Laboratory practice No. 5: Divide to conquer and Dynamic programming

Kevin Alexander Herrera
Universidad Eafit
Medellín, Colombia
kaherrerag@eafit.edu.co

Jose Joab Romero
Universidad Eafit
Medellín, Colombia
jjromeroh@eafit.edu.co

3) Practice for final project defense presentation

3.1 The data structure that the program use is a matrix of costs and another matrix of visited vertices. And the algorithm travel through the matrix, and the horizontal index of the vertex will represent the power set of the verteces not including the origen vertex which is the final vertex too. While the algorithm creates previus values of the costs of going from one vertex to another going through other. So a lot of costs depends of the previus values, so, it can return and take them without doing the process again.

3.2 The program is going to have to do n times the power set of a set made up of n-1 elements so it is going to be n*(2^(n-1)), arround 2,81x(10^16) operations

3.3 The algorithm works searching the shortest path that the delivery man have to cross for going by every vertex in the graph but obviusly in the reality the delivery man tooks shortcuts, so the algorithm must have as verteces the points of delivery. And the distance between two points of delivery is calculated directly if there isn't another vertex between them, or searching which path is the shortest going by another point.

3.4 The data structure used is a matrix of String type, that will be fill by "K" that is the position of Karolina, "R" that is the position of the radioactive waste, and numbers that are the distance from Karolina to that exactly point. So, the algorithm completes that numbers taking into account the position of Karolina while she get every waste, and acumulate the distance that Karolina cross.

3.5 the complexity of the algorithm is O((XxY)xR)

3.6 O((XxY)xR) being X and Y the dimensions of the board and R the number of radioactive waste.

4) Practice for midterms

4.1.1

		C	A	L	L	E
	0	1	2	3	4	5
C	1	0	1	2	3	4
A	2	1	0	1	2	3
S	3	2	1	1	2	3
A	2	1	0	1	2	3

4.1.2

		M	A	D	R	E
	0	1	2	3	4	5
M	1	0	1	2	3	4
A	2	1	0	1	2	3
M	3	2	1	1	2	3
A	4	3	2	2	2	3

4.2.1 O(i*j)
4.2.2 return table[lenx][leny];
4.3.1 O(n)
4.3.2 a.
4.4.1 c.
4.5.1 c.
4.5.2 a[mitad]
4.5.3 a,mitad+1,der,z
4.6.1 scm[i]=arr[i];
4.6.2 max++;
4.6.3 max=scm[i];
4.6.4 O(n^2)
4.7.1 d[i][j]
4.7.2 d[i, k-1]
4.7.3 d[k-1, j]
4.7.4 O(n^2)