
        for(int i=0 ; i<3; i++) {
            if(i == 0) {
                backtracking(level + 1);
                counter++;
            }
            else if(i == 1) {
                Song song = listOfSongs.get(level);

                if(!listA.contains(song) && !listB.contains(song)) {
                    listA.add(song);

                    if(minuteLimit(listA)) {

                        int test = compareScore(listA, song);
                        if(test != -1) {
                            listA.remove(test);
                        }

                        else {
                            listA.remove(listA.size()-1);
                        }
                    }
                    backtracking(level + 1);
                    counter++;
                }
            }
            else if(i == 2) {
                Song song = listOfSongs.get(level);

                if(!listB.contains(song) && !listA.contains(song)) {
                    listB.add(song);

                    if(minuteLimit(listB)) {

                        int test = compareScore(listB, song);
                        if(test != -1) {
                            listB.remove(test);
                        }

                        else {
                            listB.remove(listB.size()-1);
                        }
                    }
                    backtracking(level + 1);
                    counter++;
                }
            }
        }
    }
}

```

Algorithmics	Student information	Date	Number of session
	UO: 276824	21/04/2021	6
	Surname: García Fernández		
	Name: Pablo		

And my solution to the program is this one:

```
Number of songs: 10
```

```
List of songs:
```

```
id: 3ld4R7 seconds: 4:27 score: 3475
id: 8j4gE3 seconds: 5:22 score: 2834
id: 0fmvy3 seconds: 4:40 score: 3842
id: 8id4R7 seconds: 4:27 score: 3475
id: 9u4gE3 seconds: 6:59 score: 2834
id: 2lsdf9 seconds: 3:22 score: 3842
id: 3j4yQ6 seconds: 5:02 score: 2834
id: 06rwq3 seconds: 4:48 score: 3842
id: 87UKo2 seconds: 3:27 score: 3475
id: 5rtZe9 seconds: 4:44 score: 2834
```

```
Length of the blocks: 20:0
```

```
Total score: 27619
```

```
Total counter: 377
```

```
Best block A:
```

```
id: 2lsdf9 seconds: 3:22 score: 3842
id: 5rtZe9 seconds: 4:44 score: 2834
id: 06rwq3 seconds: 4:48 score: 3842
id: 87UKo2 seconds: 3:27 score: 3475
```

```
Best block B:
```

```
id: 3ld4R7 seconds: 4:27 score: 3475
id: 3j4yQ6 seconds: 5:02 score: 2834
id: 8id4R7 seconds: 4:27 score: 3475
id: 0fmvy3 seconds: 4:40 score: 3842
```

The counter I think it is wrong implemented but the maximum iterations would be 88572, the total score is the correct one and the block are the correct ones too.