Algorithmics	Student information	Date	Number of session
	UO: 276903	30/4/21	7
	Surname: Garriga Suárez		
	Name: Carlos		

Escuela de Ingeniería Informática
Universidad de Oviedo

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Activity 1. An explanation of the heuristic chosen.

In the case of my heuristic, I decided to discard all the states in which any of the list of songs passes over 20 minutes then I set the heuristic value to Integer.MAX_VALUE. In that case that option will never be considered as a part for the solution. As our initial prune limit is set to Integer.MAX_VALUE, we are interested in having the lower one, so, if the lists do not exceed 20 minutes, then I compute which is the score that is going to be obtained if I continue searching in that branch of the tree.

```
        int score = getScorePlayList(blockA) + getScorePlayList(blockB);
        int time1 = getDurationPlayList(blockA);
        int time2 = getDurationPlayList(blockB);

        for(int i = depth; i < allSongs.size(); i++) {
            Song song = allSongs.get(i);

            if(time1 + song.getSeconds() <= MAX_SECONDS) {
                score += song.getScore();
                time1 += song.getSeconds();
            }else if (time2 + song.getSeconds() <= MAX_SECONDS) {
                score += song.getScore();
                time2 += song.getSeconds();
            }
        }

        heuristicValue = -score;
    }
```

This is the way I compute the heuristic value. The thing is to compute the actual score of both blocks A and B and then from the depth in which I am, I get which is the maximum score I will obtain by means of filling both blocks. Then, with that score computed I give it negative value as the lower the heuristic, the better for the solution.

Verification Results

```
Step 9:
Total score: 27619
Best block A:

ID: 0fmvy3,     Seconds: 4:40,  Score: 3842
ID: 2lsdf9,     Seconds: 3:22,  Score: 3842
ID: 3j4yQ6,     Seconds: 5:02,  Score: 2834
ID: 06rwq3,     Seconds: 4:48,  Score: 3842
Best block B:

ID: 3ld4R7,     Seconds: 4:27,  Score: 3475
ID: 8id4R7,     Seconds: 4:27,  Score: 3475
ID: 9u4gE3,     Seconds: 6:59,  Score: 2834
ID: 87UKo2,     Seconds: 3:27,  Score: 3475

Depth: 9

Solution with 9 step(s).

Generated Nodes: 93

Processed Nodes: 82

Trimmed Nodes: 9
```

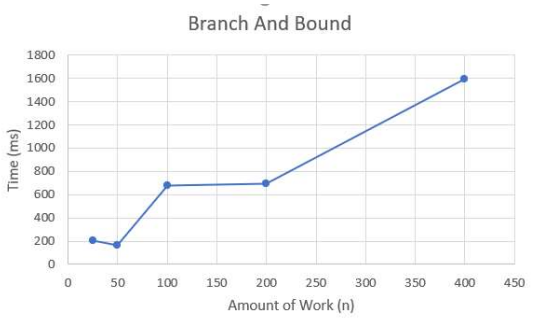
With this heuristic I lead to the best solution possible with 9 steps.

Algorithmics	Student information	Date	Number of session
	UO: 276903	30/4/21	7
	Surname: Garriga Suárez		
	Name: Carlos		

Activity 2. Measures taken.

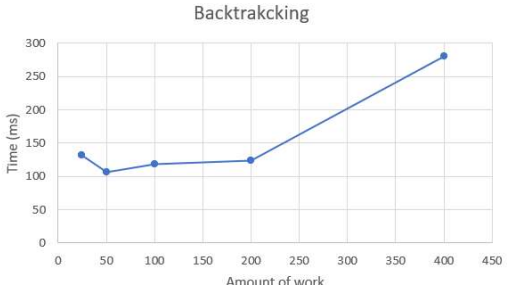
Measures for branch and bound.

n	Time(ms)
25	208
50	165
100	675
200	698
400	1593



Measures for backtracking.

n	Time(ms)
25	132
50	107
100	118
200	124
400	280



The measures for backtracking may not be reliable as they are measured with random songs that can lead to variations of time.

Activity 3. Discussion about Backtracking vs B&B.

The main difference between backtracking and branch and bound is that backtracking is used for decision problems and branch and bound is used for optimization problems.

The thing is that in backtracking have the option to undo something that considers it wrong and choose another way to find a solution while in branch and bound, when it realises that it already has a better solution than the solution before gave, it cuts that previous solution. It completely searches the state space tree to get best solution.

In backtracking, the tree is traversed by means of depth-first search until it finds the solution and in branch and bound the best solution might be found in any state of the tree so this results in searching the full tree.

As a conclusion, backtracking is more efficient than branch and bound.