Lecture 6: Greedy Algorithms

1. **Fractional Knapsack Problem**

   \( W = 10 \ (w_i, v_i) = (6, 20), (5, 15), (4, 10) \)

   - What are the decisions to make?
   - which item should I put into the Knapsack?
   - specify a rule for finding “best” item.
   - put in the item with max value per weight.
   - find \( i \) such that \( \frac{v_i}{w_i} \) is maximized.

   - first put in \((6, 20)\)
   - second, choose \((5, 15)\), put as large fraction as possible.

   \[ \text{solution} = 1 \times (6, 20) + 0.8 \times (5, 15) = (10, 32) \]

   - Proof of Correctness.
   
   idea: 1) assume there is a better solution (towards contradiction)
   
   2) prove the claimed “better” solution is not better.

   - Proof: without loss of generality
   
   assume items are sorted in decreasing order of \( \frac{v_i}{w_i} \).
   
   \[ \frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \frac{v_3}{w_3} \geq \cdots \geq \frac{v_n}{w_n} \]

   assume ALG gives a solution \((p_1, p_2, \ldots, p_n)\)

   \[ (1, 1, 1, 0.5, 0, 0, \ldots, 0) \]

   \[ (1, 0.8, 0) \]

   - assume (towards contradiction) that there is a better solution OPT.

   OPT has solution \((q_1, q_2, \ldots, q_n)\)

   (goal: show OPT is no better than ALG)

   Let \( i \) be the first location where \( p_i \neq q_i \)

   By design of the algorithm, we know \( p_i > q_i \)

   since OPT is assumed to be better, there must be
   item \( j \) (\( j > i \)) s.t. \( p_j < q_j \)

   (idea: remove small fraction of item \( j \) from OPT, use the capacity on item \( i \))

   if we remove \( \delta \) fraction of item \( j \) (get capacity \( \delta \cdot w_j \))

   use capacity on item \( i \)
\[ q_i' = q_i - \varepsilon \]
\[ q_i = q_i + \frac{\sum w_j}{w_i} \quad \text{OPT}' \]

Claim: New solution is as good as OPT

\[
\text{value (OPT')} = \text{value (OPT)} - \left( \sum \frac{v_j}{w_i} + \sum \frac{v_j - v_i}{w_i} \right) \\
\geq \text{value (OPT)} \left( \frac{v_i}{w_i} \geq \frac{v_j}{w_j} \right)
\]

OPT' is closer to ALG.

repeat this argument until this operation cannot be done.
eventually OPT becomes ALG, and each step can only increase the value. \[ \text{value (OPT)} \leq \text{value (ALG)} \]

Contradiction \( \square \)

- (Slightly) Simpler proof:
- merge all items with same ratio \( v_i/w_i \):
  - does not change solution because items are divisible.
  - does not change ALG because these items will be consecutive in the sorted list.
- assume wlog \( v_1/w_1 > v_2/w_2 > v_3/w_3 > \ldots > v_n/w_n \)
  - strictly larger because items with same ratio are merged.
- Suppose ALG's solution is \( (p_1, p_2, \ldots, p_n) \)
  - OPT's solution is \( (q_1, q_2, \ldots, q_n) \neq (p_1, p_2, \ldots, p_n) \)
  - let \( i \) be first item where \( p_i \neq q_i \),
    by design we know \( p_i > q_i \).
  - if value (OPT) > value (ALG), there must be an item \( j \) (\( j > i \))
    such that \( p_j < q_j \).
  - let \( OPT' \) be a solution \( (q_1', q_2', \ldots, q_n') \) \( q_t' = q_t \) for \( t \neq i,j \)
    \[ q_i' = q_i - \varepsilon \]
    \[ q_i = q_i + \frac{\sum w_j}{w_i} \]
    \[
    \text{value (OPT')} = \text{value (OPT)} - \sum \frac{v_j}{w_i} + \sum \frac{\varepsilon w_j}{w_i} \geq \text{value (OPT)} \quad \text{(because } \frac{v_i}{w_i} > \frac{v_j}{w_j})
    \]
    Contradiction, so OPT cannot be better than ALG \( \square \)

- Interval Scheduling
  - which is the first meeting to schedule?
    (earliest meeting)
- Intuition: earlier the better

\[ (1, 4), (2, 3), (2, 5) \]

Choose \((2, 3)\) over \((1, 4)\) because it ends earlier.

- **ALG:** always try to schedule the meeting with earliest ending time.

- **Proof of correctness:**
  - Assume **ALG** scheduled meetings \((i_1, i_2, \ldots, i_k)\)
  - Assume **OPT** has a better solution \((j_1, j_2, \ldots, j_t)\) \((t > k)\)
  - Both solutions are sorted in starting time \((s_{i_1} < s_{i_2} < \ldots < s_{i_k})\)
    \((s_{j_1} < s_{j_2} < \ldots < s_{j_t})\)
  - Let \(p\) be the first meeting where \(i_p \neq j_p\)
    - By design of algorithm
      - \(i_p\) ends before \(j_p\)
      - \(i_p\) ends before \(j_{p+1}\) start.
    
    Now \((i_1, i_2, \ldots, i_p, j_{p+1}, j_{p+2}, \ldots, j_t)\) is also a valid schedule.

**OPT'** is closer to **ALG**

Repeat this argument, there is an **OPT'** where \(i_p = j_p\) for all \(1 \leq p \leq k\)

\(i_1, i_2, \ldots, i_k\)

\(j_1, j_2, \ldots, j_{k+1}, \ldots, j_t\)

- That cannot happen by design of algorithm. **Contradiction**

- If you don't like the "repeat this argument" step, here is an alternative way to do it.

- **Proof (alternative):** all solutions are sorted in starting time
  - Let **ALG**'s solution be \((i_1, i_2, \ldots, i_k)\)
  - Assume (towards contradiction) that there is a better solution
    - Let \((j_1, j_2, \ldots, j_t)\) \((t > k)\) be an optimal solution that share the longest prefix with **ALG**
If \( i_p = j_p \) for all \( p \leq k \),
OPT scheduled \( j_{k+1} \) after \( j_k \)
ALG did not schedule \( j_{k+1} \)
this is impossible, because \( t_{j_{k+1}} > t_{j_k} \),
ALG tries to schedule \( j_{k+1} \) after \( j_k \), and should succeed.
So this is impossible.

Else let \( P \) be the first meeting that \( i_p \neq j_p \)
by design we know
\[
\tau_{i_p} \leq \tau_{j_p}
\]
since \( \tau_{j_p} \leq s_{j_{p+1}} \), we also have
\[
\tau_{i_p} \leq s_{j_{p+1}}
\]
therefore \((i_1, i_2, \ldots, i_p, j_{p+1}, \ldots, j_t)\) is also an optimal
solution, and it shares a longer prefix with ALG
this contradicts with the assumption.

Therefore OPT cannot be better than ALG \( \square \)