Greedy Algorithms: General Ideas

- Basic Idea
Make decision that looks best now.

- Simple Example: Navigation

Greedy: always move closer to target

greedy fails

- Fractional Knapsack
  - Knapsack with capacity $m$
  - $n$ items, each item has weight $w_i$, value $v_i$
  - Items are "divisible" put a fraction of item into knapsack
  - Can put $\frac{1}{2}$ item 1 weight $w_1$, value $\frac{v_1}{2}$
  - Goal: put items into knapsack to get maximum value.

Ex: $m = 10$

$w_1 = 6 \quad v_1 = 20$
$w_2 = 5 \quad v_2 = 15$
$w_3 = 4 \quad v_3 = 10$

Solution: item 1 + $\frac{4}{5}$ item 2
total weight = 6 + \frac{4}{5} \cdot 5 = 10

total value = 20 + \frac{\leq}{5} \cdot 15 = 32

- Q: What is the first item I want to put in
  
  Idea: choose the largest \( V_i/W_i \)

- Algorithm
  
  sort items in decreasing order of \( V_i/W_i \)
  
  while knapsack is not full
  
  put the item with largest \( V_i/W_i \) remaining into knapsack

- Analysis: General Idea: Proof by Contradiction

1. Assume the optimal solution is different.
2. Show that we can improve optimal solution.

Proof: First, if two items have same ratio \( V_i/W_i \):

\[ V_1/W_1 = V_2/W_2, \text{ then can combine these items} \]

\( V_1 = 20, W_1 = 6 \quad V_2 = 10, W_2 = 3 \Rightarrow V = 30, W = 9 \)

Without loss of generality, \( V_1/W_1 > V_2/W_2 > V_3/W_3 > ... > V_n/W_n > 0 \)

\( \{ \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)

\( \)
Design an algorithm that can schedule as many meetings as possible.

\[(S_{i,j}) = (1,3), (2,4), (4,5), (4,6), (6,7)\]

Optimal: 3 meetings \((1,3) (4,5) (6,7)\)

\((2,4) (4,6) (6,7)\)

Q: What is the first meeting to schedule?

A: the one with earliest end time.

**Alg:** sort meetings in increasing order of end time

for \(i = 1 \text{ to } n\)

if meeting \(i\) can be scheduled

Put meeting \(i\) on Schedule.

\((1,3) (2,4) (4,5) (4,6) (6,7)\)

**Proof:** assume the meetings alg scheduled is

\[U_1, U_2, \ldots, U_k\]

\[(1,3) (4,5) (6,7)\]

Assume optimal solution is

\[V_1, V_2, \ldots, V_L\]

\(L > K\) \((\text{OPT} > \text{ALG})\)

(will show this is impossible)

(always think of \((U_1, \ldots, U_k), (V_1, \ldots, V_L)\) are sorted in time)

Claim: If \(i (1 \leq i \leq k)\) is the smallest index where \(U_i \neq V_i\)

\[U_1, U_2, \ldots, U_{i-1}, V_i, V_{i+1}, \ldots, V_L\]

are the same

by design of algorithm, \(U_i\) is the meeting with earliest end time, among meetings compatible with \(U_1, U_2, \ldots, U_{i-1}\)

in particular, end-time of \(U_i\) \(\leq\) end-time of \(V_i\)

\((U_1, U_2, \ldots, U_i, V_{i+1}, \ldots, V_L)\) is also a solution, because

start time of \(V_{i+1} \geq\) end time of \(V_i\)
Keep modifying optimal solution by Claim 1, then get an optimal solution where $u_i = v_i$ for all $i = 1, 2, \ldots, k$.

alg $u_1, u_2, \ldots, u_k$

opt $u_1, u_2, \ldots, u_k, v_{k+1}, \ldots, v_k$

by design of alg, if $\exists$ a meeting compatible with $u_1, u_2, \ldots, u_k$

it will be scheduled.

$v_{k+1}$ cannot exist.

This contradicts with our assumption. \(\square\)