


Algorithmics	Student information	Date	Number of session
	UO: 296503	21/04/2025	7
	Surname: Mulet Alonso	 Escuela de Ingeniería Informática Universidad de Oviedo	
	Name: Sergio		



## Activity 1. Explain what the proposed branching heuristic consists of

To find as soon as possible the solution, I am going to prioritize (sort) nodes by the value of its edges, so we start with the edge that puts the cost near to 0.

Then, we could prune all those edges that with any combination with other nodes put the value higher than 99. For instance, let's say we have a cost of 400 when arriving to one node, and with that solution there are only two more nodes remaining, as the minimum value those nodes may have is -99, it is impossible to accomplish the 99 cost.

Unfortunately, I was not able to implement this in my code, I have try and left some commented code but is not working as expected (no solution or too big execution time)

## Activity 2. NullPathBB.java

For this implementation, I'm reusing utility classes BranchAndBoud, Heap and Node.

I have implemented my own version of Node, called BBNode which is the one that is going to be used by NullPathBB, creating in the constructor the root node, then by means of expand() method, the rest of children are generated.

As I told at the start, I couldn't implement the expand() and initialPruneLimit() that I wanted, so the algorithm in its current state is inefficient.

Algorithmics	Student information	Date	Number of session
	UO: 296503	21/04/2025	7
	Surname: Mulet Alonso		
	Name: Sergio		

## Activity 3. NullPathBBTimes

As I said, my implementation is inefficient, and this is very good shown in the times. I'm not testing with the 100 repetitions because it would be so much time, so impossible to measure.

The complexity should be  $O(n!)$  and there is an interesting thing, from 30 to 25 there is a big jump in the execution time, but then, from 35 to 45 times are quite close, maybe because of pruning

Algorithmics	Student information	Date	Number of session
	UO: 296503	21/04/2025	7
	Surname: Mulet Alonso		
	Name: Sergio		

## Activity4. Execution times and comparing

These are the execution times obtained in branch and bound.

n	t(ms)
20	15695
25	18917
30	31688
35	389758
40	393481
45	395548

And these are the execution times obtained with backtracking

n	t(ms)
200	12,7
205	12,78
210	13,35
215	14
220	14,74
...	...
300	28,29
900	251

If we compare it with previous lab, branch and bound times are way bigger, as I have reached a bad solution.

Theoretically, branch and bound and backtracking should have the same complexity, and in fact my implementations should have. The main problem is that with backtracking I could reach an efficient solution and with branch and bound I couldn't improve the solution.