CS4335: Design and Analysis of Algorithms

Lecturers:

Dr. Shuai Cheng Li

- Dept of CS, AC1-G6520, T: 9412, E: shuaicli@

Prof. Qingfu Zhang

- Dept. of CS, AC1-G7353, T: 8632, E: qingfu.zhang@

Course web site: canvas

Text Book:

- J. Kleinberg and E. Tardos, **Algorithm design**, Addison-Wesley, 2013.
- We will add more materials in the handout.

References:

- T. H. Cormen, C. E. Leiserson, R. L. Rivest, **Introduction to Algorithms**, The MIT Press, 2009.

- M.R. Garry and D. S. Johnson, **Computers and intractability, a guide to the theory of NP-completeness**, W.H. Freeman and company, 1979 one of the most famous books in CS.

Algorithms

Any **well-defined computational procedure** that takes some value, or set of values, as **input** and produces some value, or set of values, as **output**.

A **sequence of computational steps** that transform the **input** into **output**.

A sequence of computational steps for solving a ***well-specified*** **computational problem**.

Example of well-specified problem: Sorting

Input: a sequence of numbers: 1, 100, 8, 25, 11, 9, 2, 1, 200.

Output: a sorted (increasing order or decreasing order) sequence of numbers

1, 2, 8, 9, 11, 25, 100, 200.

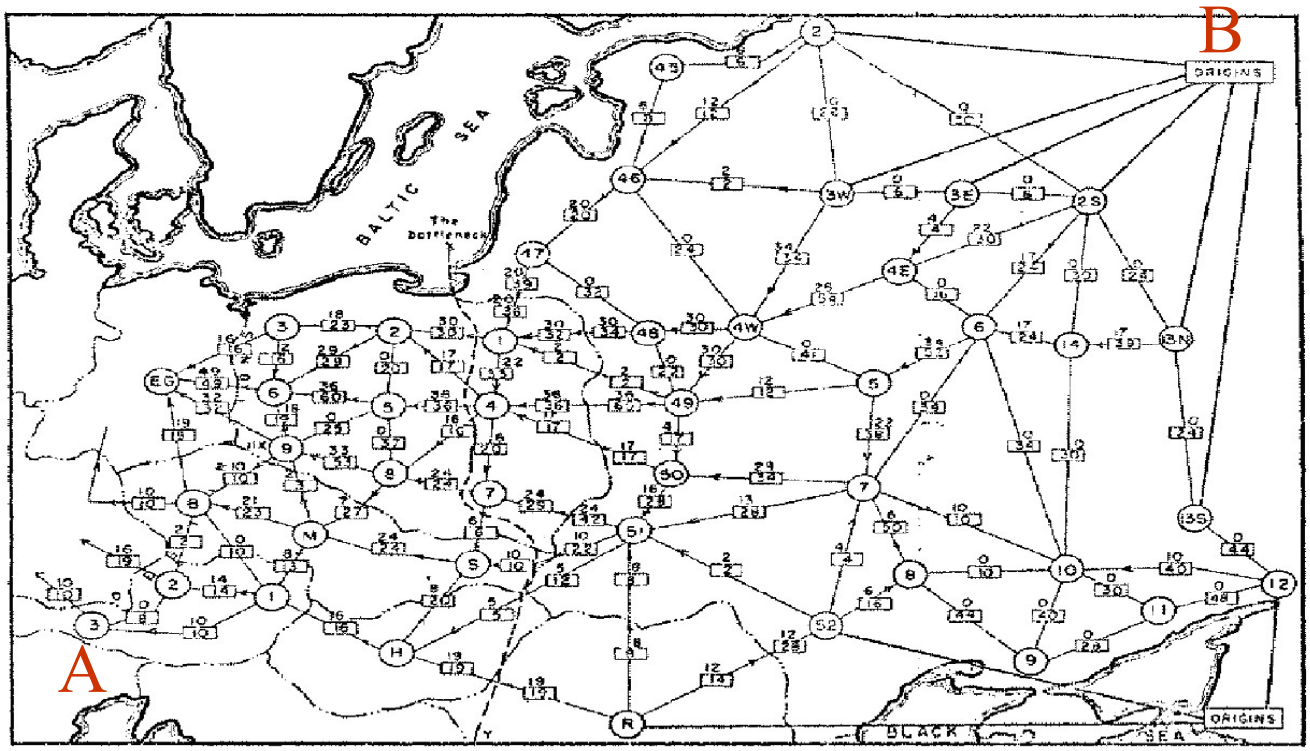
Another example:

Create web page that contains a list of papers using HTML.

-everybody can do it.

Do not need to design computational steps.

Soviet Rail Network, 1955



Reference: *On the history of the transportation and maximum flow problems.* Alexander Schrijver in Math Programming, 91: 3, 2002.

Find a shortest path from station A to station B.

-need serious thinking to get a correct algorithm.

GPS Navigation System

Given an electronic map (stored on a cellphone), the position of your car (provided by GPS), and the destination,

the system can tell you the way to go to the destination.

Tell you tern left or right 40 meters before according to the shortest path.

If you did not follow the direction, re-calculate the shortest path.

Dijkstra's Algorithm: (Recall)

Dijkstra's algorithm assumes that $w(e) \geq 0$ for each e in the graph.

maintain a set S of vertices such that

Every vertex $v \in S$, $d[v] = \delta(s, v)$, i.e., the shortest-path from s to v has been found. (Intial values: S=empty, d[s]=0 and d[v]=∞)

(a) select the vertex $u \in V\text{-}S$ such that

$d[u] = min\ \{d[x] | x \in V\text{-}S\}$. Set S=S$\cup${u}

(b) ***for*** each node v adjacent to u ***do*** RELAX(u, v, w).

Repeat step (a) and (b) until S=V.

Descriptions of Algorithms

- Flow charts
- Pseudo-codes
- Natural languages
- Computer Programs

The purpose:

Allow a *well-trained programmer* to write a program to solve the computational problem.

Any body who can talk about algorithm MUST have basic programming skills

Also CS3334: Data structures.

What We Cover:

1. Some classic algorithms in various domains

   Graph algorithms

   - Euler path, shortest path, minimum spanning trees, maximum flow, Hamilton Cycle, traveling salesman problem,

   String Algorithms

   - Exact string matching, Approximate string matching, Applications in web searching engines

   Scheduling problems

   Computational Geometry

2. Techniques for designing efficient algorithms

   divide-and-conquer approach, greedy approach, dynamic programming, linear programming.

What We Cover(continued):

3. Introduction to computational complexity
   - NP-complete problems
4. Approximation algorithms
   - Vertex cover
   - Steiner trees
   - Traveling salesman problem

Why You have to take this course

Apply learned techniques to solve various problems

Have a sense of complexities of various problems in different domains

Find excuses if you cannot solve a problem.

College graduates vs. University graduates

Supervisor vs. low level working force

Why You have to take this course

You can apply learned techniques to solve various problems

Have a sense of complexities of various problems in different domains

College graduates vs. University graduates

Supervisor v.s. low level working force

The boss wants to produce programs to solve the following two problems

Euler circuit problem:

- given a graph *G*, find a way to go through each edge exactly once.

Hamilton circuit problem:

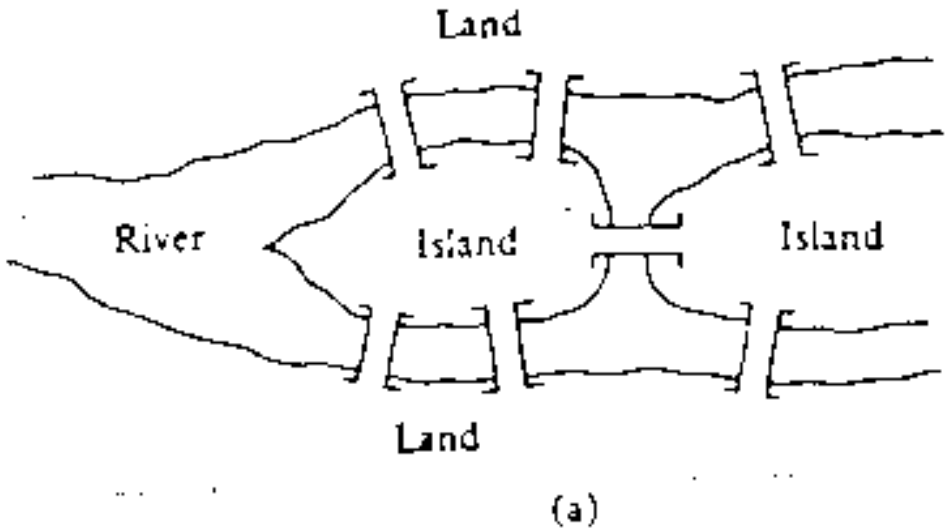
- given a graph *G*, find a way to go through each vertex exactly once.

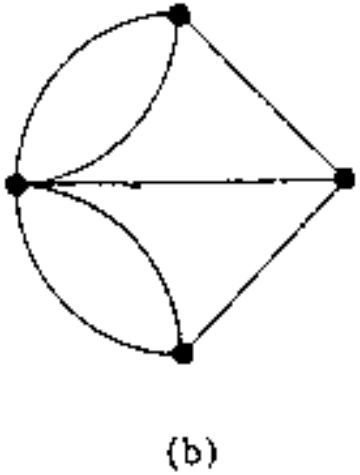
The two problems seem to be very similar.

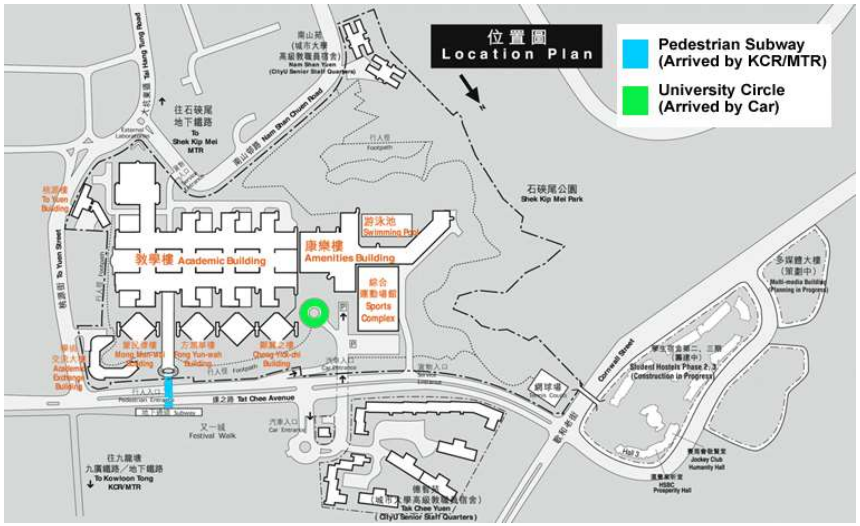
Person A takes the first problem and person B takes the second.

Outcome: Person A quickly completes the program, whereas person B works 24 hours per day and is ***fired*** after a few months.

Euler Circuit: The original Konigsberg bridge

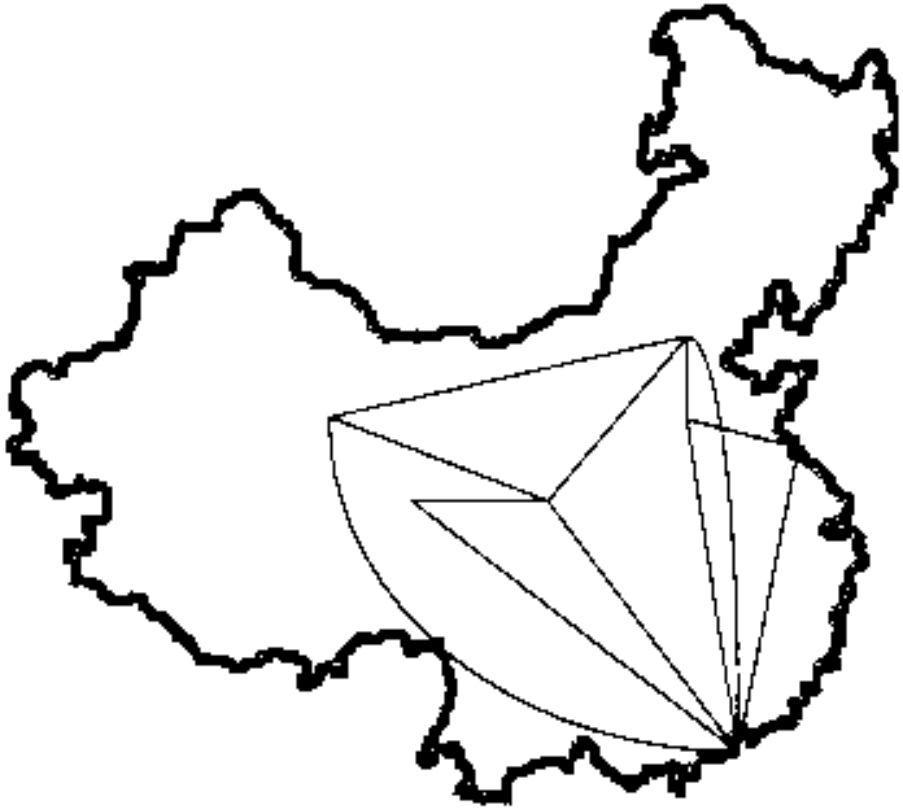






Königsberg graph

Hamilton Circuit



Traveling salesman problem (TSP),

Why?

Explanation:

Euler circuit problem can be easily solved in polynomial time.

Hamilton circuit problem is proved to be **NP-hard**.

So far, no body in the world can give a polynomial time algorithm for a NP-hard problem.

Conjecture: there does not exist polynomial time algorithm for this problem.

Evaluation of the Course

Course work: 30%

- 4 points for each of the **two** assignments
- 12 points for midterm exam (week 8)
- 0.5 point for each in-lecture exercise (from weeks 2 to 11)
- 0.5 point for each tutorial (for tutorials 2 to 11)

final exam: 70%

Evaluation of the Course (OBTL)

No.CILOs .

1. prove the correctness and analyze the running time of the basic algorithms for those classic problems in various domains;
2. apply the algorithms and design techniques to solve problems;
3. analyze the complexities of various problems in different domains.

For CILO 1- CILO 3, we have assignments to evaluate.

You have to pass all CILOs in order to pass the course.

The Process of Design an Algorithm

- Formulating the problem
  - with enough mathematical precision, we can ask a concrete question
  - start to solve it.
- Design the algorithm
  - list the “precise” steps. (an expert can translate the algorithm into a computer program.)
- Analyze the algorithm
  - prove that it is correct
  - establish the efficiency
    - the running time or sometimes, space complexity

How to Teach

Contents are divided into four classes

1. Basic part -- every body must understand in order to pass

2. Moderate part -- most of students should understand.
   Aim at B or above.

3. Hard part -- used to distinguish students.
   Aim at A or above.

4. Fun and Challenging part -- useful knowledge
   that will not be tested.

How to Learn

1. Attend **every** lecture (2 hours per week) and tutorial (1 hour per week)

2. Try to go with me when I am talking

3. Ask questions immediately

4. Try to fix all problems during the 1 hour tutorial

5. Ask others.

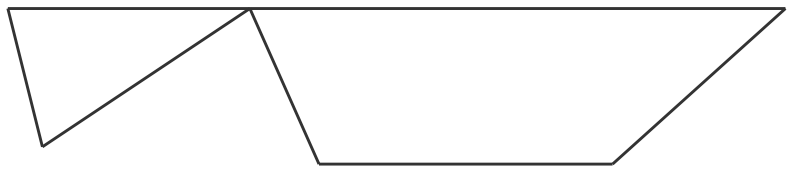
Terminologies

A *Graph G=(V,E):* V---set of vertices and E--set of edges.

Path in *G*: sequence $v_1, v_2, \ldots, v_k$ of vertices in V such that (v_i, v_{i+1}) is in E.

v_i and v_j could be the same

Circuit: A path $v_1, v_2, \ldots, v_k$ such that $v_1 = v_k$



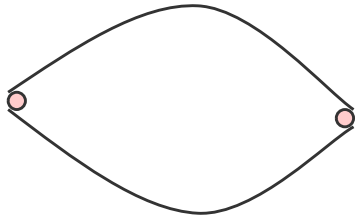
Degree of a vertex: number of edges incident to the vertex.

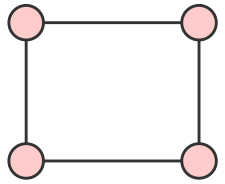
Euler Circuit Problem

Input: a connected graph $G=(V, E)$

Problem: is there a circuit in G that uses each edge exactly once.

Note: G can have *multiple* edges, .i.e., two or more edges connect vertices u and v.





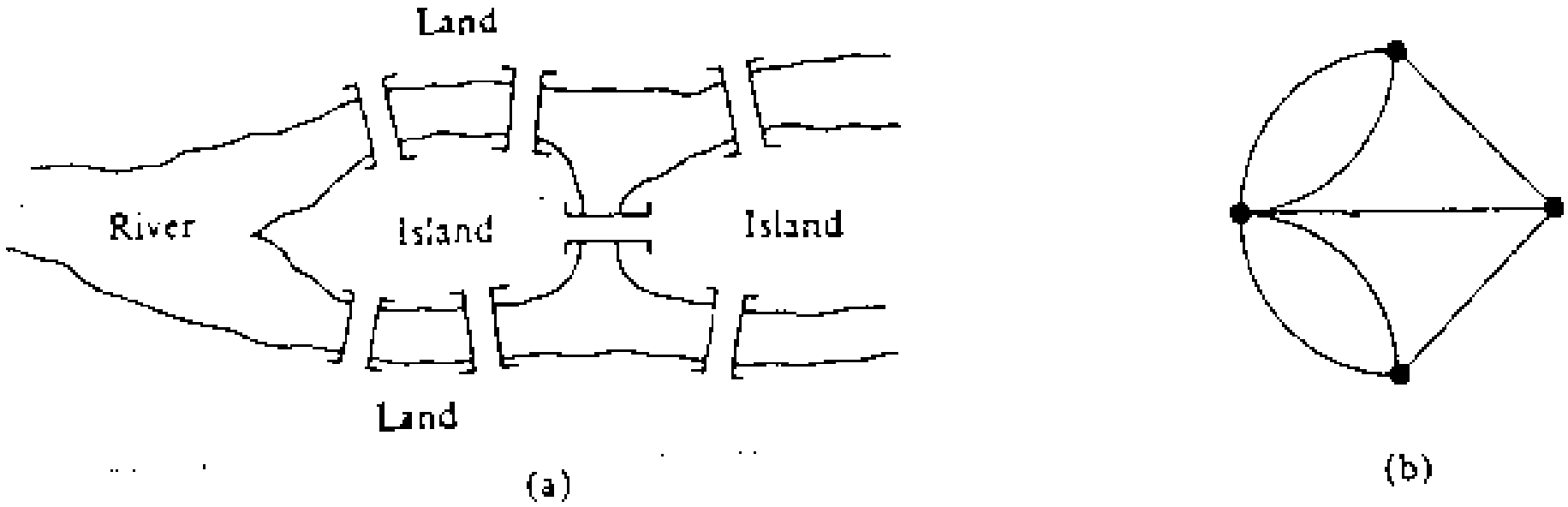
Euler Circuit problem (continued)

solved by Leonhard Euler [pronounced OIL-er] (1736)

The first problem solved by using graph theory

A graph is constructed to describe the town.

The original Konigsberg bridge (Figure 1)



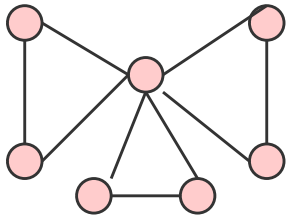
Königsberg graph

Theorem for Euler circuit

Theorem 1 (Euler's Theorem) A connected *graph has an Euler circuit* ***if and only if*** *all the vertices in the graph have even degrees.*

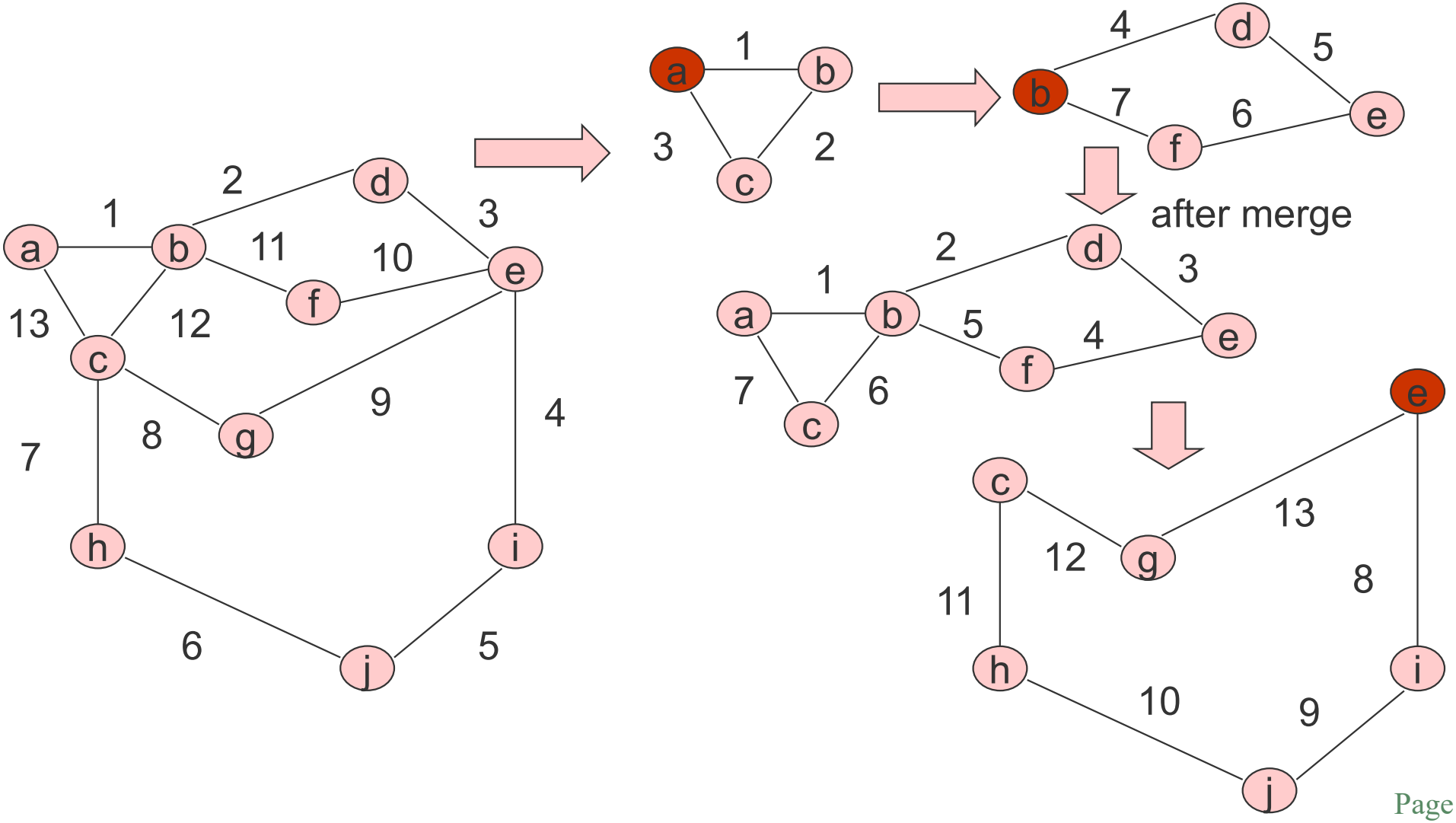
Proof: if an Euler circuit exists, then the degree of each node is even.

Going through the circuit, each time a vertex is visited, the degree is increased by 2. Thus, the degree of each vertex is even.



Example 1:

C1:abca, C2:bdefb, =>C3:abdefbca.
C4:eijhcge. C3+C4=>abdeijhcgefbca



An efficient algorithm for Euler circuit

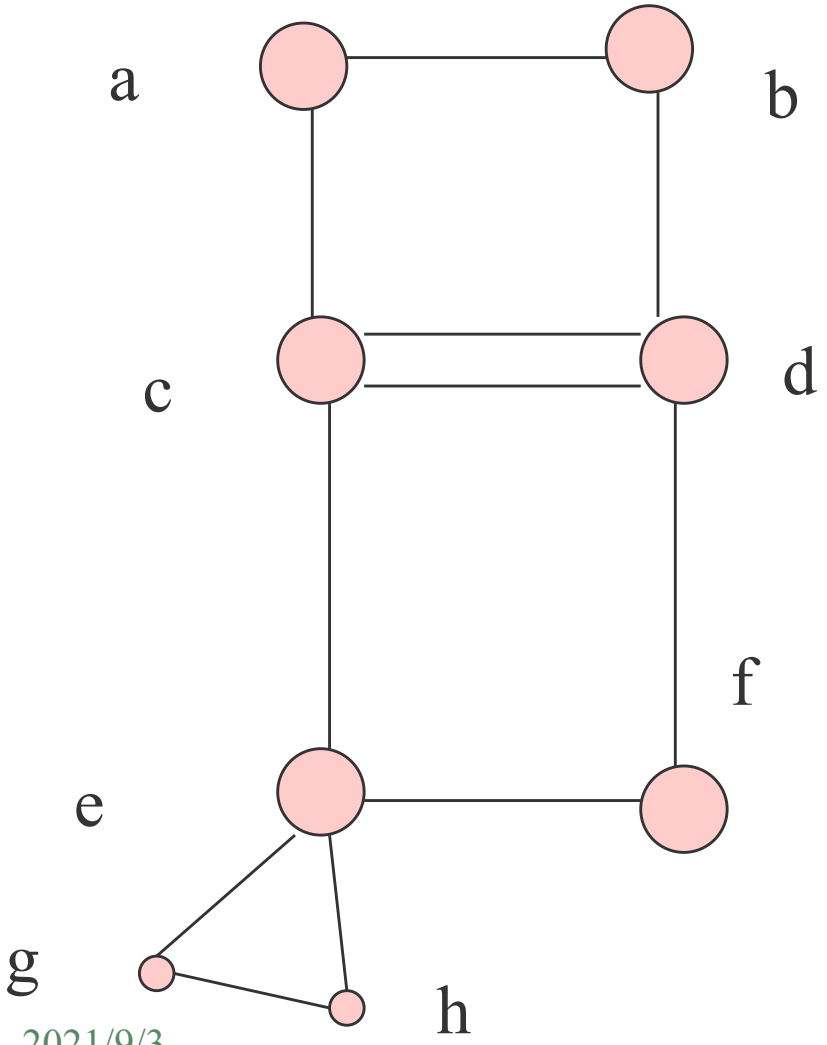
1. Starting with any vertex *u* in *G*, take an unused edge *(u,v)* (if there is any) incident to *u*
2. Do Step 1 for *v* and continue the process until *v* has no unused edge. (a circuit C is obtained)
3. If every node in C has no unused edge, stop.
4. Otherwise, select a vertex, say, *u* in *C*, with some unused edge incident to *u* and do Steps 1 and 2 until another circuit is obtained.
5. Merge the two circuits obtained to form one circuit
6. Goto Step 3.

Proof of Theorem 1: if the degree of every node is even, then there is an Euler circuit.

We give a way to find an Euler circuit for a graph in which every vertex has an even degree.

- Since each node *v* has even degree, when we first enter *v*, there is an unused edge that can be used to get out *v*.
- The only exception is when v is a starting node.
- Then we get a circuit (may not contain all edges in G)
- If every node in the circuit has no unused edge, all the edges in G have been used since G is connected.
- Otherwise, we can construct another circuit, merge the two circuits and get a larger circuit.
- In this way, every edge in G can be used.

Example 2:



First circuit: a-b-d-c-a

Second circuit: d-f-e-c-d.

Merge: a-b-d-f-e-c-d-c-a.

3rd circuit: e-g-h-e

Merge: a-b-d-f-e-g-h-e-c-d-c-a

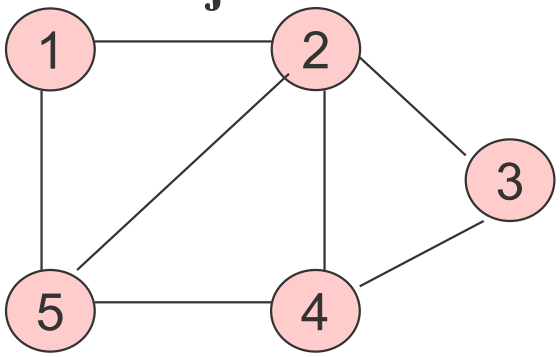
Note: There are multiple edges.

Representations of Graphs (Fun Part)

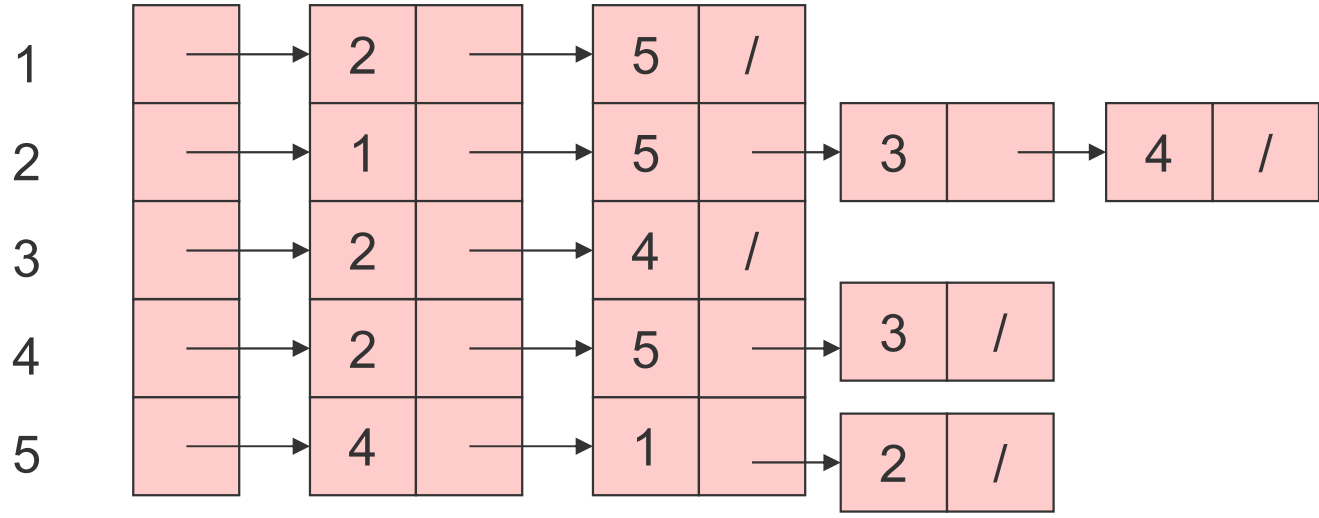
- Two standard ways
  - Adjacency-list representation
    - Space required $O(|E|)$
  - Adjacency-matrix representation
    - Space required $O(n^2)$.
- Depending on problems, both representations are useful.

Adjacency-list representation

- Let G=(V, E) be a graph.
  - V– set of nodes (vertices)
  - E– set of edges.
- For each u∈V, the adjacency list Adj[u] contains all nodes in V that are adjacent to u.

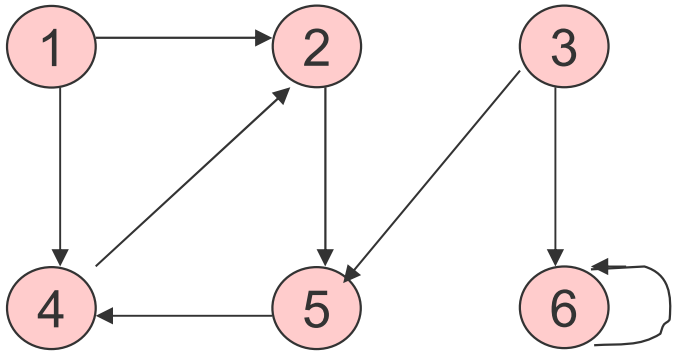


(a)

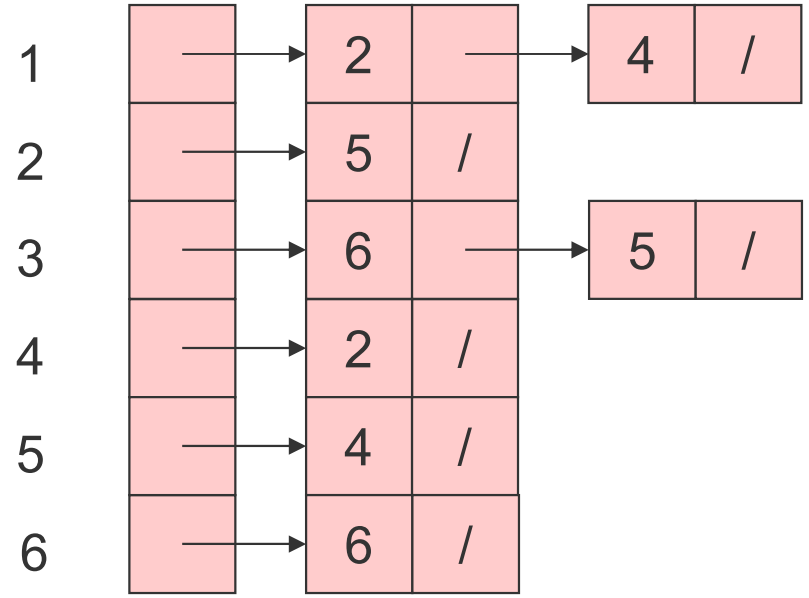


(b)

Adjacency-list representation for directed graph



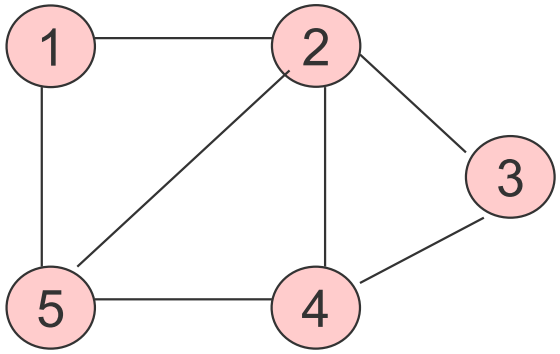
(a)



(b)

Adjacency-matrix representation

- Assume that the nodes are numbered *1, 2, …, n*.
- The adjacency-matrix consists of a $|V| \times |V|$ matrix $A=(a_{ij})$ such that $a_{ij}= 1$ if $(i,j) \in E$, otherwise $a_{ij}= 0$.

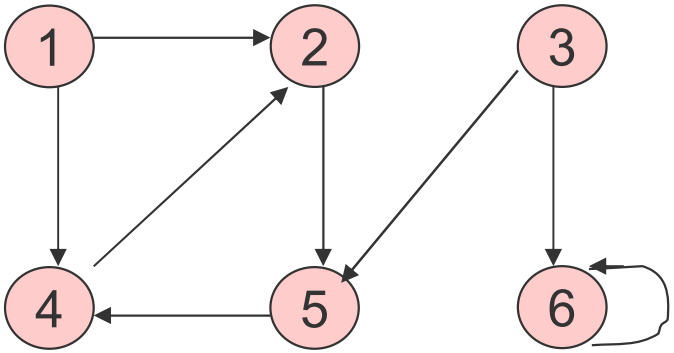


(a)

	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

(c)

Adjacency-matrix representation for directed graph
It is NOT symmetric.



(a)

	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

(c)

Implementation of Euler circuit algorithm (Not required)

- Data structures:
  - Adjacency-list representation
  - Each node in V has an adjacency list
  - Also, we have two lists to store the circuits
    - One for the circuit produced in Steps 1-2.
    - One for the circuit produced in Step 4

In Step 1: when we take an unused edge (u, v), this edge is deleted from the adjacency-list of nodes u and v.

Implementation of Euler circuit algorithm

In Step 2: if the adjacency list of *v* is empty, *v* has no unused edge.

1. Testing whether adjacency-list v is empty.
2. A circuit (may not contain all edges) is obtained if the above condition is true.

In Step 3: if each of the nodes in C, the adjacency-list is empty, stop.

In step 4: if some adjacency-list is not empty, use it in step 4.

Time complexity (Fun Part)

- If it takes O(|V|) time to add an edge to a circuit, then the time complexity is O(|E||V|).

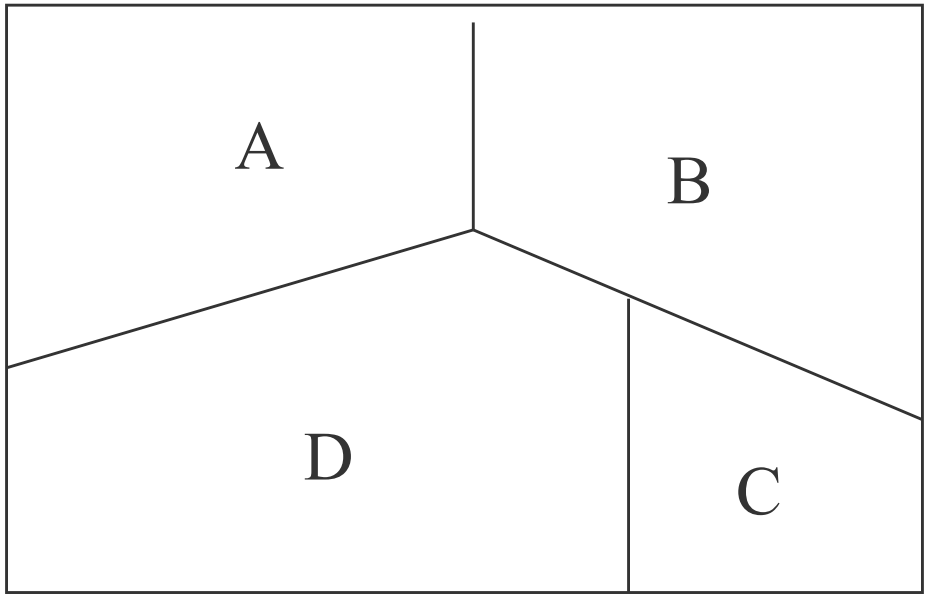
This can be done by using an adjacency list.

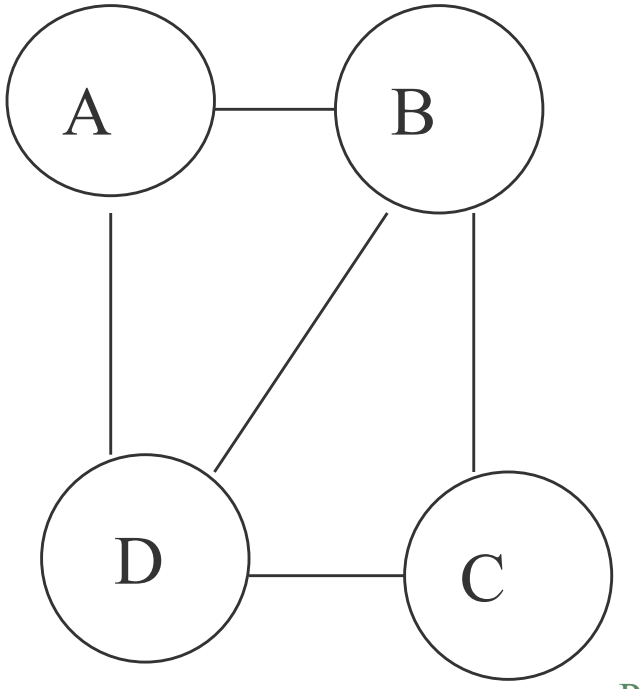
- The implementation is *tricky*
- Even O(|V||E|) algorithm is not trivial.
- Do not be disappointed if you do not completely understand the implementation
- **The best algorithm takes O(|E|) time.**
- **Euler Circuit: A complete example for Designing Algorithms.**

  -formulation, design algorithm, analysis.

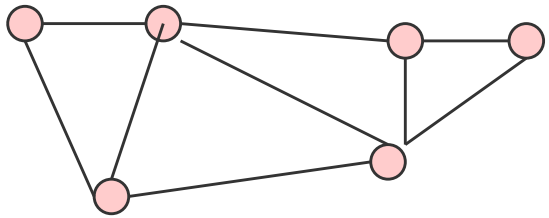
Application 1:

In a map, two countries may share a common border. Given a map, we want to know if it is possible to find a tour starting from a country, going across each shared border once and come back to the starting country.





Application 2:



Formulating a graph problem:

Given a map containing Guizhou, Hunan, Jiangxi, Fujian, Guangxi, Guangdong, construct a graph such that if two provinces share some common boarder, then there is an edge connecting the two corresponding vertices.

If we want to find a tour to visit each province once, then we need to find a Hamilton circuit in the graph.

Challenge Problem : (only for those who are interested)

(1) Prove that given a graph G=(V, E), one can always add some edges to the graph such that the new graph (multiple edges are allowed) has an Euler circuit.

(2) Design an algorithm to add *minimum* number of edges to an input graph such that the resulting graph has an Euler circuit.

(3) Prove the correctness of your algorithm.

Summary of Euler Circuit Algorithm

- Design a good algorithm needs two parts
  1. Theorem, high level part
  2. Implementation: low level part. Data structures are important.
     - Our course emphasizes the first part and demonstrates the second part whenever possible.
     - We will not emphasize too much about data structures.

Data Structures Needed: Heap and disjoint set