Reading material

At the lecture we will continue with dynamic programming. We will talk about the knapsack problem and sequence alignment. You should read KT section 6.4 and 6.6.

Exercises

1 [w] **Knapsack** Solve the following knapsack problem by filling out the table below. The items are givens as pairs (w_i, v_i): $(5,7),(2,6),(3,3),(2,1)$. The capacity $W = 6$.

4							
3							
2							
1							
0							
$i \setminus w$	0	1	2	3	4	5	6

2 [w] **Sequence alignment** Consider the strings `APPLE` and `PAPE` over the alphabet $\Sigma = \{$`A`,`E`,`L`,`P`$\}$ and a penalty matrix P:

	A	E	L	P
A	0	1	3	1
E	1	0	2	1
L	3	2	0	2
P	1	1	2	0

Compute the sequence alignment of the two strings when the penalty for a gap $\delta = 2$. Fill the dynamic programming table below, and explain how the minimum cost sequence alignment is found in it.

	j	0	1	2	3	4	5
i			A	P	P	L	E
0							
1	P						
2	A						
3	P						
4	E						

3 **Book Shop** You are in a book shop which sells n different books. You know the price h_i and number of pages s_i of each book $i \in \{1, \ldots, n\}$. You have decided that the total price of your purchases will be at most x and you will buy each book at most once.

3.1 Give an algorithm that computes the maximum number of pages you can buy. Analyse the time and space usage of your algorithm.

3.2 Modify your algorithm to only use $O(x)$ space (if it doesn't already). It is possible to solve the problem using only a single 1-dimensional array D.

3.3 Implement your linear space algorithm on CSES: https://cses.fi/problemset/task/1158

4 Longest palindrome subsequence A *palindrome* is a (nonempty) string over an alphabet Σ that reads the same forward and backward. For example are `abba` and `racecar` palindromes. A string P is a *subsequence* of string T if we can obtain P from T by removing 0 or more characters in T. For instance, `abba` is a subsequence of `bcadfbbba`.

Give an efficient algorithm to find the longest palindrome that is a subsequence of a given input string. To do this first give a recurrence for the problem and then write pseudo code for an algorithm based on your recurrence and dynamic programming. Argue that your recurrence is correct and analyse the running time and space usage of your algorithm.

5 Defending Zion Solve KT 6.8

Puzzle of the week: The Blind Man A blind man was handed a deck of 52 cards with exactly 10 cards facing up. How can he divide it into two piles, each of which having the same number of cards facing up?