Problem:
1 machine, it can be working on at most one job at a time
n jobs \( J_1, \ldots, J_n \)

unit length, non-preemptive
release time 0
once the job has started, it may not stop until it has completed
deadline \( d_j \in \mathbb{Z}^+ \)
profit \( g_j \in \mathbb{Z}^+ \)

goal: feasibly schedule a subset \( S' \) of jobs where \( S \) has max profit.
\[
\sum_{j \in S'} g_j \geq \sum_{j \in S} g_j
\]

Examples:

\[
\begin{array}{c|c}
\hline
\text{job} & \text{profit} \\
\hline
J_1 & 20 \\
J_2 & 15 \\
J_3 & 10 \\
J_4 & 7 \\
J_5 & 5 \\
J_6 & 3 \\
\hline
\end{array}
\]

The greedy algorithm for this problem bears heavy resemblance to Kruskal's alg. for minimum spanning trees.

High-level of Greedy Scheduler:
- sort jobs s.t. \( g_1 \geq g_2 \geq \ldots \geq g_n \)
- \( J \leftarrow \emptyset \)
- for each job \( j \) from 1 to \( n \) do
  - if feasible \( \text{feas}(J \cup \{j\}) \) do
    - \( J \leftarrow J \cup \{j\} \)

(Recall Kruskal's alg):
- sort edges s.t. \( w(e_1) \leq w(e_2) \leq \ldots \leq w(e_m) \)
- \( T \leftarrow \emptyset \)
- for each edge \( e \) from 1 to \( m \) do
  - if feasible \( \text{feas}(T \cup \{e\}) \) do
    - \( T \leftarrow T \cup \{e\} \)

Oracle telling us whether \( J \cup \{j\} \) is a feasible set of jobs, i.e. whether there exists a schedule of \( J \cup \{j\} \).

Nevermind for now how the oracle determines whether a set of jobs is feasible. We will come back to this.

Note: An optimal schedule having no idle time.
Claim: Greedy Scheduler maximizes profit.

Proof: 

- let J denote jobs satisfied in GS' solution and $S_J$ be some feasible schedule of jobs J.
- let I denote jobs satisfied by the optimal solution (OPT); $S_I$ a feasible schedule of I.

one can rearrange $S_J$ and $S_I$ into $S'_J$ and $S'_I$ s.t. any $j \in I \cap J$ is done in some slot, $S'_J$ and $S'_I$:

for every job $a \in I \cap J$:

(i) if a in same slot in $S_J$ and $S_I$, nothing to rearrange.

(ii) if $S_J$ schedules a earlier than $S_I$, schedules a,

\[ S_J \rightarrow a \leftarrow b \rightarrow \]
\[ S_I \rightarrow a \rightarrow \]
\[ d_a \text{ is after this point} \]

(b may be nothing if $S_J$ didn't schedule anything in the slot where $S_I$ scheduled a. Can still "swap")

(iii) if $S_J$ sch's a later than $S_I$ sch's a,

\[ S_J \rightarrow a \rightarrow \]
\[ S_I \rightarrow a \leftarrow b \rightarrow \]

in $S_I$, swap a with b. again, still feasible.

- once job a has been moved into agreement, it never needs to move again. can repeat this argument for all of $I \cap J$, each time decreasing number of common but unsynced jobs.

so profit ($S'_J$) = profit($S_J$) and profit ($S'_I$) = profit($S_I$).

- $S'_J$ and $S'_I$ can still look different, but only because $I \neq J$. How can this happen?

Case 1: $S'_J$ \[ \rightarrow a \rightarrow \] some job a is sched in $S'_J$ opposite an empty slot in $S'_I$ and $a \notin I$.

\[ S'_J \rightarrow f \rightarrow \] Empty

% contradicts optimality of I since $I \cup IaJ$ is feasible and more profitable.

Case 2: $S'_J$ \[ \rightarrow b \rightarrow \] some job a is sched in $S'_I$ opposite empty slot in $S'_J$ and $a \notin J$.

\[ J \cup IaJ \text{ is feasible and Greedy Scheduler wouldn't have skipped it.} \]

\[ J \cup IaJ \rightarrow a \rightarrow \]

Case 3: $S'_J$ \[ \rightarrow a \rightarrow \] $a \notin I$.

\[ S'_I \rightarrow b \rightarrow \] $b \notin J$.

Case 3.1: $g_a > g_b$. $I \{b, v, IaJ\}$ is more profitable than I.

% optimality of I.

Case 3.2: $g_a < g_b$. then Greedy Scheduler skipped over b to eventually pick a. % greediness

Case 3.3: $g_a = g_b$. How is the only thing that can happen.

\[ t \text{ timeslots} t, \text{ at } S'_J \text{ and } S'_I \text{ schedule} \]

- no jobs
- same job
- two jobs with same profit.

\[ \Rightarrow \text{profit of } S'_J = \text{profit of } S'_I \]

\[ \Rightarrow \text{schedule } S'_I \text{ yields optimal (i.e. maximum) profit.} \]
Determining feasibility:

FeasOracle(J): for each $i \in J$, schedule $i$ at $t$, the latest possible free slot $t \leq \min(n, d_i)$.

Lemma: $J$ is feasible iff FeasOracle(J) returns a feasible solution.

Proof: $\Leftarrow$: trivial.

$\Rightarrow$: suppose $J$ is feasible.

then $\exists$ a feas. sch.

then $\exists$ a feas. sch. scheduling all jobs in first $|J|$ timeslots "left-shifted sch."

Can always move jobs earlier.

Suppose FeasOracle(J) does not return a feasible sol'n, i.e. $\exists$ job $i \in J$ s.t. FeasOracle was unable to add it before $\min(|J|, d_i)$

since slot $s$ is empty, all $(s-1)$ jobs scheduled here have deadline $\leq s-1$

$\therefore$ $J$ has at least $s$ jobs, each of whose deadline is $\leq s$.

By Pigeonhole Prin., $J$ cannot be feasible.

How to implement Greedy Scheduler with FeasOracle: note that FeasOracle doesn't specify order in which jobs of $J$ are added to schedule. We will choose to add them in same order of non-increasing $g_j$ so that we don't have to rebuild the sch. from scratch with each oracle call.

Greedy Scheduler details

- Sort jobs s.t. $g_1 \geq g_2 \geq \ldots \geq g_n$ (compute $d_{\max}$ along the way)

- for each $t \leftarrow 1$ to $\min(n, d_{\max})$ do
  - $S[t] \leftarrow NIL$ (Schedule)
  - free[$t$] $\leftarrow t$ (latest free slot earlier than or equal to $t$)

- for each job $j \leftarrow 1$ to $n$ do
  - $m \leftarrow \min(n, d_j)$ (get latest free slot $\leq \min(n, d_j)$)
  - if $m > 0$ do
    - $S[m] \leftarrow j$ (schedule $j$ there)
    - $m' \leftarrow m$
    - while $S[m'] \neq NIL$ do
      - $free[m'] \leftarrow free[m'-1]$
      - $m' \leftarrow m' + 1$

  - Total time: $O(n \log n) + O(n^2) \Rightarrow O(n^2)$. 

m' iterates over this

free[m-i] former

free[\min(n,d_j)]