1 Dynamic Programming

1.1 Knapsack

Assuming $w > 0$, $i \geq 0$ and $w \geq w_i$

$$DP[w][i] = \max(V_i + DP[w - w_i][i-1], DP[w][i-1])$$

Base cases:

$$DP[w][i] = \begin{cases} DP[w][i] & \text{if } w < w_i \\ 0 & \text{if } i < 0 \text{ or } w \leq 0 \end{cases}$$

1.2 Matrix Chain Multiplication

Given a monoid M with operation $\cdot$, a cost function $c : M^2 \to \mathbb{R}_{>0}$, and a finite sequence of elements of M, $a_1, \ldots, a_n$, we can calculate the minimum cost of operating the elements (i.e. where to put the parenthesis on $a_1 \cdot \ldots \cdot a_n$ such that the total cost is minimized (the cost function is used when every operation is done)) with the following DP:

$$DP[i][j] = \min_{i \leq k < j} (c(A_{i,k}, A_{k+1,j}) + DP[i][k] + DP[k+1][j])$$

Where $1 \leq i \leq j \leq n$ and $A_{i,j} = a_i \cdot \ldots \cdot a_j$

1.3 Longest Increasing Subsequence

Given a sequence $a_1, \ldots, a_n$, we can find the LIS using the following method, using an auxiliary array L of size n and a counter $currL$, set $L[0] = a_1$ and $currL = 1$ then, in the sequence order, for each element a_i search for the lower bound $L[j]$ of a_i between $L[0]$ and $L[currL - 1]$, if $j = currL - 1$ then set $L[currL] = a_i$ and increase $curr_l$ by one, in the other case set $L[j] = a_i$.