250H-Discussion: Algorithm Design Techniques

2/9/26

Brute Force

- Attempting all possible solutions, seeing which one is the “best”

- Guaranteed to give you the best possible solution

- Not always efficient
  - Exponential runtimes
  - P VS NP

- We get better solutions using design techniques
  - Narrowing down solution space (LPs)
  - Memoization
  - Heuristics

Greedy Algorithms

- An algorithm is greedy if it builds up a solution in small steps, each a “locally” optimum solution satisfying some criteria

- Although greedy algorithms are not optimal for all problems, they can produce optimal solutions for some
  - Typically very intuitive and easy to prove correctness
  - We won’t be going over the proof techniques

Example 1) Making Change

- Suppose you are a cashier and you need to make change for someone who just paid you
- They are annoying and ask for the minimum number of coins in return
- What approach do you take?

Taking the biggest coin

- Naturally, taking the coin with the most value lowers the remaining amount the most

Taking the biggest coin

- Naturally, taking the coin with the most value lowers the remaining amount the most

- This is the algorithm! take the largest possible coin available and subtract its value from the total, repeating until total is 0

Taking the biggest coin

Suppose the change you need to make is 87 cents

- Take as many quarters as you can
  - 1 – 62 remaining
  - 2 – 37 remaining
  - 3 – 12 remaining
  - Can’t take anymore
- Take as many dimes as you can
  - 1 – 2 remaining
  - Can’t take anymore
- Take as many nickels
  - Can’t take any!
- Take as many pennies
  - 2 pennies

Coin values

Vote: The algorithm we gave before

1. GIVES OPTIMAL SOLUTIONS FOR ALL COIN VALUES
2. DOESN"T GIVE OPTIMAL SOLUTIONS FOR ALL COIN VALUES

Coin values

- Not all coin values give you the optimal solution with this algorithm

Coin values

- Not all coin values give you the optimal solution with this algorithm
- {1, 3, 4,}

Coin values

- Not all coin values give you the optimal solution with this algorithm
- {1, 3, 4,}
- {1, 15, 25}

Coin values

- Not all coin values give you the optimal solution with this algorithm

- {1, 3, 4,} – 6
  - Greedy = {4, 1, 1}
  - Optimal = {3, 3}

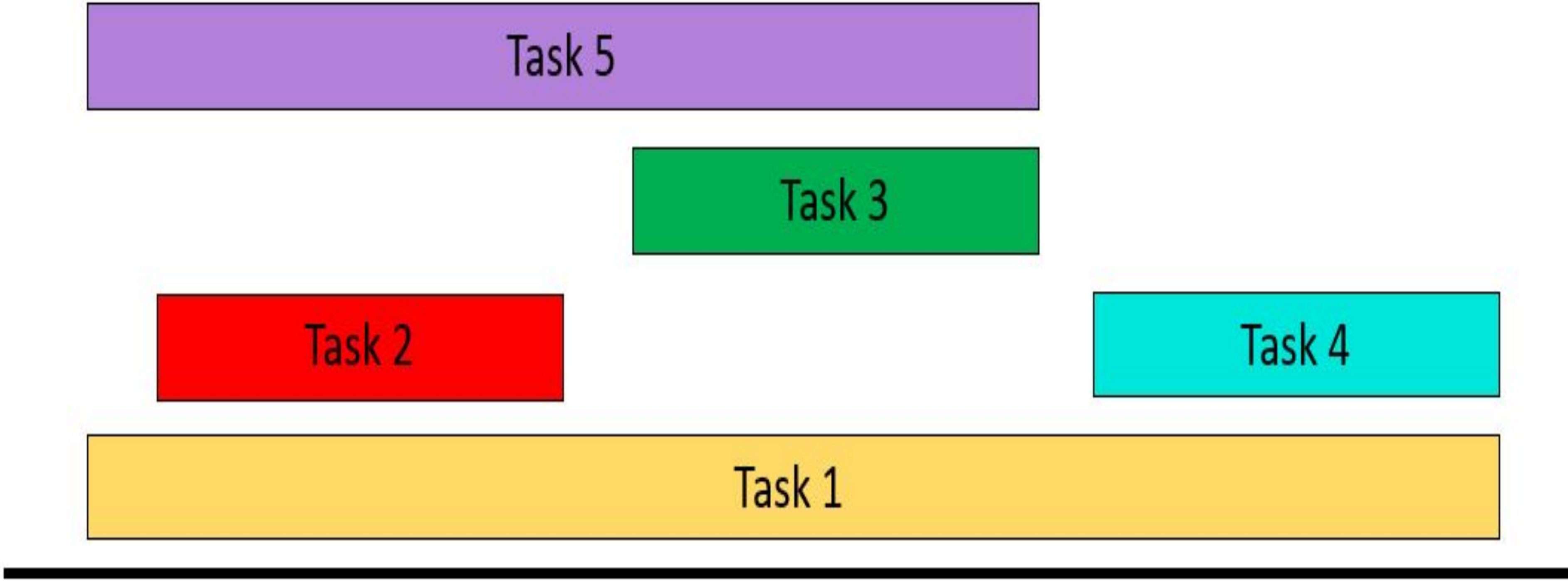
- {1, 15, 25} – 35
  - Greedy = {25, 1 (x10)}
  - Optimal = {15, 15, 1, 1, 1, 1, 1}

Brute Force?

What is a brute force algorithm for this?

Example 2) Interval Scheduling

- We have n requests, labeled 1, .., n, to use a room between times a-b. (i.e [a, b]).
- Each request has a starting time, S(i), and a finish time, F(i).
- We want to schedule as many of these as possible, without creating any overlap


Task 5
Task 3
Task 2
Task 4
Task 1

Answer:

We schedule Tasks 2, 3, and 4

Strategy/Heuristic

- How can we make a selection at each step based on the problem’s properties

- What are some ideas?
    - Select the one with the earliest start time?

Strategy/Heuristic

- How can we make a selection at each step based on the problem's properties

- What are some ideas?
  - Select the one with the earliest start time?
  - Select the one shortest in length?

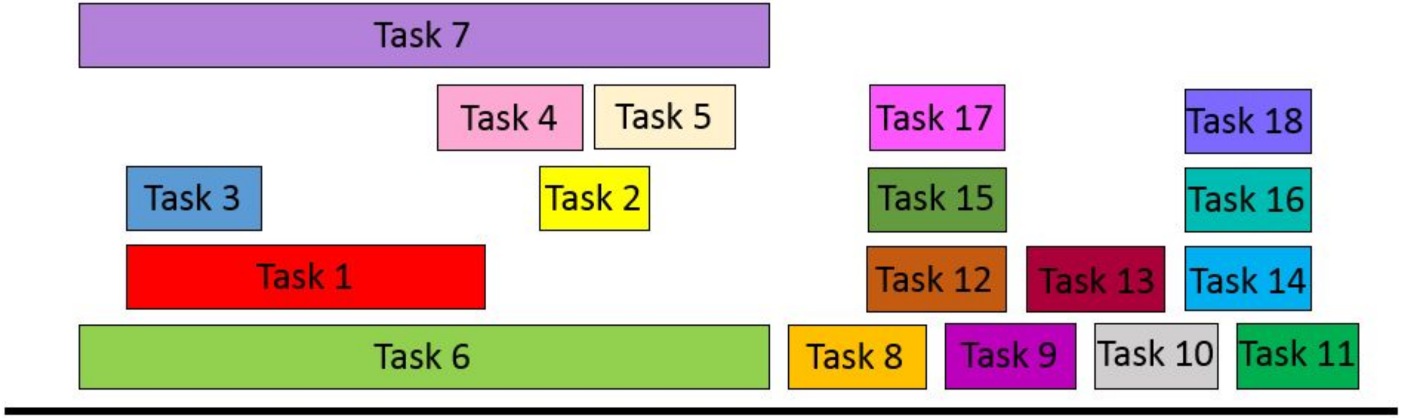
Strategy/Heuristic

- How can we make a selection at each step based on the problem's properties

- What are some ideas?
  - Select the one with the earliest start time?
    - Nope!
  - Select the one shortest in length?
    - Nope!
  - Select the one that finishes first?
    - Works!

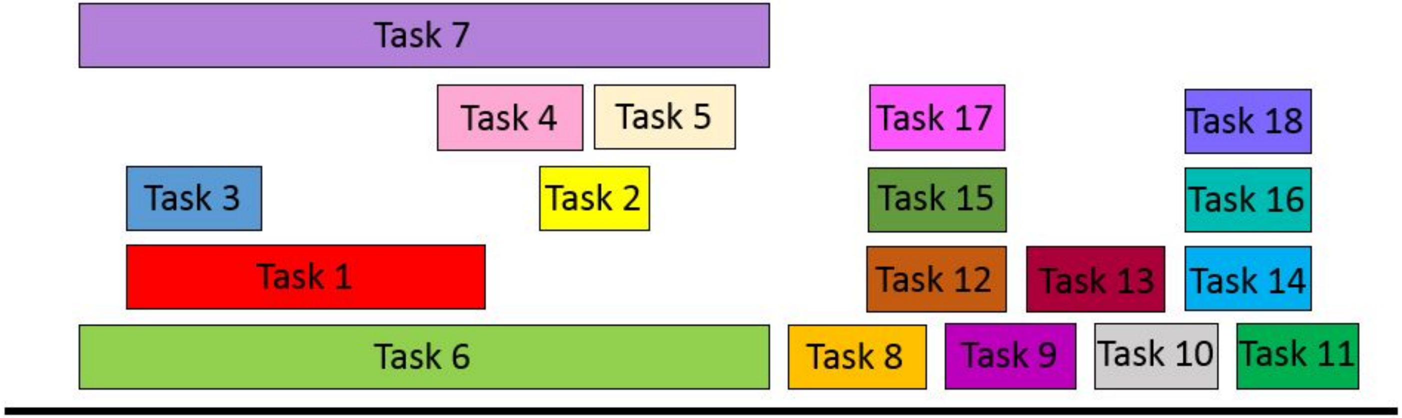
Sort by Finish Times

- Intuition:
    - This ensures that we have the most possible **REMAINING TIME** to schedule the rest of the problems



Sort by Finish Times

- Intuition:
  - This ensures that we have the most possible **REMAINING TIME** to schedule the rest of the problems
- Answer: Task 3, 4, 5, 8, 9, and 11

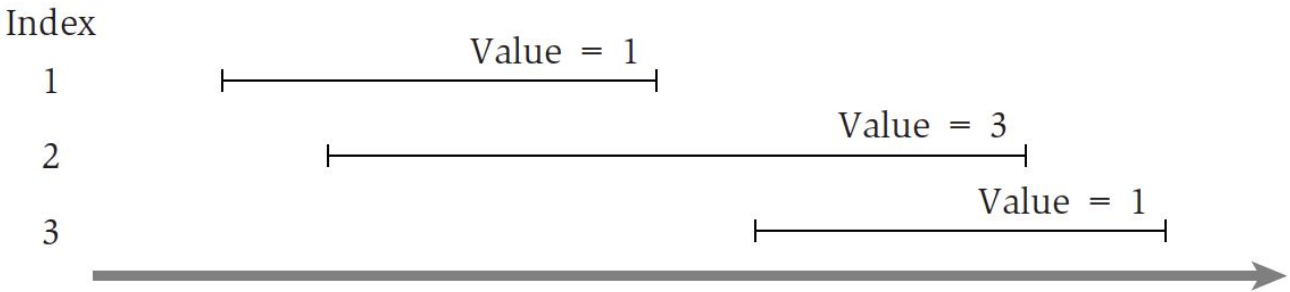


Expanding this Problem - Weighted I.S

- What if we assign weights to each of these requests
  - This represents some requests being more important than others
  - Each request now has a weight, w(i).
  - We want to maximize weight!
- Would our algorithm work now?

Expanding this Problem - Weighted I.S

- What if we assign weights to each of these requests
  - This represents some requests being more important than others
  - Each request now has a weight, w(i).
  - We want to maximize weight!
- Would our algorithm work now?
  - No



Proving Optimality of Greedy Algorithms

- Suppose we have access to an optimal solution, O.
- Our greedy solution: A

1. Assume A is not optimal
2. Compare it to O
3. Derive a contradiction by the properties of our algorithm

Or

1. Compare A and O at each step
2. Show that at each step A does **AT LEAST** as well as O