Outline

- Activity selection
- Fractional knapsack
- Huffman encoding
- Later:
  - Dijkstra (single source shortest path)
  - Prim and Kruskal (minimum spanning tree)
Optimization problems

- A class of problems in which we are asked to find a set (or a sequence) of “items” that satisfy some constraints and simultaneously optimize (i.e., maximize or minimize) some objective function

Greedy method

- Typically applied to optimization problems, that is, problems that involve searching through a set of configurations to find one that minimizes/maximizes an objective function defined on these configuration
- Greedy strategy: at each step of the optimization procedure, choose the configuration which seems the best between all of those possible
Searching for the global minimum

Greedy method

- There are problems for which the globally optimal solution can be found by making a series of locally optimal (greedy) choices
  - Make whatever choice seems best at the moment and then solve the sub-problem arising after the choice is made
  - The choice made by a greedy algorithm may depend on choices so far, but it cannot depend on any future choices or on the solutions to sub-problems
- The greedy strategy does not always lead to the global optimal solution
Searching for the global minimum

Elements of greedy strategy

- Two ingredients that are exhibited by most problems that lend themselves to a greedy strategy
  - **Greedy-choice property**: a globally optimal solution can be reached by making a locally optimal choice
  - **Optimal substructure**: optimal solution to the problem consists of optimal solutions to sub-problems
Activity selection

(aka, “task scheduling”)

Activity Selection

• **Input**: A set of activities $S = \{a_1, \ldots, a_n\}$
• Each activity has start time and a finish time $a_i = (s_i, f_i)$
• Two activities are *conflicting* if and only if their interval overlap
• **Output**: a maximum-size subset of non-conflicting activities
Activity Selection

• Here are a set of start and finish times

<table>
<thead>
<tr>
<th></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>$s_i$</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>8</td>
<td>8</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>$f_i$</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
</tbody>
</table>

• What is the maximum number of activities that can be completed?
  – $\{a_3, a_9, a_{11}\}$ can be completed
  – But so can $\{a_1, a_4, a_8, a_{11}\}$ which is a larger set
  – But it is not unique, consider $\{a_2, a_4, a_9, a_{11}\}$
“Greedy” Strategies

1. Longest first
2. Shortest first
3. Early start first
4. Early finish first
5. None of the above
Early Finish Greedy strategy

- Sort the activities by finish time
- Schedule the first activity
- Then, schedule the next activity (in sorted list) which starts after previous activity finishes (first non-conflicting task)
- Repeat until no more activities
Activity selection in Python

```python
def greedy_activity_selection(A):
    A.sort(key=itemgetter(1))  # Remark: sort A by finish time
    result = [A[0]]  # Remark: first activity in the solution
    i = 0
    for j in range(1,len(A)):
        if A[j][0] >= A[i][1]:  # Remark: start[j] >= finish[i]
            result.append(A[j])
            i = j
    return result
```

Time complexity? $O(n \log n)$ to sort, the rest is linear.

Greedy

- Goal: build a solution in steps, never make a “mistake”, i.e., maintain the invariant that the partial solution so far is always extendible to an optimal solution
- Choosing the earliest finish time activity for the first job maximizes the time to schedule the largest set of remaining non-conflicting jobs
Correctness (optimality)

- **Greedy choice property**: The first choice is consistent with some optimal solution

- **Optimal substructure property**: After the first choice, to solve the entire problem optimally, it is enough to solve the remaining sub-problem optimally

Greedy-Choice Property

- **Claim**: There is an optimal solution that begins with a greedy choice (i.e., with the first activity, which has the earliest finish time)
Greedy-Choice Property

- **Proof.** Suppose \( A \subseteq S \) is an optimal solution
  - Order the activities in \( A \) by finish time
    Let \( k \) be the first activity in \( A \)
    - If \( k = 1 \), the schedule \( A \) begins with a greedy choice
    - If \( k \neq 1 \), show that there is another optimal solution \( B \) that begins with the greedy choice (activity 1)
  - Let \( B = A \setminus \{k\} \cup \{1\} \)
    - Activities in \( B \) are non-conflicting because activities in \( A \) are non-conflicting, \( k \) is the first activity to finish and \( f_1 \leq f_k \)
    - \( B \) has the same number of activities as \( A \) thus, \( B \) is optimal

Optimal Substructure

- After the greedy choice of the first activity, the problem reduces to finding an optimal solution for the activity-selection problem over those activities in \( S \) that are compatible with the first activity
- Sub-problem is \( S' = \{i \text{ in } S: s_i \geq f_1\} \)

\[ A' \] is an optimal solution for \( S' \)
\[ A' \cup \{1\} \] is an optimal solution for \( S \)
Optimal Substructure

Claim.
$A'$ is an optimal solution for $S' = \{ i \text{ in } S: s_i \geq f_1 \}$

$\iff$

$A = A' \cup \{1\}$ is an optimal solution for $S$

Proof. ($\Rightarrow$)
Let $A'$ be any optimal solution for $S'$. If $A' \cup \{1\}$ is not optimal for $S$, then (by greedy choice) there is a larger solution $B' \cup \{1\}$ for $S$. But then $B'$ is a solution for $S'$, and $B'$ has more activities than $A'$, contradicting the optimality of $A'$.

Proof. ($\Leftarrow$)
Let $A' \cup \{1\}$ be an optimal solution for $S$. If we could find a solution $B'$ to $S'$ with more activities than $A'$, adding activity 1 to $B'$ would yield a solution $B$ to $S$ with more activities than $A$ contradicting the optimality of $A$. 
Greedy-Choice + Opt substructure

Claim. Greedy is optimal for activity selection.

Proof. By induction on $|S|$. Base case. For $|S|=1$, greedy($\{(s_1,f_1)\}$) = $\{s_1,f_1\}$ = opt($\{(s_1,f_1)\}$).

Induction step. When $|S|>1$

$\text{greedy}(S)$

$= \{1\} \cup \text{greedy}(S')$ \quad - \text{definition of greedy}

$= \{1\} \cup \text{opt}(S')$ \quad - \text{induction on } |S|

$= \text{opt}(S)$ \quad - \text{optimal substructure}$