1 Interval Scheduling

Given n jobs sort them based on their finishing time, and perhaps after renaming, assume that $f(1) \leq f(2) \leq \cdots \leq f(n)$.

Now for $1 \leq j \leq n$, define $OPT(j)$:=the sum of the weights of the maximum weight compatible set of jobs among $1, \ldots j$ with respect to the above sorted order.

Base Case: $OPT(0) = 0$ since we have no jobs.

IH: Suppose we have computed $OPT(i)$ for all $i < j$ for some $j \geq 1$.

IS: We want to find $OPT(j)$. First, we guess whether job j is in the optimum solution or not.

- **Case 1: Job j is chosen in the optimum**. Then, every job not compatible with j are not in OPT(j). Let $p(j)$ be the largest index job which end before start of job j start, i.e., $p(j) = \max\{i : f(i) \leq s(j)\}$.

  We claim that all jobs $1, \ldots, p(j)$ are compatible with j and all jobs $p(j)+1, \ldots, j-1$ are not compatible. This is because of sorting: Every $i \leq p(j)$ satisfies $f(i) \leq f(p(j)) \leq s(j)$ so it is compatible. On the other hand, because $p(j)$ is the largest index, every job $i > p(j)$ satisfies $s(j) < f(i) \leq f(j)$ so it is in-comaptible.

  Using the above claim, when including job j, the rest of jobs chosen in $OPT(j)$ must be the maximum weight set of compatible jobs from $1, \ldots, p(j)$. But, that is exactly the subproblem $OPT(p(j))$. So, in this case we have $OPT(j) = v_j + OPT(p(j))$.

- **Case 2: Job j is not in the optimum**. Then, we can simply remove j and $OPT(j)$ would be the maximum weight set of compatible jobs in the range $1, \ldots, j-1$, $OPT(j) = OPT(j-1)$.

Since OPT can take the best of the above two cases, and we have a maximization problem,

$$OPT(j) = \max\{OPT(j-1), v_j + OPT(p(j))\}.$$

Once we compute $OPT(j)$ for all j we can simply output, $OPT(n)$.