


Algorithmics	Student information	Date	Number of session
	UO:269546	04-05-21	7
	Surname: Fernández Arias	 Escuela de Ingeniería Informática Universidad de Oviedo	
	Name: Sara		



Activity 1. Heuristic design

For this laboratory session , I've implemented a java class called SongsProblem.java, that contains all the utility logic for the branch and bound method to work.

It contains two internal subclasses: BlockGenerator and BlocksOfSongs.

BlockGenerator will extend BranchAndBound.java. It will represent the whole solution tree.

BlocksOfSongs will extend Node, and it's attributes allow it to represent each different possible state of the solution of the problem. For that, it will represent the nodes of the solution tree.

1.1 BlocksOfSongs explanation.

```

s
/**
 * "Heuristics are criteria, methods or principles for deciding which among
 * several alternative courses of action promises to be the most effective in
 * order to achieve some goal Heuristic function at a node n is an estimate of
 * the optimum cost from the current node n to a Goal and it is denoted by h(n)
 * ." Using that logic
 */
@Override
public void calculateHeuristicValue() {
    int counter = 0;
    if (prune()) { // when we prune a node, we want it to be removed / ignored

        this.heuristicValue = Integer.MAX_VALUE; // if we want this value to be ignored, we put a higher value,
        // representing that it's cost to the goal is not optimal
    } else {
        // we add 1 for each child node that separates the current one from the solution.
        for (int i = getDepth(); i < numberOfSongs; i++) { // there will be a level per song.
            counter++;
        }
        // if the node is at the final level it's heuristic value is 0.
    }

    this.heuristicValue = counter;
}

```

The logic I used for calculating the heuristic value is that , it can be defined as the number of levels that distance the current one from the final solution. If the node should be pruned, it's heuristic value will be <<infinite>> since the greatest the heuristic value is , the further the node is from the solution. Then , it will never be considered as a solution , nor it's attributes will be used in order to search for the solution. A correct solution cannot be obtained from incorrect data.

Algorithmics	Student information	Date	Number of session
	UO:269546	04-05-21	7
	Surname: Fernández Arias		
	Name: Sara		

```

/**
 * Check if the current solution is valid. If not valid, it should be pruned,
 * since no solution starting from an incorrect point can be correct.
 *
 * @return
 */
private boolean prune() {
    if (isValidSolution())
        return false;

    return true;
}

private boolean isValidSolution() {
    if (getBlockDuration(blockA) > blockLength || getBlockDuration(blockB) > blockLength)
        return false;
    else if (blockA.size() > numberOfSongs || blockB.size() > numberOfSongs) {
        return false;
    }

    return true;
}

```

Prune is a simple method, that will get rid of unwanted solutions that would lead to time and memory waste. It will return true (that a node should be pruned) if the node is not valid. That is, if the blocks have a size greater than the number of songs allowed or if their length (time) is greater than the one allowed passed as parameter in the main method.

```

@Override
public boolean isSolution() {
    calculateHeuristicValue();
    return (heuristicValue == 0) ? true : false;
}

```

For isSolution() , I calculate the heuristic value of the node from which it's called. If the heuristic value is 0 the current node is a solution. False otherwise.

Algorithmics	Student information	Date	Number of session
	UO:269546	04-05-21	7
	Surname: Fernández Arias		
	Name: Sara		

Regarding expand() it will create the three children the current node could possibly have:

- the one where the next song is added to block a
- the one where the next song is added to block b
- the one where the song is not added anywhere.

It will just generate children if there are any song left. Else it will return an empty children list.

```

/**
 * To get the child nodes of the current one. The children point to the parent
 * through the inherited parentID field.
 */
@Override
public ArrayList<Node> expand() {
    ArrayList<Node> result = new ArrayList<Node>();
    ArrayList<Song> auxA = new ArrayList<Song>(blockA);
    ArrayList<Song> auxB = new ArrayList<Song>(blockB);
    if (getDepth() < numberOfSongs - 1) {

        // branch and bound involves breadth first search, then, width will be explored
        // first instead of depth.
        int nextLevel = getDepth() + 1; // + 1;
        Song song = allSongs.get(nextLevel);
        // System.out.println(song.toString());
        auxA.add(song); // node: songs is added to block A
        result.add(
            new BlocksOfSongs(auxA, blockB, allSongs, nextLevel, numberOfSongs, blockLength, this.getID()));
        auxB.add(song); // node: songs is added to block A
        result.add(
            new BlocksOfSongs(blockA, auxB, allSongs, nextLevel, numberOfSongs, blockLength, this.getID()));
        // node: songs is added to NONE
        result.add(new BlocksOfSongs(blockA, blockB, allSongs, nextLevel, numberOfSongs, blockLength,
            this.getID()));
        // in this problem ,each node will have 3 children.

        // if no children the returned list is empty.
    }
    return result;
}

```

Algorithmics	Student information	Date	Number of session
	UO:269546	04-05-21	7
	Surname: Fernández Arias		
	Name: Sara		

Activity 2. Measures taken.

When applying the backtracking implementation for the given lists I've obtained the following results:

```
List of songs:
id:3ld4R7 seconds:4:27 score:3475
id:8j4gE3 seconds:5:22 score:2834
id:Ofmvy3 seconds:4:40 score:3842
id:8ld4R7 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:20:00
Total score:27619,0
Total count:31030

Best block A:
id:3ld4R7 seconds:4:27 score:3475
id:Ofmvy3 seconds:4:40 score:3842
id:8ld4R7 seconds:4:27 score:3475
id:3j4yQ6 seconds:5:02 score:2834

Best block B:
id:9u4gE3 seconds:6:59 score:2834
id:2lsdf9 seconds:3:22 score:3842
id:06rwq3 seconds:4:48 score:3842
id:87UKo2 seconds:3:27 score:3475
```

For list1.txt

```
List of songs:
id:3ld4R7 seconds:4:27 score:3475

Length of the blocks:20:00
Total score:0,0
Total count:1

Best block A:

Best block B:
```

For lis3.txt

```
List of songs:
id:3ld4R7 seconds:4:27 score:3475
id:8j4gE3 seconds:5:22 score:2834
id:Ofmvy3 seconds:4:40 score:3842
id:2lsdf9 seconds:3:22 score:3842
id:3j4yQ6 seconds:5:02 score:2234
id:06rwq3 seconds:4:48 score:1132
id:87UKo2 seconds:3:27 score:3475
id:5rtZe9 seconds:4:44 score:2624
id:au4gE3 seconds:6:19 score:2523
id:alsdf9 seconds:0:22 score:3423
id:aj4yQ6 seconds:3:02 score:2252
id:artZe9 seconds:1:42 score:2523
id:bl4R7 seconds:4:12 score:5243
id:bj4gE3 seconds:3:22 score:1234
id:bu4gE3 seconds:6:19 score:2124
id:blsdf9 seconds:3:12 score:3234
id:bj4yQ6 seconds:3:02 score:2133
id:b6rwq3 seconds:4:09 score:3543
id:b7UKo2 seconds:3:12 score:2353
id:brtZe9 seconds:5:41 score:3134

Length of the blocks:20:00
Total score:38838,0
Total count:125297700

Best block A:
id:3ld4R7 seconds:4:27 score:3475
id:Ofmvy3 seconds:4:40 score:3842
id:2lsdf9 seconds:3:22 score:3842
id:87UKo2 seconds:3:27 score:3475
id:alsdf9 seconds:0:22 score:3423
id:aj4yQ6 seconds:3:02 score:2252

Best block B:
id:artZe9 seconds:1:42 score:2523
id:bl4R7 seconds:4:12 score:5243
id:blsdf9 seconds:3:12 score:3234
id:bj4yQ6 seconds:3:02 score:2133
id:b6rwq3 seconds:4:09 score:3543
id:b7UKo2 seconds:3:12 score:2353

List of songs:
id:3ld4R7 seconds:4:27 score:3475
```

For 2txt

With branch and bound I wasn't able to implement it correctly , obtaining the following values:

Algorithmics	Student information	Date	Number of session
	UO:269546	04-05-21	7
	Surname: Fernández Arias		
	Name: Sara		

```
Original:
BLOCK A :
BLOCK B :

Step 1:
BLOCK A :
id:8j4gE3 seconds:5:22 score:2834
BLOCK B :

Step 2:
BLOCK A :
id:8j4gE3 seconds:5:22 score:2834
id:0fmvy3 seconds:4:40 score:3842
BLOCK B :

Step 3:
BLOCK A :
id:8j4gE3 seconds:5:22 score:2834
id:0fmvy3 seconds:4:40 score:3842
id:8id4R7 seconds:4:27 score:3475
BLOCK B :

Step 4:
BLOCK A :
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
BLOCK B :

Solution with 4 step(s).
```

For list1.txt

```
Original:
BLOCK A :
BLOCK B :

Step 1:
BLOCK A :
id:8j4gE3 seconds:5:22 score:2834
BLOCK B :

Step 2:
BLOCK A :
id:8j4gE3 seconds:5:22 score:2834
id:0fmvy3 seconds:4:40 score:3842
BLOCK B :

Step 3:
BLOCK A :
id:8j4gE3 seconds:5:22 score:2834
id:0fmvy3 seconds:4:40 score:3842
id:2lsdf9 seconds:3:22 score:3342
BLOCK B :

Step 4:
BLOCK A :
id:8j4gE3 seconds:5:22 score:2834
id:0fmvy3 seconds:4:40 score:3842
id:2lsdf9 seconds:3:22 score:3342
id:3j4yQ6 seconds:5:02 score:2234
BLOCK B :

Step 5:
BLOCK A :
id:8j4gE3 seconds:5:22 score:2834
id:0fmvy3 seconds:4:40 score:3842
id:2lsdf9 seconds:3:22 score:3342
id:3j4yQ6 seconds:5:02 score:2234
id:06rwq3 seconds:4:48 score:1132
BLOCK B :
```

Solution with 5 step(s).

For 2txt

For list3.txt both results are equal , but I'm aware that my implementation for backtracking had a bug that would add n-1 songs instead of the whole set of songs. I tried to solve it but could not find it since it seems like the reading from the file is done correctly (the set of songs is loaded whole) .