Optimization Problems:

- Knapsack problem.

A knapsack with capacity $B$ and $n$ elements $e_1, e_2, \ldots, e_n$ with volumes $v_1, v_2, \ldots, v_n$ and profits $p_1, p_2, \ldots, p_n$.

$$\max \sum x_i \cdot p_i, \quad x_i = \begin{cases} 1 & \text{if } e_i \text{ is chosen} \\ 0 & \text{otherwise} \end{cases}$$

s.t. $\sum x_i \cdot v_i \leq B$

$x_i \in \{0, 1\}$

SH case of a more general class of problems called Linear Programs.

Constraints in the form of linear inequalities:

$$\begin{cases} x_1 + x_3 + x_5 \leq 5 \\ 2x_2 + 4x_3 \leq 8 \end{cases}$$
\[ x_i \geq 0 \]

Maximize some linear function in \( x \)

\[
\text{Max } c_1 x_1 + c_2 x_2 + \ldots + c_n x_n
\]

Subject to \( \text{Max } c^T x \)

\[
\text{s.t. } A^T x \leq b
\]

\[ A : m \times n \text{ matrix} \]

Maximal Spanning Tree (MST)

Graph \( G = (V, E) \) with \( m \) edges \( w : E \to \mathbb{R}^+ \)

\[
\text{Max } \sum x_i w(e_i) \quad x_i \in \{0, 1\} \text{ if } e_i \text{ is chosen}
\]

Subject to \( \text{No cycles are induced by the edge chosen} \)

Maximum Matching (Assignment Problem)
Bipartite Graph

Choosing a subset of edges such that no vertex is incident to more than 1 edge is a feasible matching.

Max matching: \[ \text{Maximum Cardinality} \]

\[ \text{Maximum Weight} \quad \text{(weights on every edge)} \]

Write the appropriate math formulation.

Common Framework for Optimization

We have a ground set

\[ S : \{ e_1, e_2, e_3, \ldots, e_n \} \]

Weight: \( W : S \rightarrow \mathbb{R}^+ \)
The set of subsets \( 2^S \) consist of all possible subsets of \( S \). However, some of them are "feasible" and others are "infeasible".

The set of feasible subsets is often called "Independent."

**Goal**: Choose a subset \( T \in 2^S \) such that the weight \( w(T) \) is maximum:

\[
    w(T) = \max_{x \in T} w(x)
\]

Some common properties: \( \emptyset \) is independent.

- If \( S_1, S_2 \in J \) and \( S_2 \subset S_1 \), then \( S_2 \in J \), subset-property

Greedy: never backtracks.

Backtracking may try all possible subsets in \( J \).
Generic Greedy

Initialize \( T = \emptyset \)

Consider the set of elements in reverse sorted order of their weights \( \tilde{e}_1, \tilde{e}_2, \tilde{e}_3, \ldots, \tilde{e}_n \)

Repeat

Add the next object \( e_i \) to \( T \)

if \( T \cup \{e_i\} \) is feasible

until no more edges remain

(Kruskal's algorithm follows this)

Running time: 1) Pre-sorting of elements
2) For the iterative part, we repeatedly "test" if by adding the next object it remains feasible

Suppose the time is \( T_i \) for the \( i \)th step

\[ \sum_{i=1}^{n} T_i \]
How about "correctness"

Is the subset $T$ the best solution for all possible instances of the problem?

<table>
<thead>
<tr>
<th>Knapsack</th>
<th>$B = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_i$</td>
<td>$p_i$</td>
</tr>
<tr>
<td>$e_1$</td>
<td>10</td>
</tr>
<tr>
<td>$e_2$</td>
<td>5</td>
</tr>
<tr>
<td>$e_3$</td>
<td>5</td>
</tr>
</tbody>
</table>

**MST:** It works!

**Matching**

The problems for which Greedy succeeds, the underlying set system is called a "Matroid" (weights have no role in their framework).
Knapsock:

Consider the profit per volume as the metric for being greedy. Show that the final solve is at least 50% of the optimal.