Reading: 4.1–4.2

Last Time:
- computational tractibility
- Big-oh

Today:
- Big-Oh (cont.)
- Graph Review
- Greedy Algorithms
- Interval Scheduling

Some common runtimes
- constant,
- logarithmic,
- linear,
- \( n \log n \),
- quadratic,
- cubic,
- exponential

Recall: \( T(n) \) is worst-case runtime for instance of size \( n \).

Recall: \( T(n) \) is \( O(f(n)) \) if \( \exists n_0, c > 0 \) such that \( \forall n > n_0, T(n) < cf(n) \).

Recall: \( \Omega(\cdot) \) and \( \Theta(\cdot) \).

Example: \( T(n) = 5n^2 - n \) is:

\[
\begin{array}{c|ccc}
\quad & O(\cdot) & \Omega(\cdot) & \Theta(\cdot) \\
\hline
n^3 & \quad & \quad & \\
n^2 & \quad & \quad & \\
n & \quad & \quad & \\
\end{array}
\]
Greedy Algorithms

- build solution in steps.
- each step myopically optimal
- hard part: prove final solution is optimal

Question: For what problems are greedy algorithms optimal?

Scheduling

- many tasks competing for limited resources.
- temporal constraints.
  - start & end times,
  - deadlines, and
  - one job at a time.
- find most efficient schedule.
  - most tasks schedules, or
  - best tasks scheduled

Example: CPU scheduling.

Interval Scheduling

“sharing a single resource”

Input:

- $n$ jobs
- one machine
- requests: job $i$ needs machine between times $s(i)$ and $f(i)$

Goal: schedule to maximize # of jobs scheduled.

Examples: Greedy by . . .

- “start time”

```
--- --- --- --- ---
-------------------
```

- “smallest size”

```
--------- ---------
------
```

2
• “fewest incompatibilities”

Greedy Algorithm for Interval Scheduling

Idea: scheduling the earliest finish time first, leaves the least constraints on remaining schedule.

Def: jobs $i$ and $j$ are

- incompatible if
  \[[s(i), f(i)] \cap [s(j), f(j)] \neq \emptyset \]
- otherwise compatible.

- set $S$ is compatible if all $i, j \in S$ are compatible.

Examples:

----- or ---- or ---

----- or ----- or ---

Algorithm: Greedy by Min. Finish Time

1. $S = \emptyset$
2. Sort jobs by increasing finish time.
3. For each job $j$ (in sorted order):
   - if $j$ if compatible with $S$
     schedule $j$: $S \leftarrow S \cup \{j\}$
   - else discard $j$

Analysis

Runtime

\[ T(n) \leq n \log n + \sum_{j} \]

sort

\approx n \log n + n^2

= $O(n^2)$. 

Idea: Job $j$ in alg. is compatible if it is compatible with last scheduled job.

\[ T(n) = n \log n + n \]

= $\Theta(n \log n)$
Correctness

“schedule is compatible and optimal”

Lemma 1: schedule of algorithm is compatible

Proof: (by induction, straightforward)

Def:

- let $i_1, \ldots, i_k$ be jobs scheduled by greedy
- let $j_1, \ldots, j_m$ be jobs scheduled by OPT

Goal: show $k = m$.

Approach: “Greedy Stays Ahead”

Lemma 2: for $r \leq k$, $f(i_r) \leq f(j_r)$

Proof: (induction on $r$)

base case: $r = 0$

- add dummy job 0 with $s(0) = f(0) = -\infty$
- only change: OPT and GREEDY schedule dummy
- so $f(i_0) = f(j_0)$

inductive hypothesis: $f(i_r) \leq f(j_r)$

inductive step:

- Let $I = \{\text{jobs starting after } f(i_r)\}$
- $J = \{\text{jobs starting after } f(j_r)\}$

- IH $\Rightarrow J \subseteq I$
- Alg $\Rightarrow f(i_{r+1}) = \min_{j \in I} f(j)$

\[
\leq \min_{j \in J} f(j) \\
\leq f(j_{r+1}).
\]

Theorem: Greedy alg. is optimal

Proof: (by contradiction)

- OPT has job $j_{k+1}$ but greedy terminates at $k$.
- lemma 2 (with $r = k$)

\[f(i_k) \leq f(j_k)\] (1)

- $j_{k+1}$ is compatible with $j_k$

\[f(j_k) \leq s(k_{k+1})\] (2)

- (1)\&(2)

\[f(i_k) \leq s(j_{k+1})\]

\[j_{k+1} \text{ is compatible with } i_k\]

\[\Rightarrow \text{alg doesn’t terminate at } k\]
**Graphs**

“encode pair-wise relationships”

**Examples:** computer networks, social networks, travel networks, dependencies.

\[ G = (V, E) \]

- **Vertices:** \( V = \{1, 2, 3, 4\} \)
- **Edges:** \( E = \{(1, 2), (2, 3), (2, 4), (3, 4)\} \)

**Concepts**

- degree
- neighbors
- paths, path length
- distance
- connectivity, connected components
- directed graphs.

**Graph Traversals**

“visit all the vertices in a connected component of graph”

- **Breadth First Search (BFS).**
  
  Example: BFS from 1: 1, 2, 3, 4 or 1, 3, 2, 4.

- **Depth First Search (DFS).**
  
  Example: DFS from 1: 1, 2, 4, 3 or 1, 3, 4, 2.