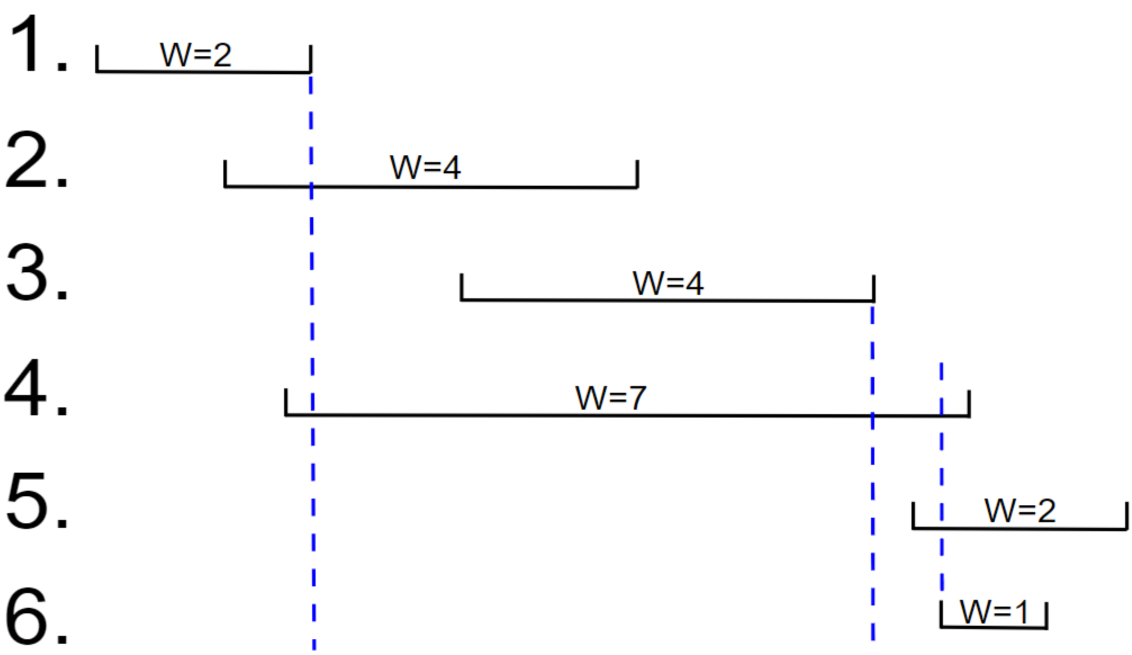
Lecture 11: Dynamic Programming

Weighted Interval Scheduling:

Choose disjunct intervals to maximize the sum of the weight of these chosen intervals $\sum_{i \in C} W_i$, C

is the set of the chosen intervals. For each interval i , it starts at time S_i and ends at time F_i. I is the set of all intervals

An example is shown below in the picture:



Observation:

- Greedy algorithms like "earliest finishing time" and "Highest weight first" don't work well for this problem.
- I_i either in C or not.
- Suppose $I_1, I_2, \ldots, I_n$ rank in finish time F_i: $F_1 \leq \cdots \leq F_i \leq \cdots \leq F_n$.

Define:

- **Prep(i)**: the maximum j that I_j doesn't overlap with I_n and $j < n$.
- **OPT[K]**: the optimum subset of $\{I_1, \cdots, I_k\}$ which has the greatest sum of weight.
- **W[k]**: the value(sum of weight) of OPT[k].
- Thus: $OPT[n]$ = $\begin{cases} OPT[n-1], & I_n \notin C \\ \\ I_n + OPT[Prep(k)], & I_n \in c \end{cases}$
- Still take the picture shown above as example: $Prep(1) = Prep(2) = Prep(4) = 0, Prep(3) = 1, Prep(5) = Prep(6) = 3$

Algorithm:

1. Input $I_1, \cdots, I_n$, for each I_n has: S_i, F_i, W_i
2. Pre-processing:
   - Calculate $Prep(k), k \in \{1, 2, \cdots, n\}$
   - $Prep(k) = max\{j|F_j < S_k\}$,
   - $Prep(k) = 0, \;\; if \;\; \{j|F_j < S_k\} = \emptyset$
3. Initiate :
   - $W[k] = NULL, k \in \{1, 2, \cdots, n\}$
   - $OPT[k] = NULL, k \in \{1, 2, \cdots, n\}$
   - $W[0] = 0, OPT[0] = \emptyset$.
4. Loop or recursive solution:

- Loop:
  1. For $k = 1, 2, \cdots, n$:
     1. if $W_k + W[Prep(k)] > W[k-1]$:
        1. $OPT[k] = \{I_k\} \bigcup OPT[Prep(n)]$
        2. $W[k] = W_k + W[Prep(k)]$
     2. else:
        1. $OPT[k] = OPT[k-1]$
        2. $W[k] = W[k-1]$
  2. End for
  3. Return $W[n], OPT[n]$

- Recursive:
  1. Optimum(k) =
     1. if $W_k \neq NULL$:
        1. return $W[k] and OPT[k]$
     2. else:
        1. $Optimum(k-1)$
        2. $Optimum(Prep(k))$
        3. if $W_k + W[Prep(k)] > W[k-1]$:
           1. $OPT[k] = \{I_k\} \bigcup OPT[Prep(n)]$
           2. $W[k] = W_k + W[Prep(k)]$
        4. else: $OPT[k] = OPT[k-1], \;\; W[k] = W[k-1]$