Review: Hallmark of a Greedy Algorithm
- Sort data according to some criteria
- Consider each piece of data in sorted order and make a local decision
- Result is globally optimal (if this problem is amenable to a greedy solution)
- Complexity is generally no better than $O(n \log n)$ due to the sort
- Important to prove that the solution is optimal

Review: Interval Scheduling: Greedy Solution
- **Greedy template**: Consider jobs in some order. Take each job provided it's compatible with the ones already taken.
- **Idea 4**: Earliest finish time. Consider jobs in ascending order of finish time $f_j$. 

![Interval Scheduling Graph]
Review: Interval Scheduling: Greedy Solution

- **Greedy template.** Consider jobs in some order. Take each job provided it's compatible with the ones already taken.
- **Idea 4: Earliest finish time.** Consider jobs in ascending order of finish time $f_j$.

![Bar chart showing intervals]

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
<td>b</td>
<td>e</td>
<td>h</td>
<td>e</td>
<td>a</td>
<td>b</td>
<td>h</td>
</tr>
</tbody>
</table>

Review: Earliest Finish Time - Optimal Solution

Sort jobs by finish times so that $f_1 \leq f_2 \leq \ldots \leq f_n$.

```plaintext
A ← {};
for j = 1 to n {
  if (job j compatible with A)
    A = A ∪ {j};
}
return A;
```

Review: Proof of Optimality

**Theorem.** Greedy algorithm is optimal.

**Proof by contradiction:**
Assume greedy is not optimal. Then the greedy solution must have fewer jobs than the optimal one.
Let $I_1, I_2, \ldots, I_r$ denote set of jobs selected by greedy.
Let $J_1, J_2, \ldots, J_m$ denote set of jobs in the optimal solution where the first $r$ jobs in the greedy and optimal solution are the same.
Consider the $(r+1)$th job.
Interval Partitioning

Lecture $j$ starts at $s_j$ and finishes at $f_j$.
Goal: find minimum number of classrooms to schedule all lectures so that no two occur at the same time in the same room.

Interval Partitioning Lower Bound

The depth of a set of intervals is the maximum number that contain any given time.

Key observation. Number of classrooms needed $\geq$ depth.

Example: Depth of schedule below = 3

Question: Does there always exist a schedule equal to depth of intervals?
Interval Partitioning: Greedy Solution

Sort intervals by starting time so that $s_1 \leq s_2 \leq \ldots \leq s_n$.

$d \leftarrow 0$ // Number of classrooms

for $j = 1$ to $n$
    if (lecture $j$ is compatible with some classroom $k$)
        schedule lecture $j$ in classroom $k$
    else
        allocate a new classroom $d + 1$
        schedule lecture $j$ in classroom $d + 1$
        $d \leftarrow d + 1$

Scheduling to Minimize Lateness

- Single computer processes one job at a time.
- Job $j$ requires $t_j$ units of processing time.
- Job $j$ has a deadline $d_j$ by which it must be done.
- If $j$ starts at time $s_j$, it finishes at time $f_j = s_j + t_j$.
- Lateness: $l_j = \max \{0, f_j - d_j\}$.
- Goal: schedule all jobs to minimize maximum lateness $L = \max_l_l_j$.

Scheduling Example

<table>
<thead>
<tr>
<th>Job</th>
<th>$t_j$</th>
<th>$d_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>15</td>
</tr>
</tbody>
</table>