# **Laboratory practice No. 3: Backtracking**

|  |  |
| --- | --- |
| **Maria Paulina Ocampo Duque**  Universidad Eafit  Medellín, Colombia  mpocampod@eafit.edu.co | **Jose Manuel Fonseca Palacio**  Universidad Eafit  Medellín, Colombia  jmfonsecap@eafit.edu.co |
|  |  |

**3) Practice for final project defense presentation**

**3.1** Algorithms like dijkstra and greedy are used to solve the problem of the shortest path. The first one consists in explore all the paths that start from the origin and lead to the other vertices and the other one consists in go through every node asking which successor has the lowest cost and choose that one to explore it.

**3.2** for list all paths, the number of paths will be n! because each node can be connected with other nodes.

**3.3**

|  |  |
| --- | --- |
| **N value** | **Execution time** |
| 5 | 4 |
| 6 | 2 |
| 7 | 6 |
| 8 | 5 |
| 9 | 10 |
| 10 | 23 |
| 11 | 40 |
| 12 | 97 |
| 13 | 595 |
| 14 | 2572 |
| 15 | 14838 |
| 16 | 110974 |
| 17 | 724601 |

**3.4** To problems where the objective is to find the shortest path, the DFS algorithm will be better because it visits the graph without stop in every node. But if you want to visit the nodes in an orderly way, the most convenient algorithm will be BFS.

**3.7** The variable N represents the quantity of nodes

**3.8** For this problem we implemented the DFS algorithm, with the objective of know the shortest path. This algorithm works in an efficient way because it avoids exploring a node that might not be relevant to the lowest cost.

The DFS compares the nodes and choose the shortest one returns the solution.

***4) Practice for midterms***



1. int res= solucionar(n-a,a,b,c);

2.res=Math.max(res,solucionar(n-b,a,b,c)

3. res=Math.max(res,solucionar(n-c,a,b,c)

* 1. *length-1*

1. graph.length

2. sePuede(v,graph,path,pos)

3.cicloHamiAux(graph,path,pos+1)

***4.5***

1. return 1 + lcs(i-1, j-1,s1,s2)

2. return MathMax(ni,nj)

3. T(n)=T(n-1)+T(n-1)

***4.7***

1. if (r<0)

2. a[r]=i

3. r +1