Data Structures and Algorithms

(CS210A)

Lecture 35

- A new algorithm design paradigm: Greedy strategy

part II

Continuing Problem from last class

JOB Scheduling

Largest subset of non-overlapping job

A job scheduling problem

Formal Description

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:

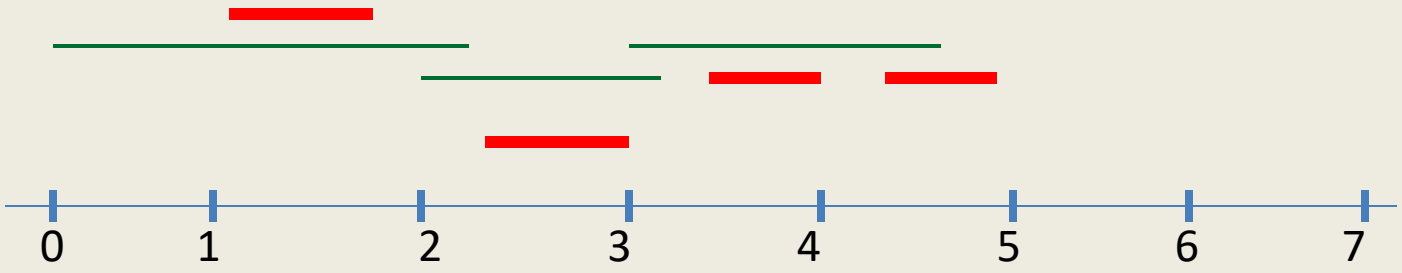
- Server can execute at most one job at any moment of time and a job.
- **Job** j_i, if scheduled, has to be scheduled during[**s**(i), **f**(i)] only.

Aim:

To select the **largest** subset

Designing algorithm for the problem

Strategy 4: Select the job with earliest finish time



Intuition:

Selecting such a job will **free** the server **earliest**

➔ hence more no. of jobs might get scheduled.

Algorithm "earliest finish time"

Algorithm (**Input :** set J of n jobs.)

1. Define $A \leftarrow \emptyset$;
2. **While** $J <> \emptyset$ **do**

   { Let $x \in J$ has earliest finish time;

   $A \leftarrow A \cup \{x\}$;

   $J \leftarrow J \backslash \textbf{Overlap}(x)$;

   }
3. Return A;

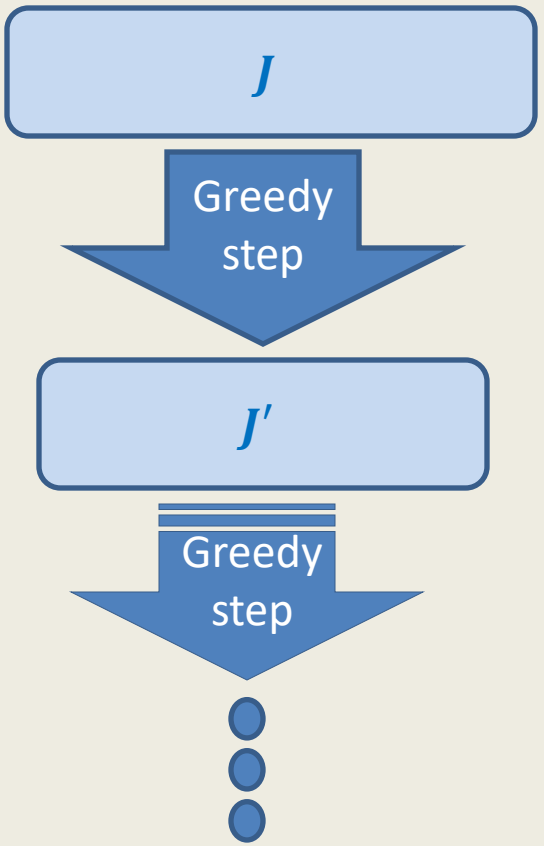
Lemma1 (last class):

There exists an optimal solution for J containing the **earliest finish time** job.

Proof of correctness ?

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

Let $J' = J \backslash \textbf{Overlap}(x)$



Algorithm "earliest finish time"

How to prove it ?

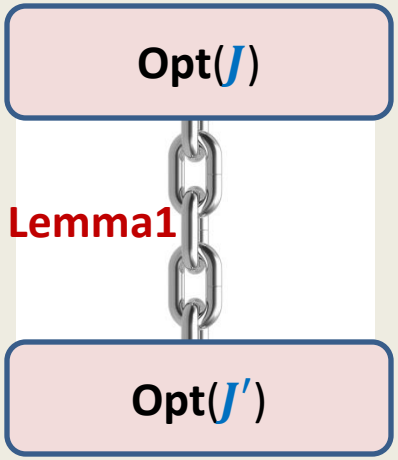
proof correct

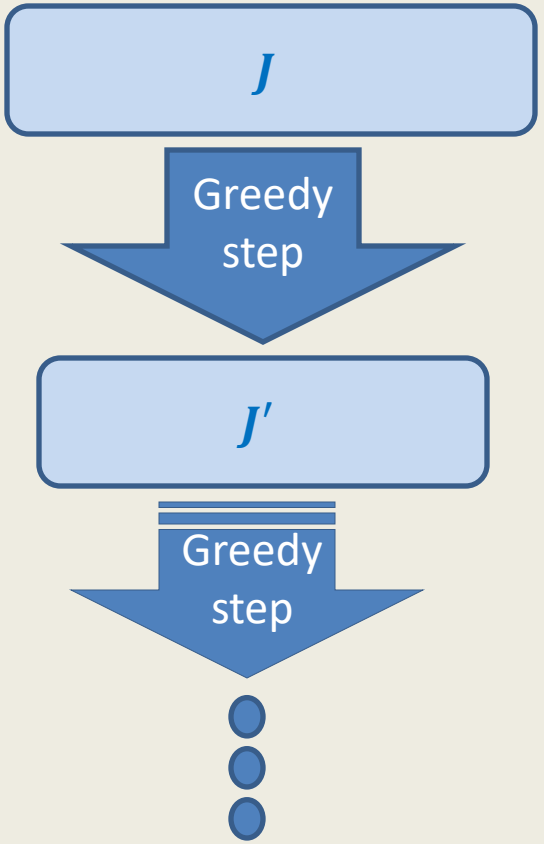
Proof of correctness ?

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

Let $J' = J \backslash \textbf{Overlap}(x)$

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





Notation:

Opt(J):

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

- Proof has two parts

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

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

  Try to give a physical interpretation to these inequalities.

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

Algorithm "earliest finish time"

Proving Opt(J) ≥ Opt(J') + 1

Observation: start time of every job in J' is greater than finish time of x.

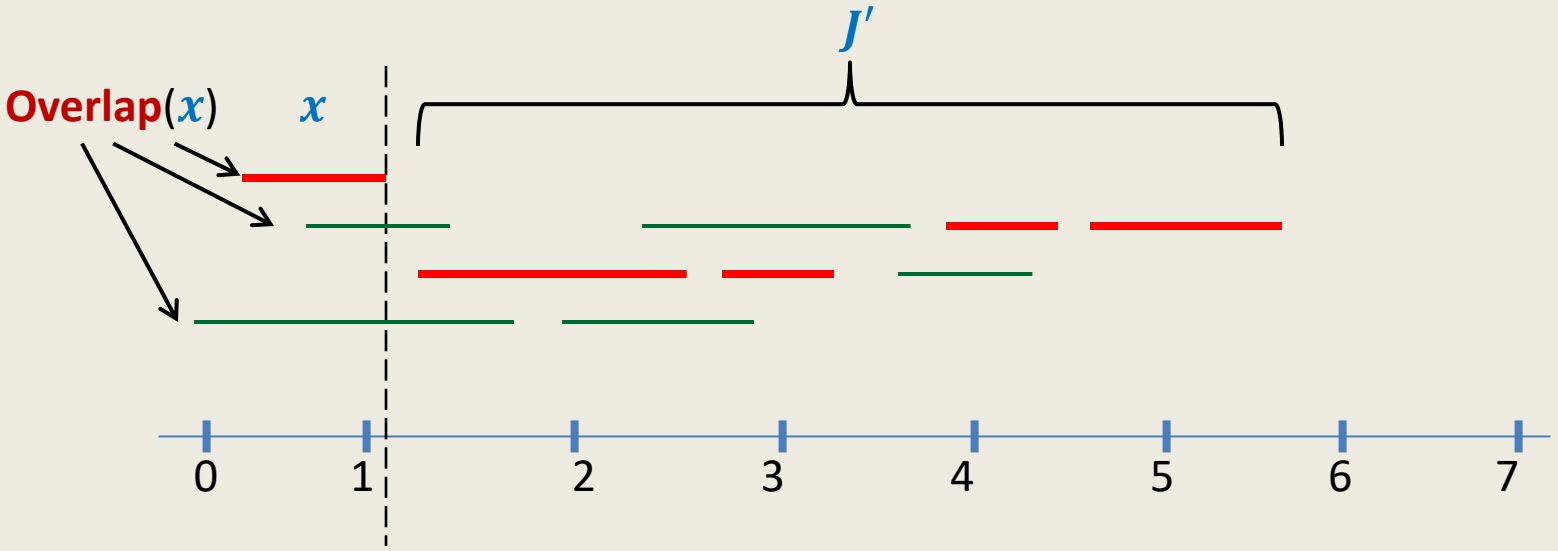
Let O' be any optimal solution for J'.

None of the jobs in

From an **optimal solution** of J'

Hence O' U$\{x\}$ is a

Therefore **Opt**(J) ≥ $|O'|$ + 1



Algorithm "earliest finish time"

Lemma1 (last class): There exists an optimal solution for J in which x is present.

Let O be an optimal solution for J containing x.

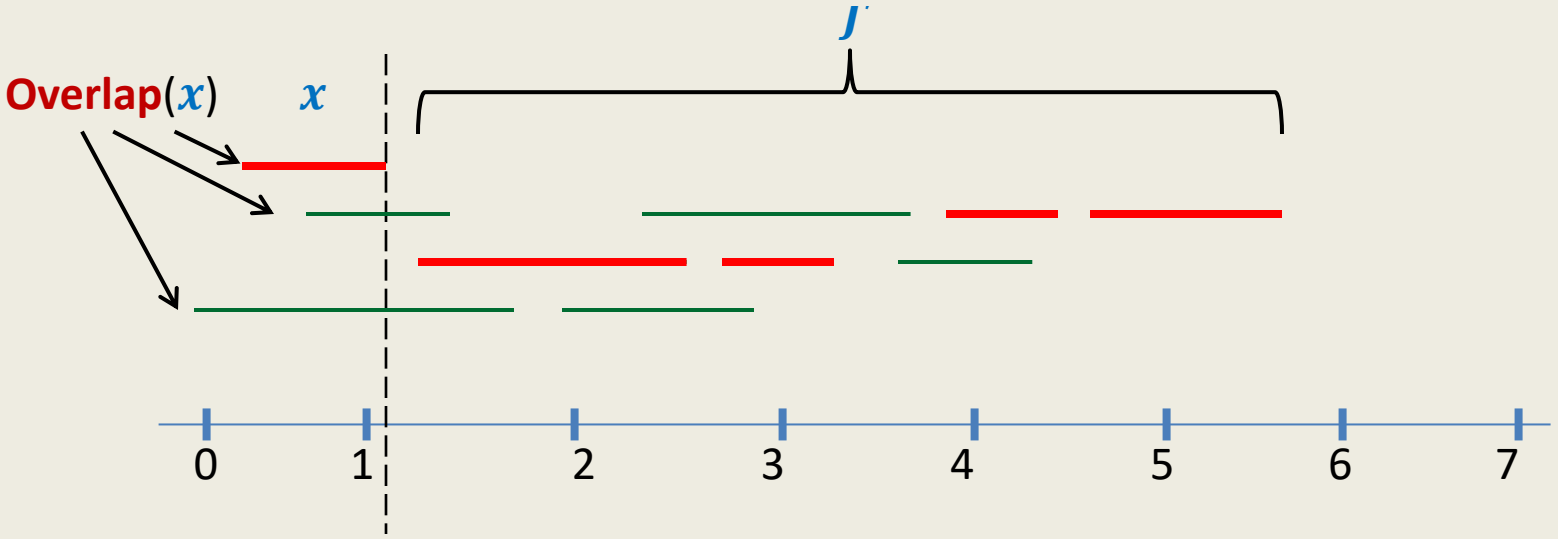
None of the jobs in O overlaps with x.

➔ Every job from O

Hence $O\backslash\{x\}$ is a subset of non-overlapping jobs from J'.

Therefore **Opt**(J') ≥ $|O|$ - 1

From an **optimal solution** of J
can you derive a **solution** for J' with one job less?



Theorem:

Given any set J of n jobs,

the algorithm based on "**earliest finish time**" approach

computes the largest subset of non-overlapping job.

O(n log n) implementation of the Algorithm

This is not the only way to achieve O(n log n) time. Many students gave multiple solutions (even simpler than this) in the class. Kudos to them! ☺

Algorithm (**Input :** set J of n jobs.)

1. Define $A \leftarrow \emptyset$;
2. **While** $J <> \emptyset$ **do**
   {
   Let $x \in J$ have earliest finish time;
   $A \leftarrow A \cup \{x\}$;
   $J \leftarrow J \backslash \text{Overlap}(x)$;
   }
3. Return A;

Maintain a **binary min-heap** for J based on **finish time** as the key.

Sort J in increasing order of **start time**.

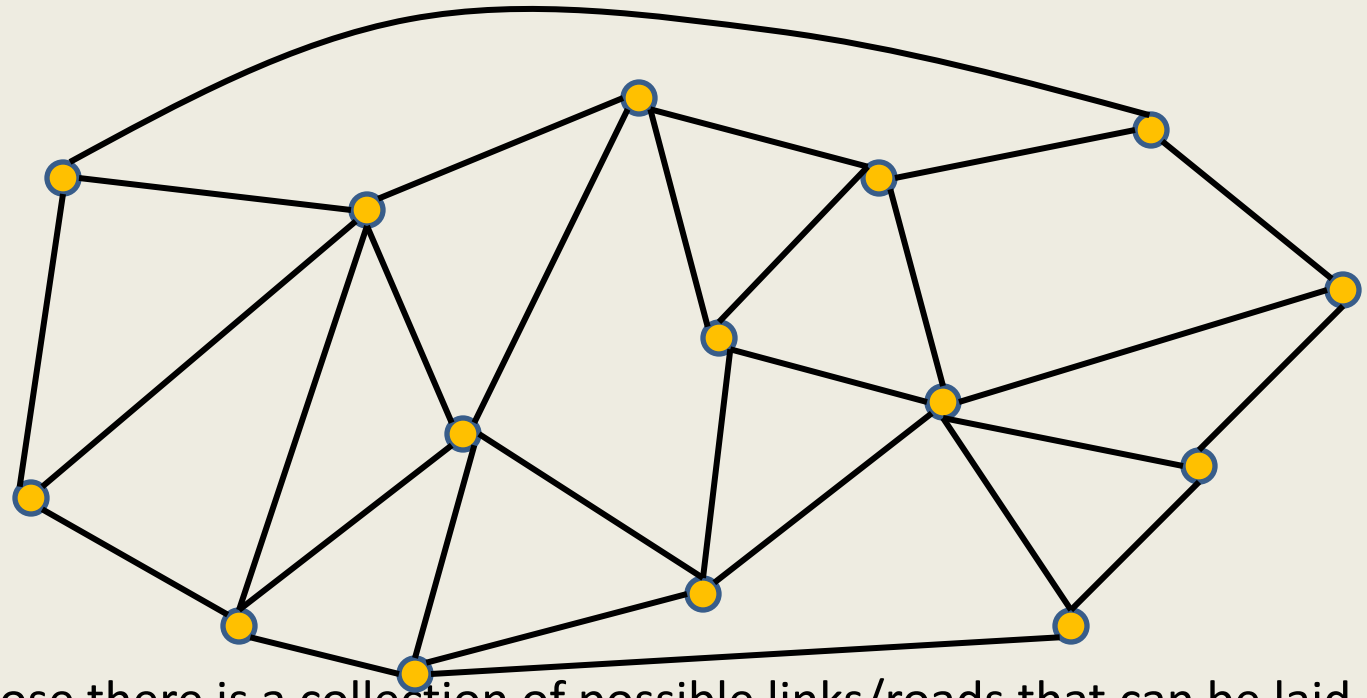
➔ **O(n^2) time complexity is obvious**

Problem 2

First we shall give motivation.

Motivation:

A road or telecommunication network



Suppose there is a collection of possible links/roads that can be laid.

But laying down each possible link/road is costly.

Aim: To lay down **least number** of links/roads to ensure **connectivity** between each pair of nodes/cities.

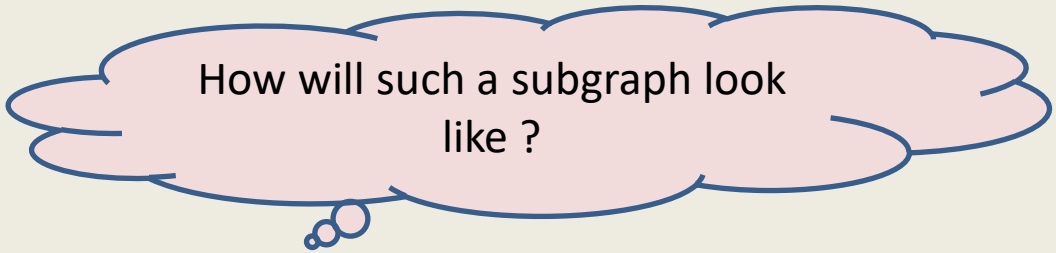
Motivation

Formal description of the problem

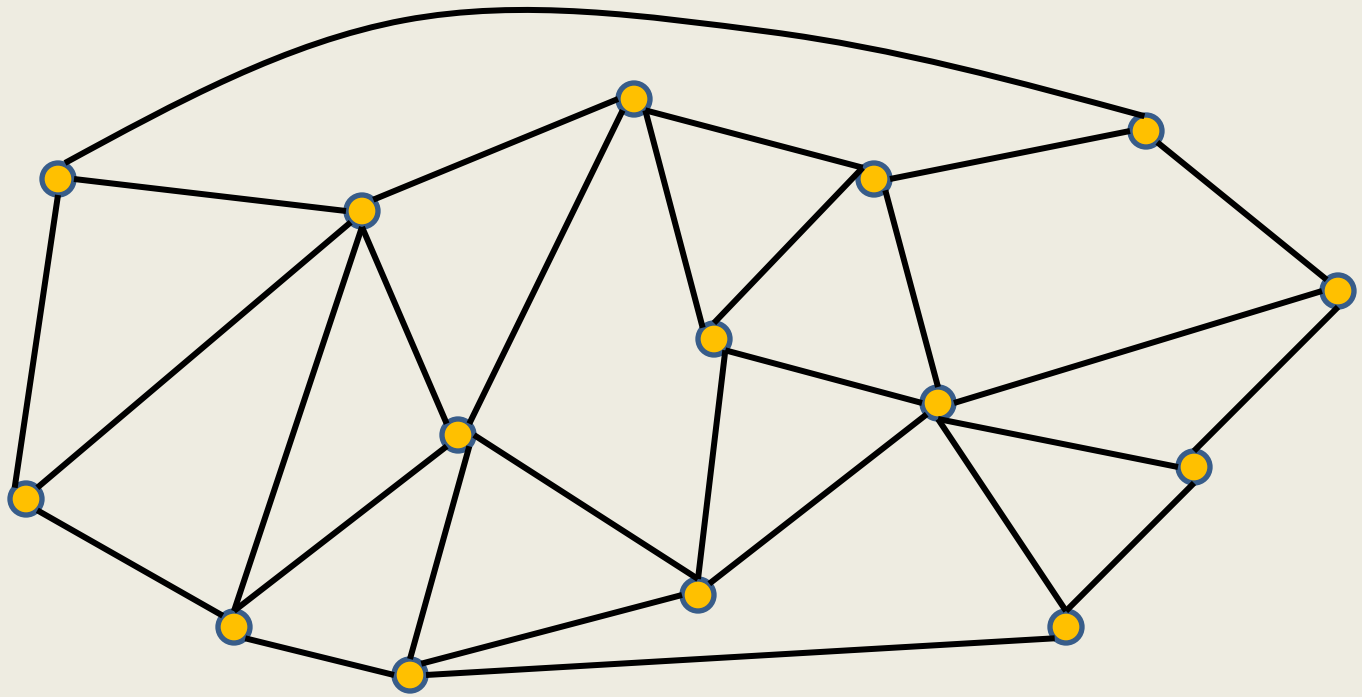
Input: an undirected graph **G**=(**V**,**E**).

Aim: compute a **subgraph** (**V**,**E'**), **E' ⊆ E** such that

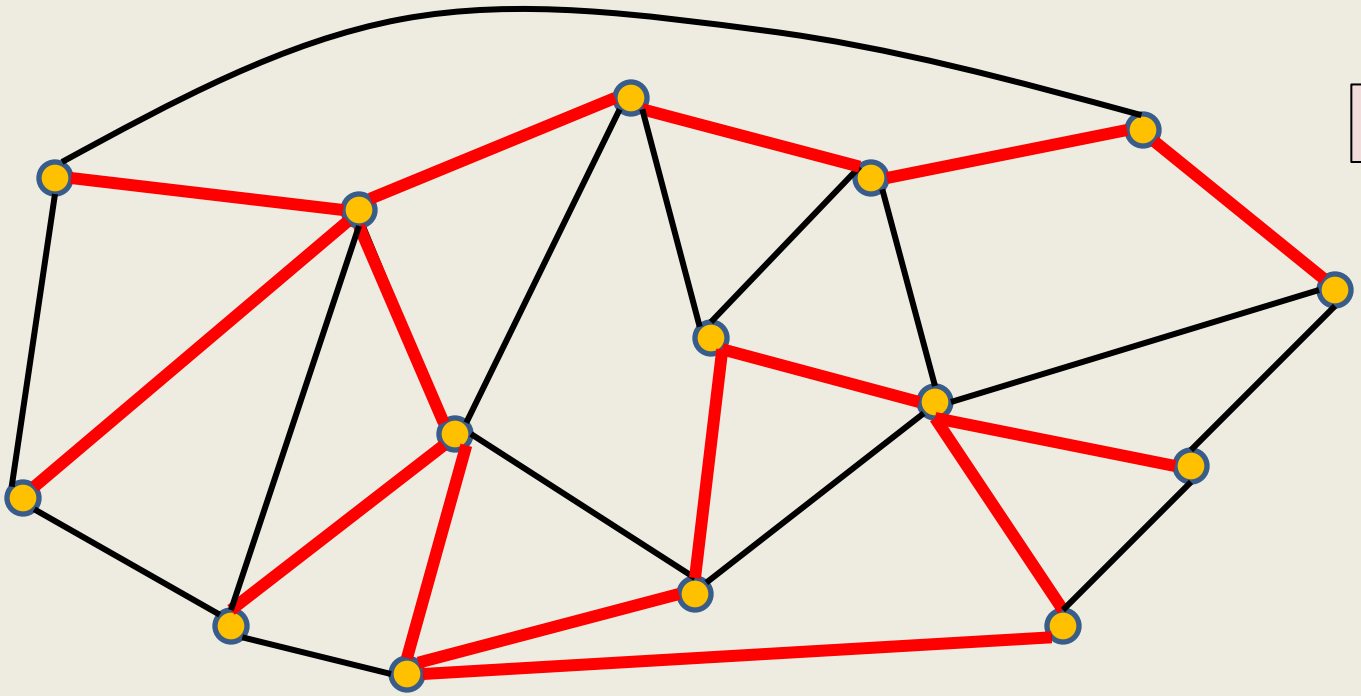
- **Connectivity** among all **V** is guaranteed in the **subgraph**.
- **|E'|** is **minimum.**



A road or telecommunication network



A road or telecommunication network



No | Yes | No

Is this subgraph meeting our requirement ?

A tree

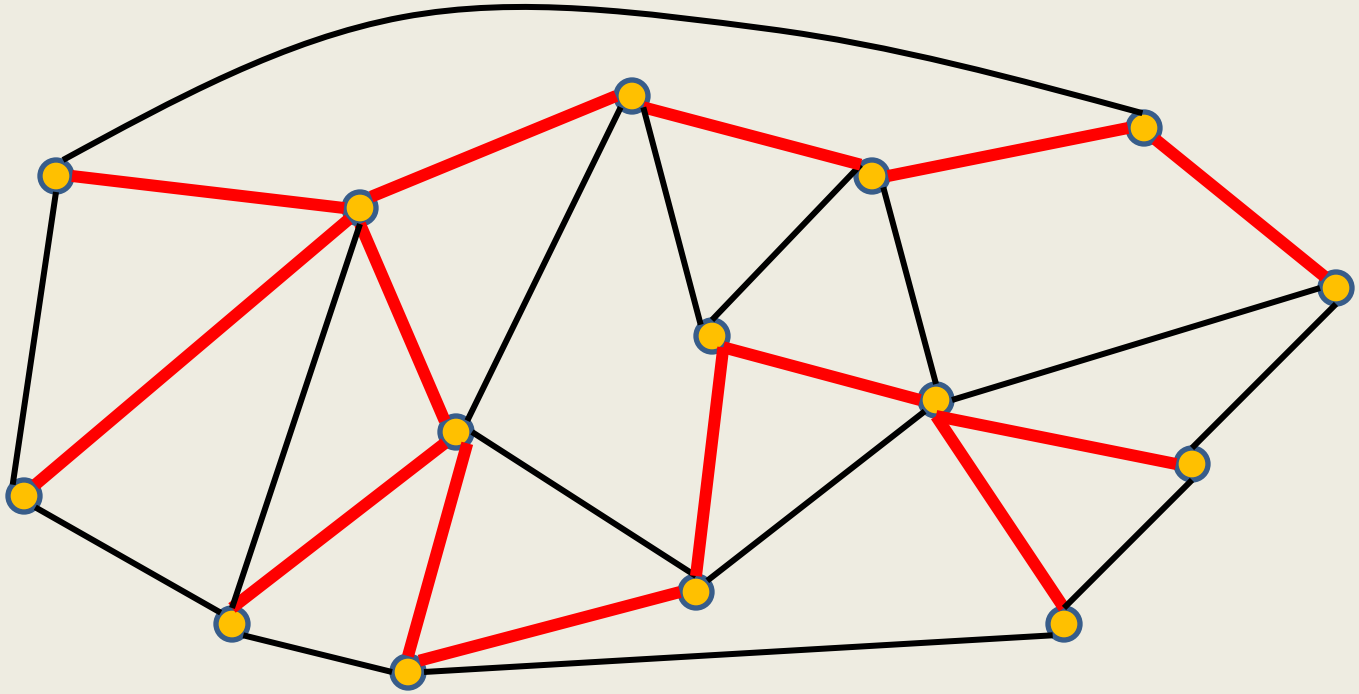
The following definitions are **equivalent**.

- An undirected graph which is **connected**
- An undirected graph where each pair of vertices has
- An undirected **connected** graph on n vertices and
- An undirected graph on n vertices and $n-1$ **edges** and

A Spanning tree

Definition: For an undirected graph (V,E),

A spanning tree is a **subgraph** (V,E'), E' ⊆ E which is a tree.

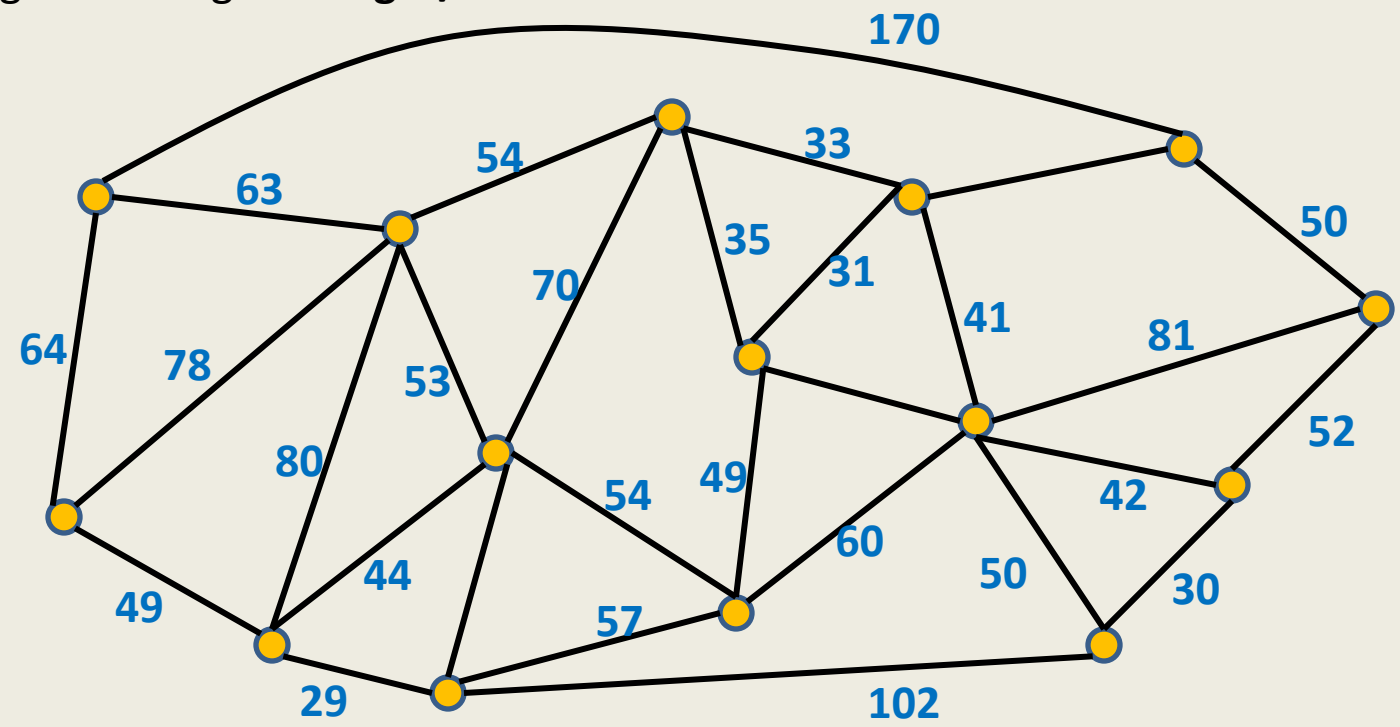


Observation: Given a spanning tree T of a graph G, adding a nontree edge e to T creates a unique cycle.

There will be total $m - n + 1$ such cycles. These are called **fundamental cycles** in G induced by the spanning tree T.

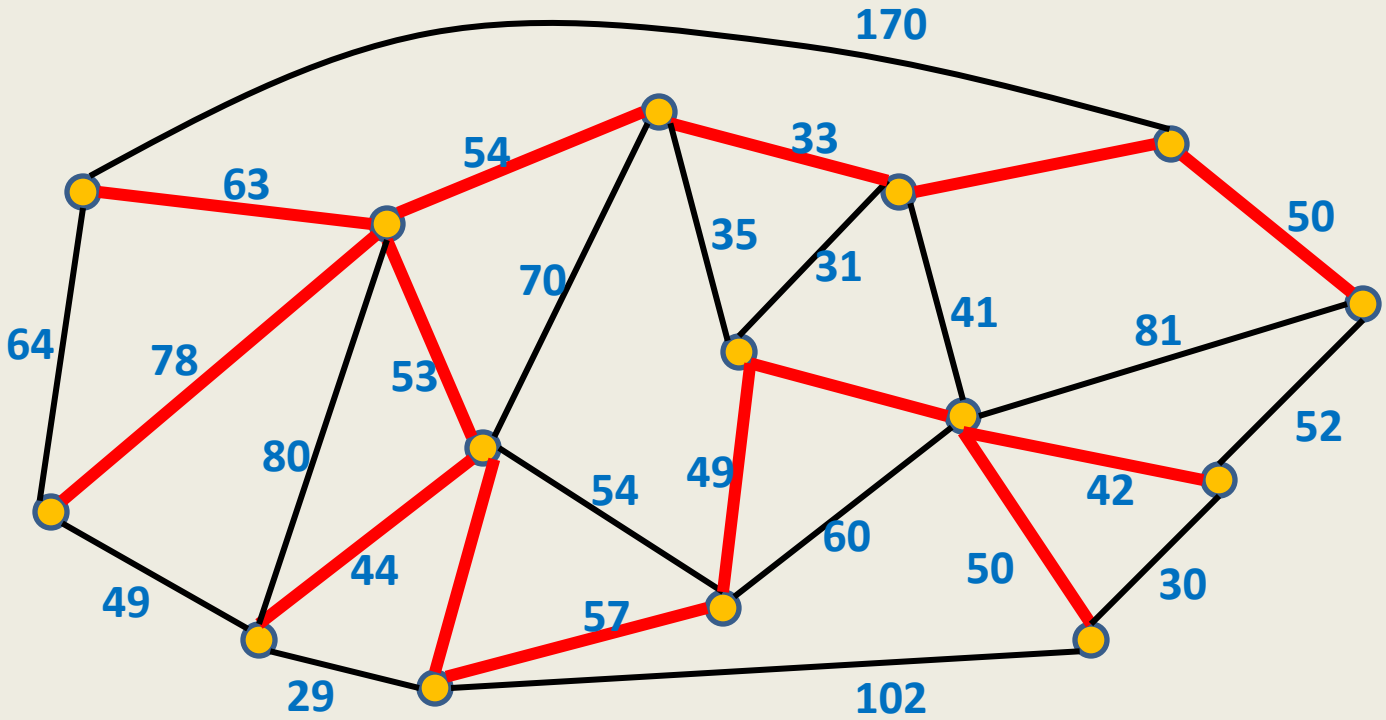
A road or telecommunication network

Assign each edge a **weight/cost**.



Adding more reality to the problem

A road or telecommunication network



Any arbitrary spanning tree (like the one shown above) will not serve our goal☹.

We need to select the spanning tree with **least weight/cost**.

Problem 2

Minimum spanning tree

Problem Description

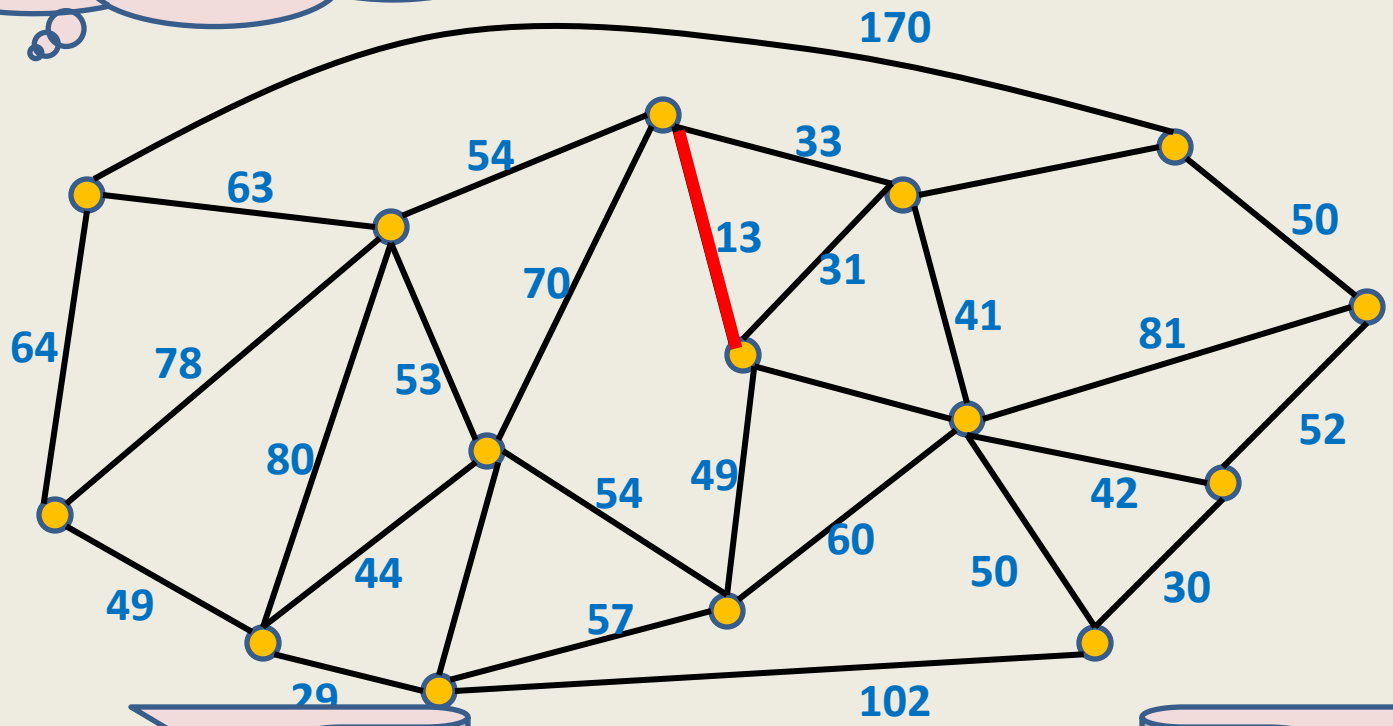
Input: an undirected graph **G**=(**V**,**E**) with **w**: **E** → $\mathbb{R}$,

Aim: compute a **spanning tree** (**V**,**E'**), **E'** ⊆ **E** such that

$\sum_{e \in E'} w(e)$ is **minimum**.

How to compute a MST ?

The least weight edge should be in MST.
But why ?



Look a nd.
Ca

Is there any edge for which you feel strongly to be present in MST ?

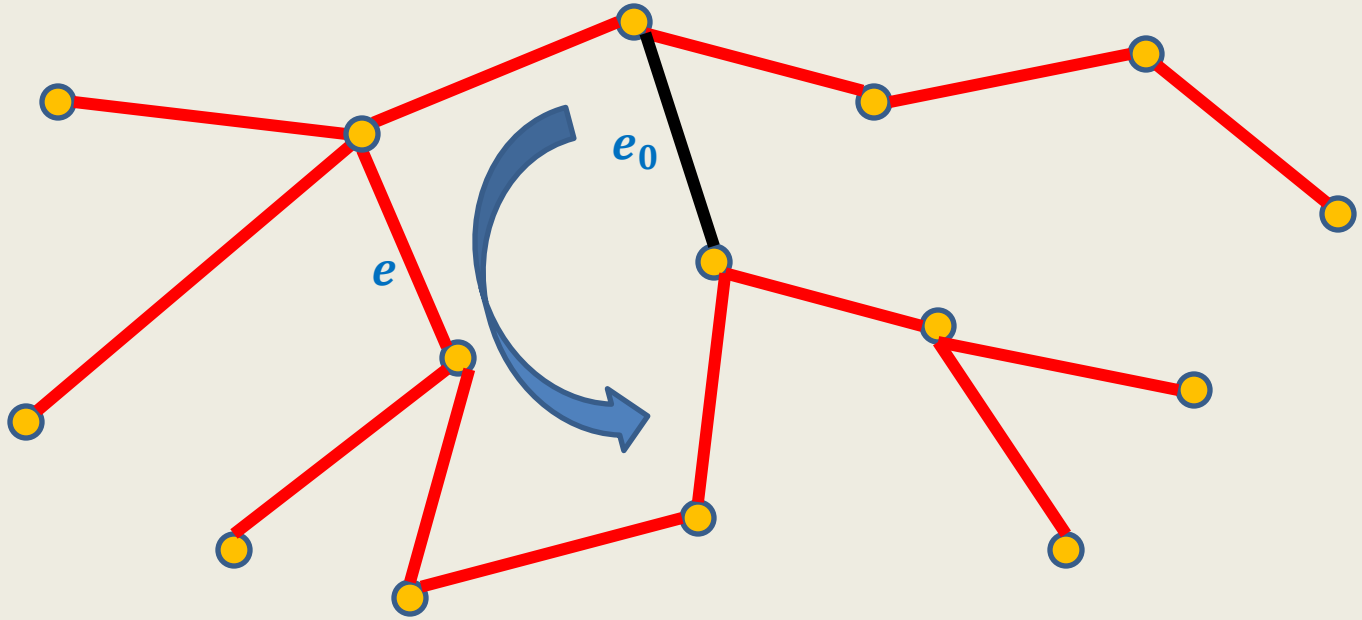
Let $e_0 \in E$ be the edge of least weight in the given graph.

Lemma2: There is a **MST** T containing e_0.

Proof: Consider any **MST** T. Let $e_0 \notin T$.

Consider the fundamental cycle C defined by e_0 in T.

Swap e_0 with any edge $e \in T$ present in C.



Let $e_0 \in E$ be the edge of least weight in the given graph.

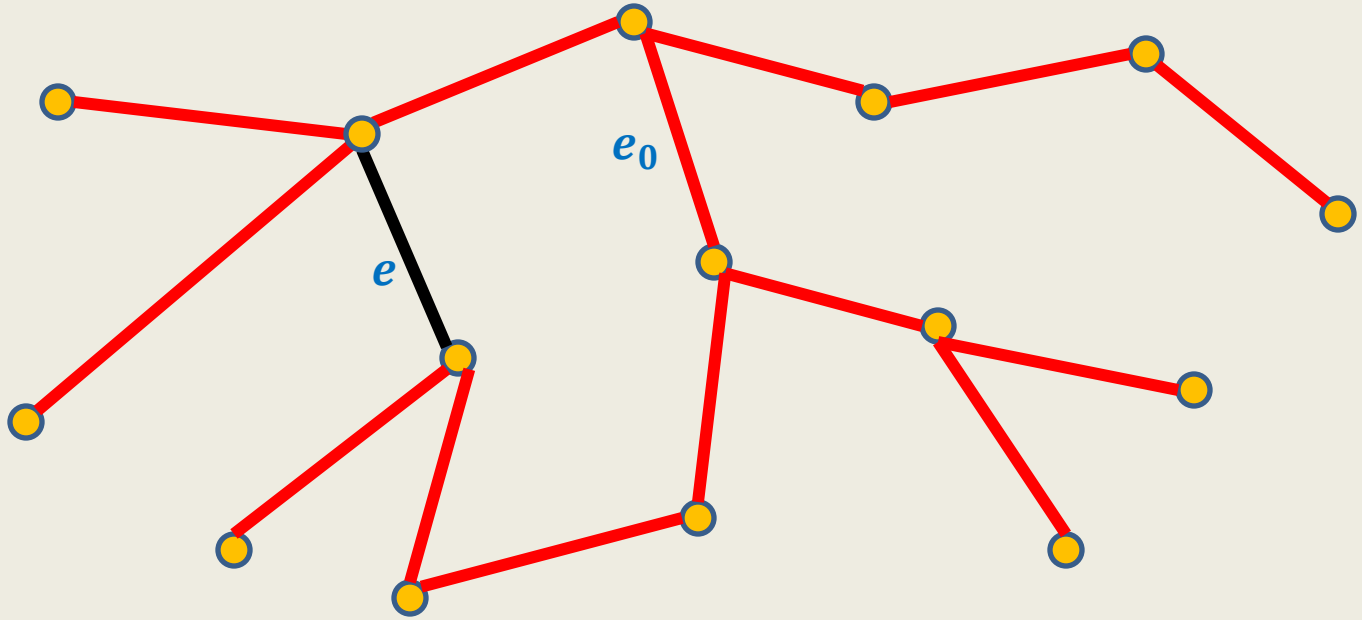
Lemma2: There is a **MST** T containing e_0.

Proof: Consider any **MST** T. Let $e_0 \notin T$.

Consider the fundamental cycle C defined by e_0 in T.

Swap e_0 with any edge $e \in T$ present in C.

We get a spanning tree of weight ≤ $w(T)$.



Try to translate Lemma2 to an algorithm for MST ?

with **inspiration** from the job scheduling problem ☺

Next lecture class: Sunday (3rd April) at 11 AM…