12 Subset-Sum and Knapsack [KT 6.4]

Definition of subset-sum. Find a subset S that maximizes $\sum_{i \in S} w_i$, subject to the constraint that $\sum_{i \in S} w_i \leq W$.

Definition of Knapsack. Find a subset S that maximizes $\sum_{i \in S} v_i$ where v_i is the value of i, subject to the constraint that $\sum_{i \in S} w_i \leq W$.

We will start from subset-sum.

Greedy does not work. Sort the items by decreasing weight, and then select items in this order as long as the total weight remains below W.

Counter-example: $\{W/2 + 1, W/2, W/2\}$.

If we sort items by increasing weight, then it fails on input $\{1, W/2, W/2 + 1\}$.

False start. If $n \notin \mathcal{O}$, then $OPT(n) = OPT(n-1)$. If $n \in \mathcal{O}$, then what? There is no analogy of deleting all the conflict intervals as that in the case of weighted interval scheduling, since after accepting n we have $W - w_n$ budget left. Therefore w_n needs to be in the picture.

The right solution. This suggests us to use more subproblems: for each initial set $\{1, \ldots, i\}$ ($i \leq n$), and each value $\{1, \ldots, w\}$ ($w \leq W$). New recursion:

1. If $n \notin \mathcal{O}$, then $OPT(n, W) = OPT(n-1, W)$.
2. If $n \in \mathcal{O}$, then $OPT(n, W) = w_n + OPT(n-1, W - w_n)$.

We should take the larger of the two:

$$OPT(n, W) = \max\{w_n + OPT(n-1, W - w_n), OPT(n-1, W)\}.$$

For the base cases, we have $OPT(i, W) = 0$ for any $1 \leq i \leq n$ and $W \leq 0$, and $OPT(i, W) = 0$ for $i = 0$ and any W.

Extend this to Knapsack. Just replace w_n with v_n and that's all.