Data Structures and Algorithms

(ESO207)

Lecture 37

- A new algorithm design paradigm: Greedy strategy

part IV

Problems solved till now

1. **Job Scheduling Problem**

2. **Mobile Tower Problem**

3. **MST**

Did you notice anything commons in their solutions ?

Ponder over this question before moving ahead

Problem 1

Job scheduling Problem

We could not say anything about the **complete solution** of this problem.

INPUT:

- A set J of n jobs $\{j_1, j_2, \ldots, j_n\}$
- job j_i is specified by two real numbers

  s(i): start time of job j_i

  f(i): finish time of job j_i
- A single server

Constraints:

- Server can execute at most one job at any moment of time and a job.
- **Job** j_i, if scheduled, has to be scheduled during[s(i), f(i)] only.

Aim:

To select the **largest** subset of **non-overlapping** jobs which can be executed by the server.

Problem 1
Job scheduling Problem

INPUT:

- A set $\boldsymbol{J}$ of $\boldsymbol{n}$ jobs $\{\boldsymbol{j}_1, \boldsymbol{j}_2,\ldots, \boldsymbol{j}_n\}$
- job $\boldsymbol{j}_i$ is specified by two real numbers

  s($\boldsymbol{i}$): start time of job $\boldsymbol{j}_i$

  f($\boldsymbol{i}$): finish time of job $\boldsymbol{j}_i$
- A single server

Constraints:

- Server can execute at most one job at any moment of time and a job.
- **Job** $\boldsymbol{j}_i$, if scheduled, has to be scheduled during[**s**($\boldsymbol{i}$), **f**($\boldsymbol{i}$)] only.

Aim:

To select the **largest** subset of **non-overlapping** jobs which can be executed by the server.

All that we could do was to make a local observation

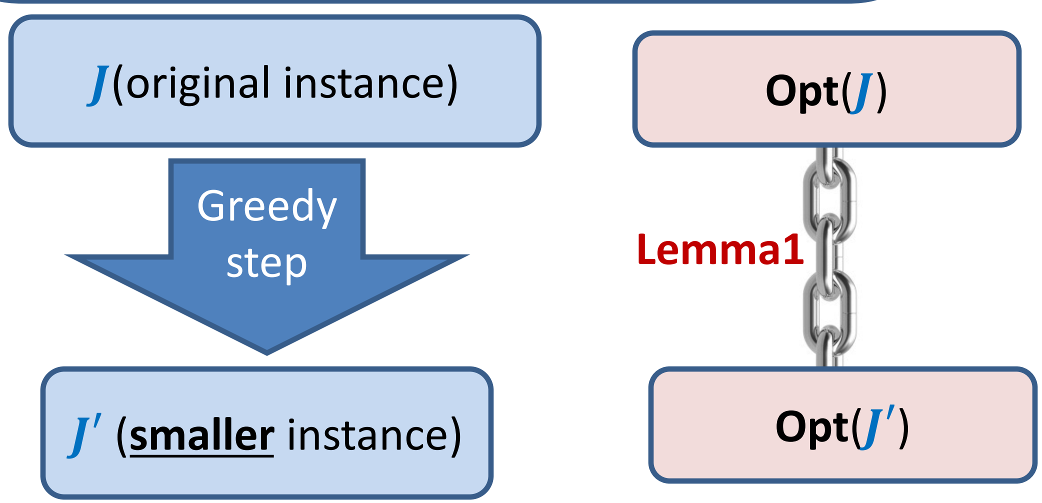
Let $x \in J$ be the job with earliest finish time.

Lemma1 : There exists an optimal solution for J in which x is present.

Let $J' = J \backslash$ **Overlap**(x)

Lemma 1 gives very small information about the optimal solution ☹
How to use it to compute this solution ?

Opt(J)= Opt(J') + 1 -- (i)



Equation **(i)** hints at recursive solution of the problem ☺

Theorem: $\mathbf{Opt}(J) = \mathbf{Opt}(J') + 1$.

- Proof has two parts

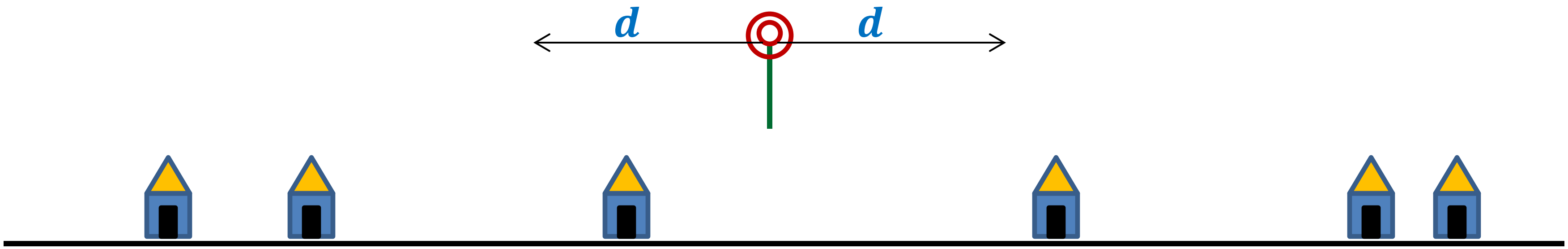
  $\mathbf{Opt}(J) \geq \mathbf{Opt}(J') + 1$

  $\mathbf{Opt}(J') \geq \mathbf{Opt}(J) - 1$

- Proof for each part is a proof by **construction**

Problem 2

Mobile Tower Problem



We could not say anything about the **complete solution** of this problem.

Problem statement:

There is a set H of n houses located along a roa

We want to place mobile towers such that

- Each house is **covered** by at least one mobile tower.
- The number of mobile towers used is **least** possible.

All that we could do was to make a local observation

Lemma 2: There is an optimal solution for the problem in which the leftmost tower is placed at distance d to the right of the first house.

Let x be the tower loc

Let H' = $H \setminus$ {all house

Lemma 1 gives very small information about the optimal solution ☹
How to use it to compute this solution ?

H(original instance)

Opt(H)

Opt(H)= Opt(H') + 1

Greedy step

Lemma 2

H'(**smaller** instance)

Opt(H')

Equation **(i)** hints at recursive solution of the problem ☺

What is a **greedy strategy** ?

A strategy that is

- Based on some **local** approach
- With the **objective to optimize** some function.

Note:

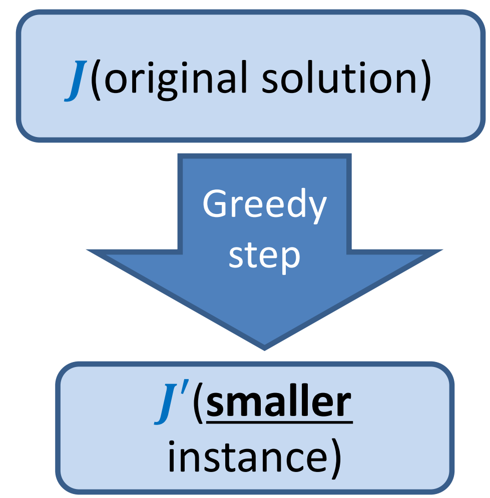
Recall that the divide and conquer strategy takes a **global approach**.

Design of a greedy algorithm

Let **A** be an instance of an optimization problem.

1. Make **a local observation** about the solution.

2. Use this observation to express optimal solution of **A** in terms of
   - Optimal solution of a smaller instance **A'**
   - Local step



3. This gives a recursive solution.
4. Transform it into iterative one.

MST

Input: an undirected graph G=(V,E) with w: $E \rightarrow \mathbb{R}$,

Aim: compute a **spanning tree** (V, E'), $E' \subseteq E$ such that $\sum_{e \in E'} \mathrm{w}(e)$ is **minimum.**

If you have understood a generic way to design a greedy algorithm, then try to solve the MST problem.

Lemma

If $e_0 \in E$ is the edge of **least weight** in G, then there is a **MST** T containing e_0.

How to use this **Lemma** to design an algorithm for **MST** ?

Problem 4
Overlapping Intervals

The aim of this problem is to make you realize that

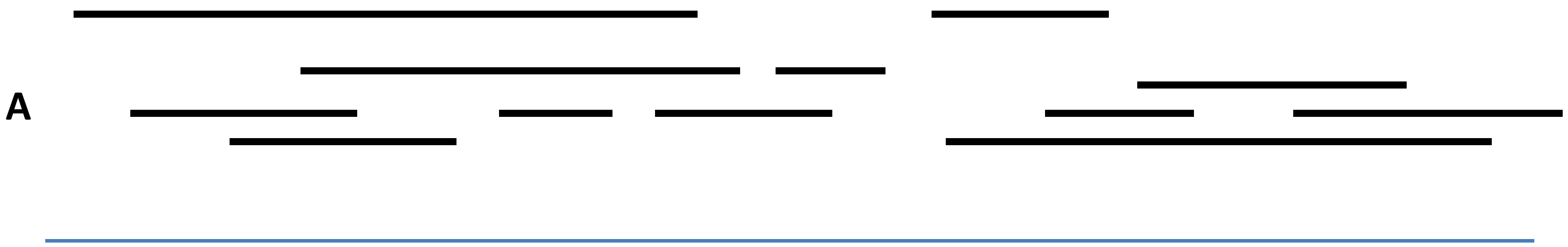
it is sometime very nontrivial to design a greedy algorithm. In particular, it is quite challenging to design the smaller instance.

Problem 4
Overlapping Intervals

Overlapping Intervals

Problem statement:

Given a set **A** of n intervals, compute smallest set **B** of intervals so that for every interval **I** in **A\B**, there is some interval in **B** which overlaps/intersects with **I**.



Overlapping Intervals

Problem statement:

Given a set **A** of n intervals, compute smallest set **B** of intervals so that for every interval **I** in **A\B**, there is some interval in **B** which overlaps/intersects with **I**.

A

another optimal solution ☺

Overlapping Intervals

Strategy 1

Interval with **maximum length** should be there in optimal solution

Intuition:

Selecting such an interval will **cover maximum** no. of other intervals

There is a counter example ☹

Overlapping Intervals

Strategy 1

Interval with **maximum length** should be there in optimal solution

Intuition:

Selecting such an interval will **cover maximum** no. of other intervals

There is a counter example ☹

Overlapping Intervals

Strategy 2

Interval that **overlaps maximum no.** of intervals should be there in optimal solution

Intuition:

Selecting such an interval will **cover maximum** no. of other intervals

There is a counter example ☹

Overlapping Intervals

Strategy 2

Interval that **overlaps maximum no.** of intervals should be there in optimal solution

Overlapping Intervals

Strategy 2

Interval that **overlaps maximum no.** of intervals should be there in optimal solution

Not an optimal solution ☹

Overlapping Intervals

Strategy 2

Interval that **overlaps maximum no.** of intervals should be there in optimal solution

An optimal solution has size 2.

Think for a while :
After failure of two strategies, how to proceed to design the algorithm.

Overlapping Intervals

Let **I*** be the interval with earliest finish time.

Let **I'** be the interval with **maximum** finish time overlapping **I***.

Lemma1: There is an optimal solution for set **A** that contains **I'.**

Proof:(sketch) :

If **I*** is overlapped by any other interval in the optimal solution, say **I^**,

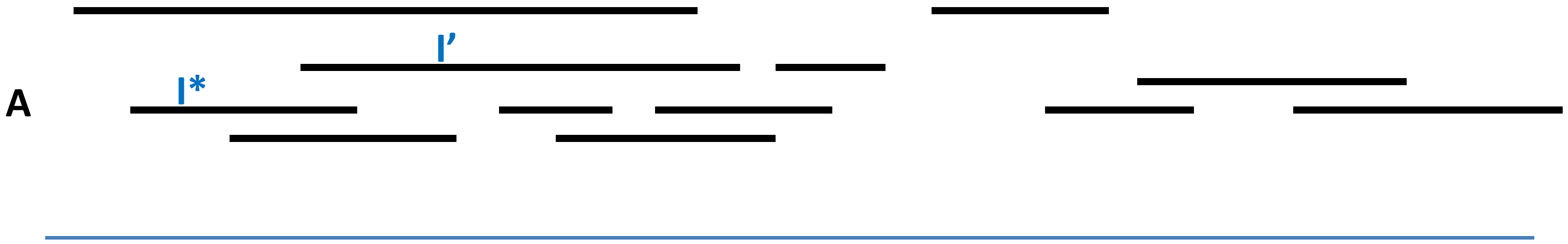
I' will surely overlap all intervals that are overlapped by **I^**.

➔ Swapping **I^** by **I'** will still give an optimal solution.

Exploit the fact that **I*** has earliest finish time for this claim.

Overlapping Intervals

Question: How to obtain smaller instance **A'** using **Lemma 1** ?

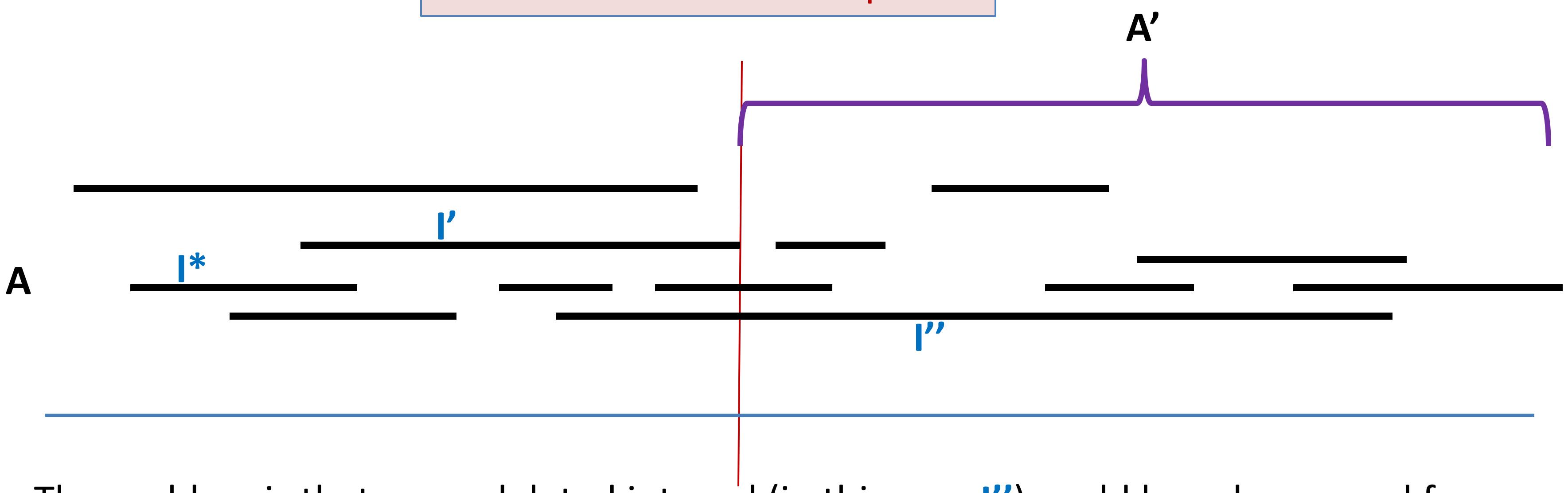


Overlapping Intervals

Question: How to obtain smaller instance **A'** using **Lemma 1** ?

Naive approach : remove from **A** all intervals which overlap with **I'**. This is **A'**.

There is a counter example ☹



The problem is that some deleted interval (in this case **I''**) could have been used for intersecting many intervals if it were not deleted. But deleting it from the instance disallows it to be selected in the solution.

Homework

- How will you form the smaller instance ?
- Design an algorithm for the problem.
- Give a neat, concise, and formal proof of correctness of the algorithm.