INT201 Decision, Computation and Language

Lecture 10 – Variants of Turing Machine

Dr Yushi Li

Xi'an Jiaotong-Liverpool University
西交利物浦大学

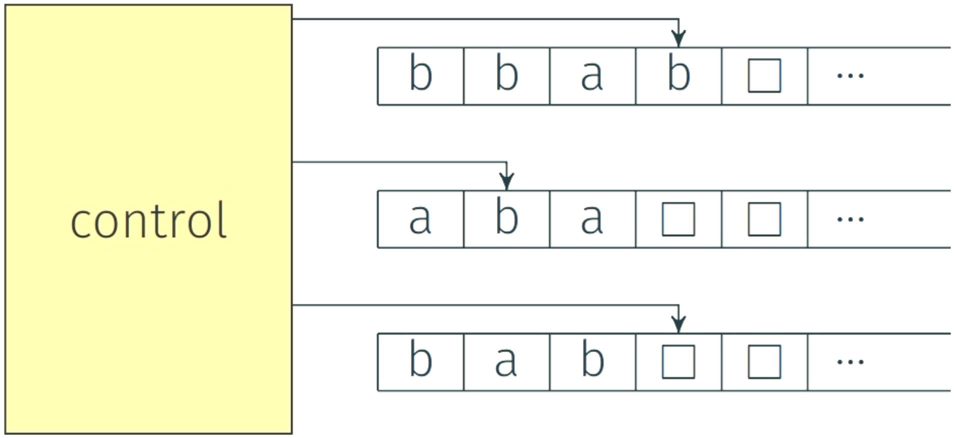
Multi-tape TM

Multi-tape TM has multiple tapes

- Each tape has its own head
- Transition determined by

  (1) state

  (2) the content read by all heads
- Reading and writing of each head are independent of others



Multi-tape TM

Definition

A k-tape Turing machine (TM) is a 7-tuple $M = (\Sigma, \Gamma, Q, \delta, q, q_{accept}, q_{reject})$ has k different tapes and k different read/write heads, where

- Σ is a finite set, called the input alphabet; the blank symbol $\sqcup$ is not contained in Σ,
- Γ is a finite set, called the tape alphabet; this alphabet contains the blank symbol $\sqcup$, and $\Sigma \subseteq \Gamma$,
- Q is a finite set, whose elements are called states,
- q is an element of Q; it is called the start state,
- q_{accept} is an element of Q; it is called the accept state,
- q_{reject} is an element of Q; it is called the reject state
- δ is called the transition function, which is a function $\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R, N\}^k$.
  $\Gamma^k = \Gamma \times \Gamma \times \ldots \times \Gamma$

Multi-tape TM

Transition

Transition function:

$$\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R, N\}^k$$

Given $\delta(q_i, a_1, a_2,\ldots, a_k)=(q_j, b_1, b_2,\ldots, b_k, L, R, \ldots, L)$

- TM is in state q_i
- heads $1\text{-}k$ read $a_1, a_2,\ldots, a_k$

Then

- TM moves to q_j
- heads $1\text{-}k$ write $b_1, b_2,\ldots, b_k$
- Heads move (left or right) or don't move as specified (L, R, N).

Multi-tape TM

Example

- Multiple tapes are convenient
- Some tapes can serve as temporary storage

Multi-tape TM equivalent to 1-tape TM

Let $k \geq 1$ be an integer. Any k-tape Turing machine can be converted to an equivalent 1-tape Turing machine.

For every multi-tape TM M, there is a single-tape TM M' such that $L(M) = L(M')$.

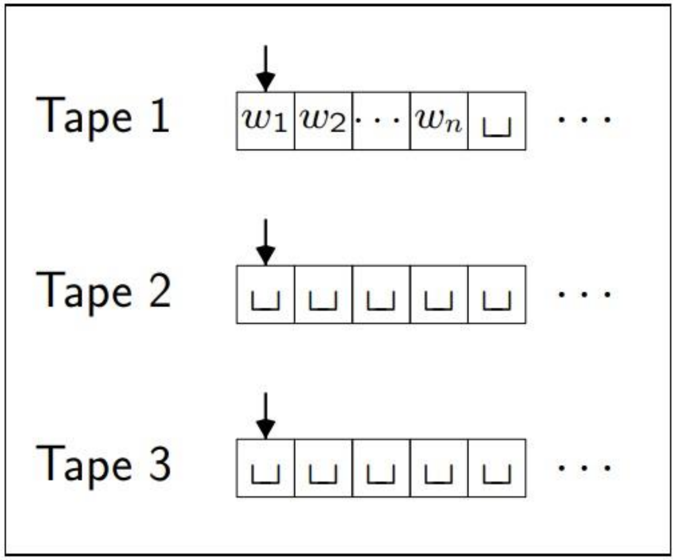
Proof

Basic idea: simulate k-tape TM using 1-tape TM.

Multi-tape TM equivalent to 1-tape TM

Proof

Let TM $M = (\Sigma, \Gamma, Q, \delta, q, q_{accept}, q_{reject})$ be a k-tape TM.

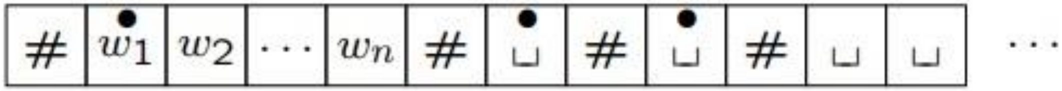


M has:

- input $w = w_1, w_2, \ldots, w_k$ on tape 1
- other tapes contain only blanks $\sqcup$
- each head points to first cell.

Construct 1-tape TM M' by extending tape alphabet

$$\Gamma' = \Gamma \cup \dot{\Gamma} \cup \{\#\}$$



Note: head positions of different tapes are marked by dotted symbol

Multi-tape TM equivalent to 1-tape TM

Proof

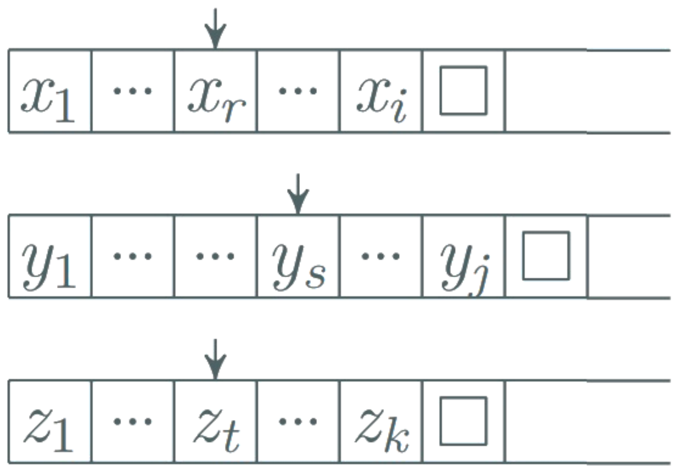
For each step of k-tape TM M, 1-tape M' operates its tape as:

- At the start of the simulation, the tape head of M' is on the leftmost #
- Scans the tape from first # to $(k+1)$st # to read symbols under heads.
- Rescans to write new symbols and move heads.

Multi-tape TM equivalent to 1-tape TM

Example

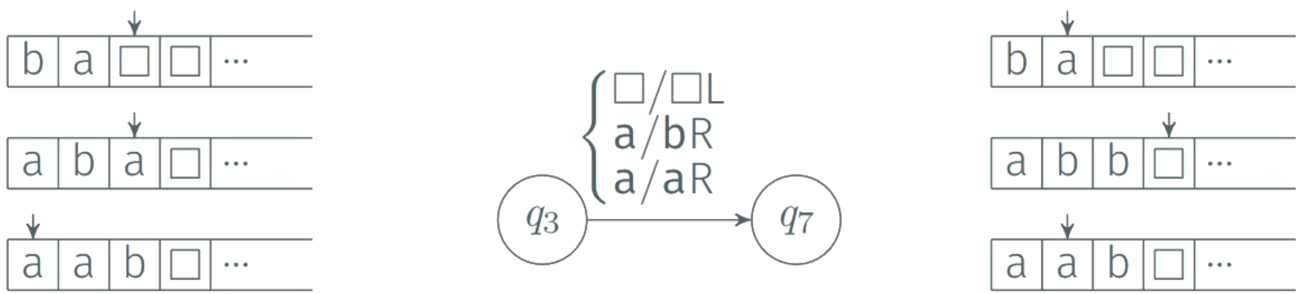
Simulate a 3-tape TM



Multi-tape TM equivalent to 1-tape TM

Example

Suppose the given TM M moves like this:



Multi-tape TM equivalent to 1-tape TM

Key points of simulation

To simulate a model M by another model N:

- Say how the state and storage of N is used to represent the state and storage of M
- Say what should be initially done to convert the input of N
- Say how each transition of M can be implemented by a sequence of transitions of N

Turing-recognizable ⟷ Multiple-tape TM

Language L is TM-recognizable if and only if some multi-tape TM recognizes L.

Nondeterministic TM

A **nondeterministic Turing machine** (NTM) M can have several options at every step. It is defined by the 7-tuple $M = (\Sigma, \Gamma, Q, \delta, q, q_{accept}, q_{reject})$, where

- Σ is input alphabet (without blank ␣)
- Γ is tape alphabet with $\{ ␣ \} \cup \Sigma \subseteq \Gamma$
- Q is a finite set, whose elements are called states δ is

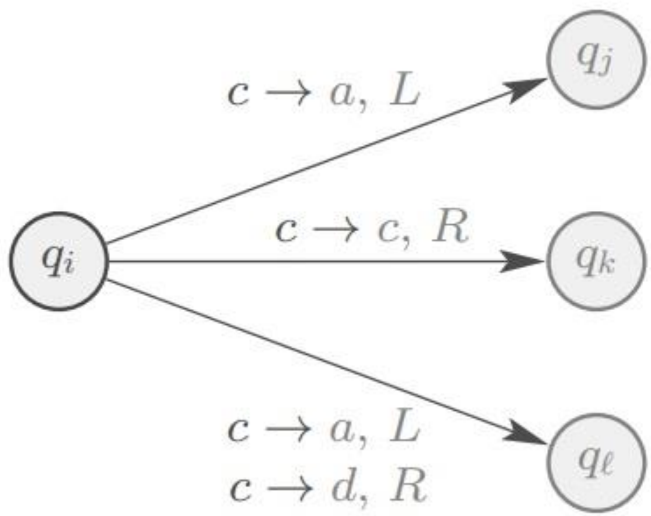
  transition function $\delta : Q \times \Gamma \to P(Q \times \Gamma \times \{L, R\})$
- q is start state $\in Q$
- q_{accept} is accept state $\in Q$
- q_{reject} is reject state $\in Q$

Nondeterministic TM

Transition

Transition function

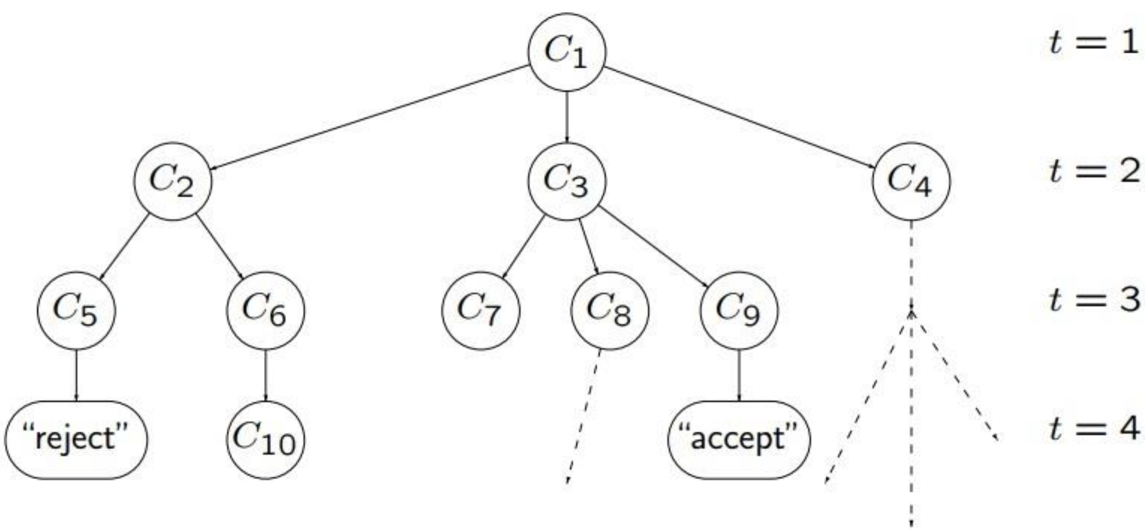
$$\delta : Q \times \Gamma \rightarrow P(Q \times \Gamma \times \{L, R\})$$



Nondeterministic TM (NTM)

Computation

With any input w, computation of NTM is represented by a configuration tree.



If ∃ (at least) one accepting leaf, then NTM accepts.

NTM equivalent to TM

Every nondeterministic TM has an equivalent deterministic TM.

Proof

- Build TM D to simulate NTM N on each input w. D tries all possible branches of N's tree of configurations.
- If D finds any accepting configuration, then it accepts input w.
- If all branches reject, then D rejects input w.
- If no branch accepts and at least one loops, then D loops on w.

Address

- Every node in the tree has at most b children.
- b is size of largest set of possible choices for N's transition function.
- Every node in tree has an address that is a string over the alphabet $\Gamma_b = \{1, 2, \ldots, b\}$

  To get to node with address 231:

  (1) start at root

  (2) take second branch

  (3) then take third branch

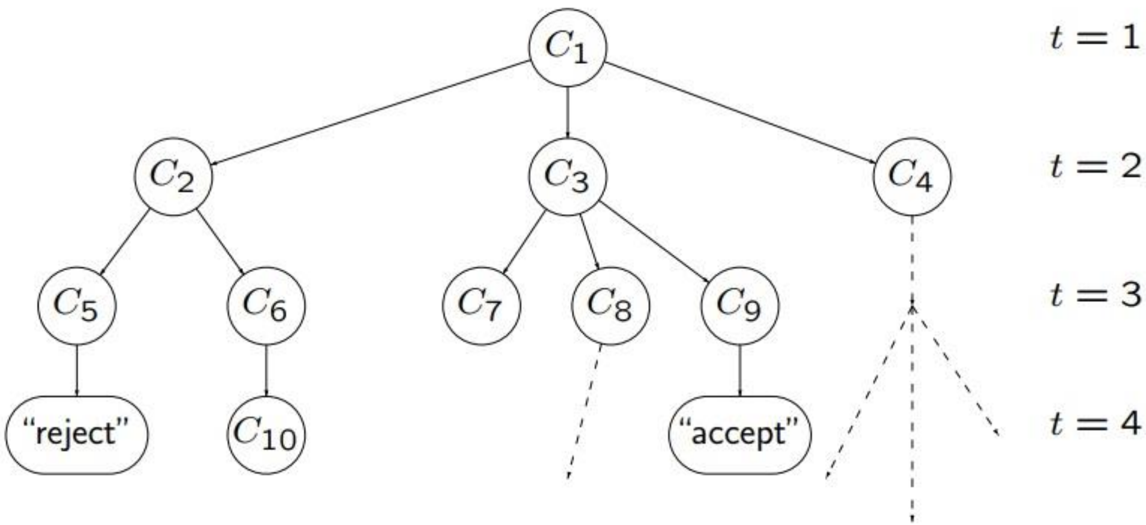
  (4) then take first branch

- Ignore meaningless addresses.
- Visit nodes in breadth-first search order by listing addresses in Γ_b^* in string order:

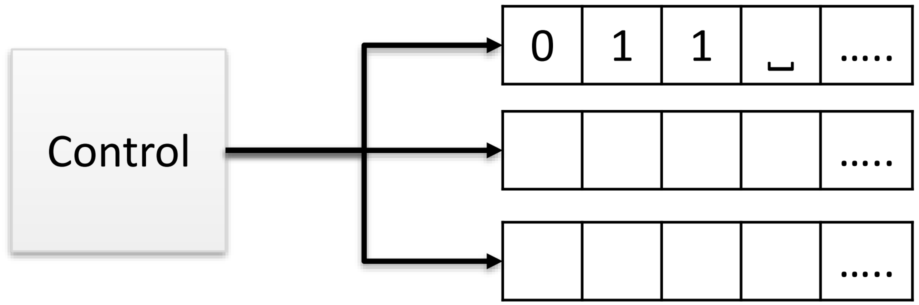
$$\varepsilon, 1, 2, \ldots, b, 11, 12, \ldots, 1b, 21, 22, \ldots$$

Address

Example



Simulating NTM by DTM



1. Initially, input tape contains input string w. Simulation and address tapes are initially empty.
2. Copy input tape to simulation tape.
3. Use simulation tape to simulate NTM N on input w on path in tree from root to the address on address tape.
   - At each node, consult next symbol on address tape to determine which branch to take.
   - Accept if accepting configuration reached.
   - Skip to next step if
     a. symbols on address tape exhausted
     b. nondeterministic choice invalid
     c. rejecting configuration reached
4. Replace string on address tape with next string in Γ_b^* in string order, and go to Stage 2

Turing-recognizable and Turing-decidable

Turing-recognizable ⟷ Multiple-tape TM

Language L is TM-recognizable if a NTM recognizes it.

- Multiple-tape TMs and NTMs are not more powerful than standard TMs

Turing-Decidable ⟷ NTM decidable

A nondeterministic TM is a decider if all branches halt on all inputs.

A language is decidable if some nondeterministic TM decides it.

Enumerable Language and Enumerator

A language is **enumerable** if some TM recognizes it.

An enumerator is usually represented as a 2-tape Turing machine. One working tape, and one print tape.

Language A is Turing-recognizable if some enumerator enumerates it.

Encoding

Input to a Turing machine is a string of symbols over an alphabet.

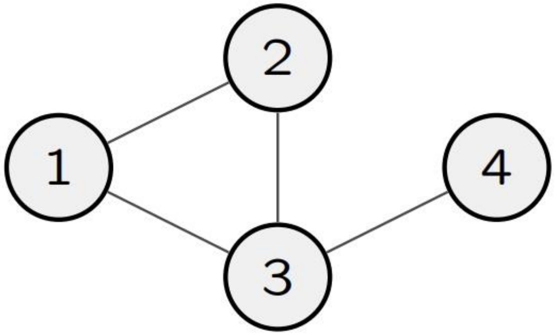
When we want TMs to work on different objects such as:

- Polynomials
- Graphs
- Grammars
- etc

We need to encode this object as a string of symbols over an alphabet.

Encoding of Graph

Given an undirected graph G

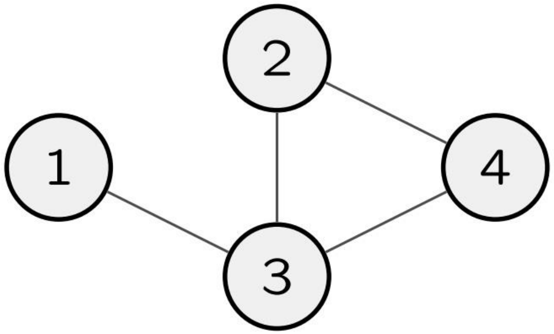


One possible encoding

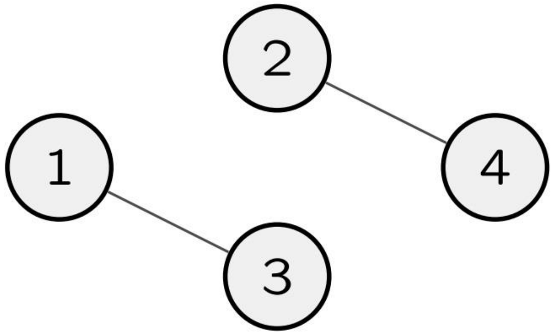
$\langle \mathrm{G} \rangle$ of graph G is string of symbols over some alphabet Σ, where the string starts with list of nodes and followed by list of edges.

Encoding of Graph

An undirected graph is **connected** if every node can be reached from any other node by travelling along edge



Connected graph G_1



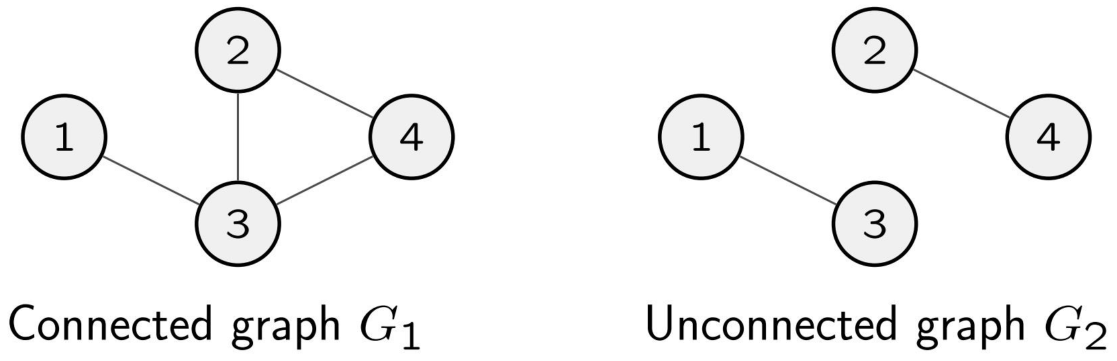
Unconnected graph G_2

Let A be the language consisting of strings representing connected undirected graph

$$\mathrm{A} = \{ \langle \mathrm{G} \rangle \mid \mathrm{G} \text{ is a connected undirected graph } \}$$

So $\langle \mathrm{G}_1 \rangle \in$ A and $\langle \mathrm{G}_2 \rangle \notin$ A.

TM to decide the connectedness of a Graph



A = { ⟨G⟩ | G is a connected undirected graph }

On input ⟨G⟩ ∈ Ω, where G is an undirected graph:

1. Check if G is a valid graph encoding. If not, reject.
2. Select first node of G and mark it.
3. Repeat until no new nodes marked.
4. For each node in G, mark it if it's attached by an edge to a node already marked
5. Scan all nodes of G to see whether they all are marked. If they are, accept; otherwise, reject."

Ω denotes the **universe** of a decision problem, comprising all instances.

TM to decide the connectedness of a Graph

For TM M that decides $A = \{ \langle G \rangle \mid G$ is a connected undirected graph $\}$

$$\Omega = \{\langle G \rangle \mid G \text{ is an undirected graph } \}$$

Step 1 checks that input $\langle G \rangle \in \Omega$ is valid encoding:

- Two list

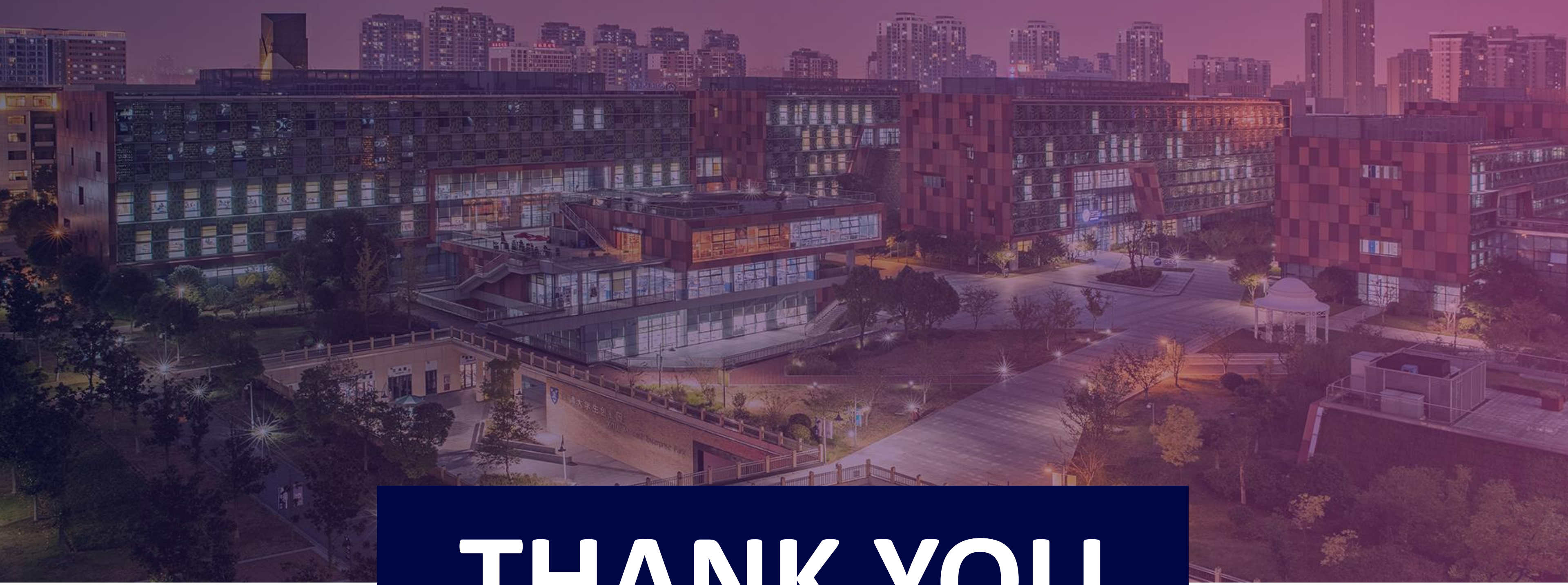
  (a) first is a list of numbers

  (b) second is a list of pairs of numbers

- First list contains no duplicate
- Every node in second list appears in first list

Step 2-5 check if G is connected.

Language Hierarchy



THANK YOU

Xi'an Jiaotong-Liverpool University
西交利物浦大學

XJTLU | SCHOOL OF FILM AND TV ARTS