**Algorithm Lab**

**Week 6: 0/1 Knapsack Problem**

Text, letter

Description automatically generated

1. Analyze space and time complexity of a recursive implementation without cache.

Produces a recursion tree in which a same subproblem is computed more than one time so

Time Complexity: O(2^C).

Since it doesn’t use cache, it has constant space

Space Complexity: O(1)

2. Design a table to cache the answer of subproblems.

Given a capacity C, and a set with n elements create an array K[][], considering the capacity from 1 to C and the weights as rows. K[i][j] will contain the maximum value encountered when analyzing Wi. When analyzing Wi+1 it will have Wi for reference. This is done for the values from 1 to C.

3. Analyze space and time complexity of implementation at Q2.

N is the number of elements and C is the capacity. The operation of comparison is a set value, and it is performed for ever element of the array so

Time Complexity: O(N\*C).

Since it uses a 2d array to store the values

Space Complexity :O(N\*C)

4. Please explain why the algorithm is a pseudo polynomial time algorithm (kind of

exponential time algorithm), not a polynomial time algorithm.

It is pseudo-polynomial because the number of computations that have to be done depend on the value given as a maximum capacity and not on the number of values given in the array.