CS4335: Design and Analysis of Algorithms

- **Who we are:**
  - Dr. Lusheng WANG
  - Dept. of Computer Science
  - office: B6422
  - phone: 2788 9820
  - e-mail: lwang@cs.cityu.edu.hk
  - Course web site: http://www.cs.cityu.edu.hk/~lwang/ccs3335.html
  - Wong Tsui Fong Helena will do tutorials for all groups.
  - **Lecture: Tuesday (15:30-17:20)**
    Monday (19:30-21:20)

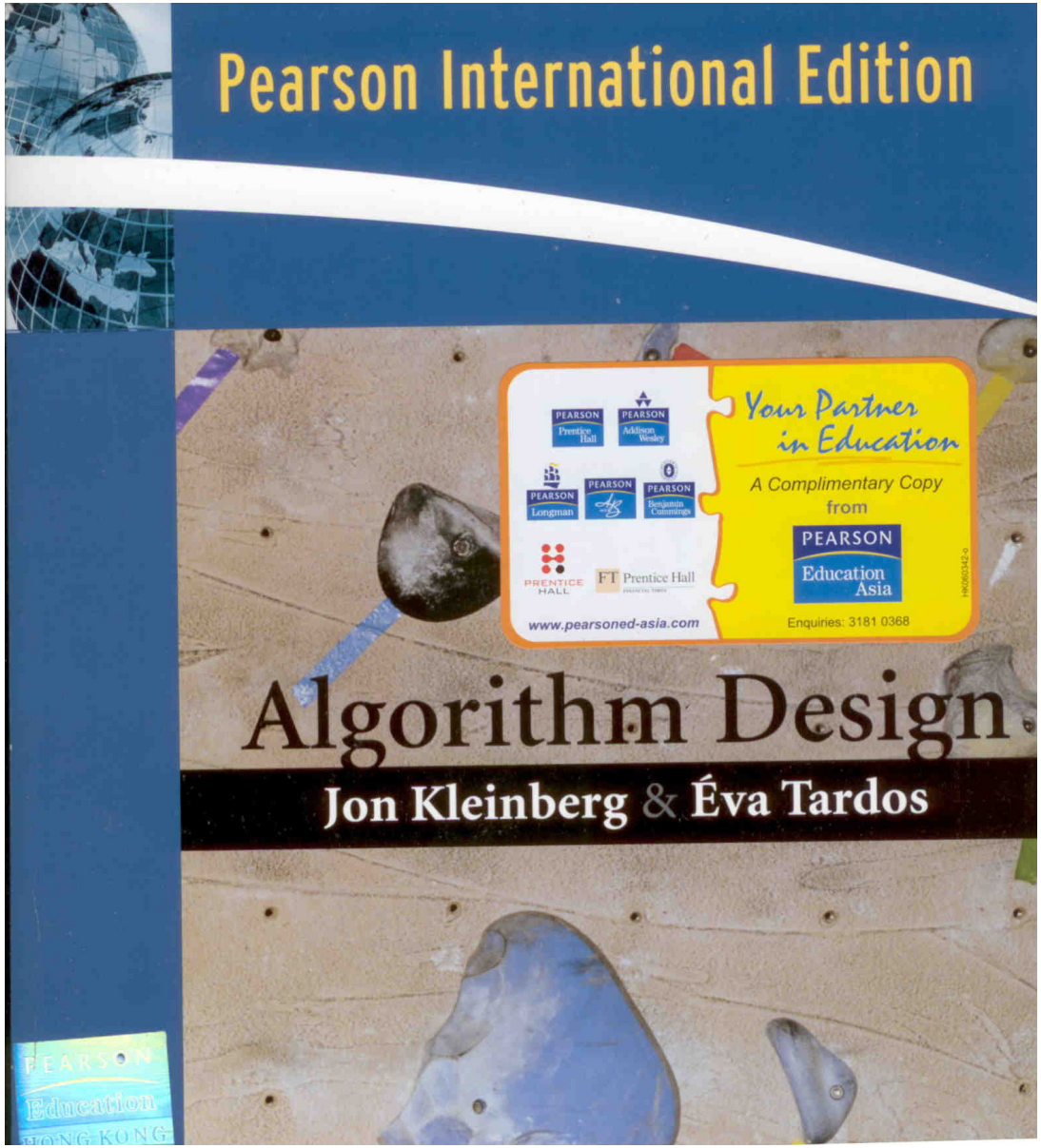
Text Book:

- J. Kleinberg and E. Tardos, Algorithm design, Addison-Wesley, 2005.
- We will add more material in the handout.

References:

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

  http://theory.lcs.mit.edu/~clr/
- R. Sedgewick, Algorithms in C++, Addison-Wesley, 2002.
- A. Levitin, Introduction to the design and analysis of algorithms, Addison-Wesley, 2003.
- M.R. Garry and D. S. Johnson, Computers and intractability, a guide to the theory of NP-completeness, W.H. Freeman and company, 1979


Pearson International Edition
Your Partner in Education
A Complimentary Copy from
PEARSON Education Asia
Enquiries: 3181 0368
www.pearsoned-asia.com
Algorithm Design
Jon Kleinberg & Éva Tardos
PEARSON Education HONG KONG

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.

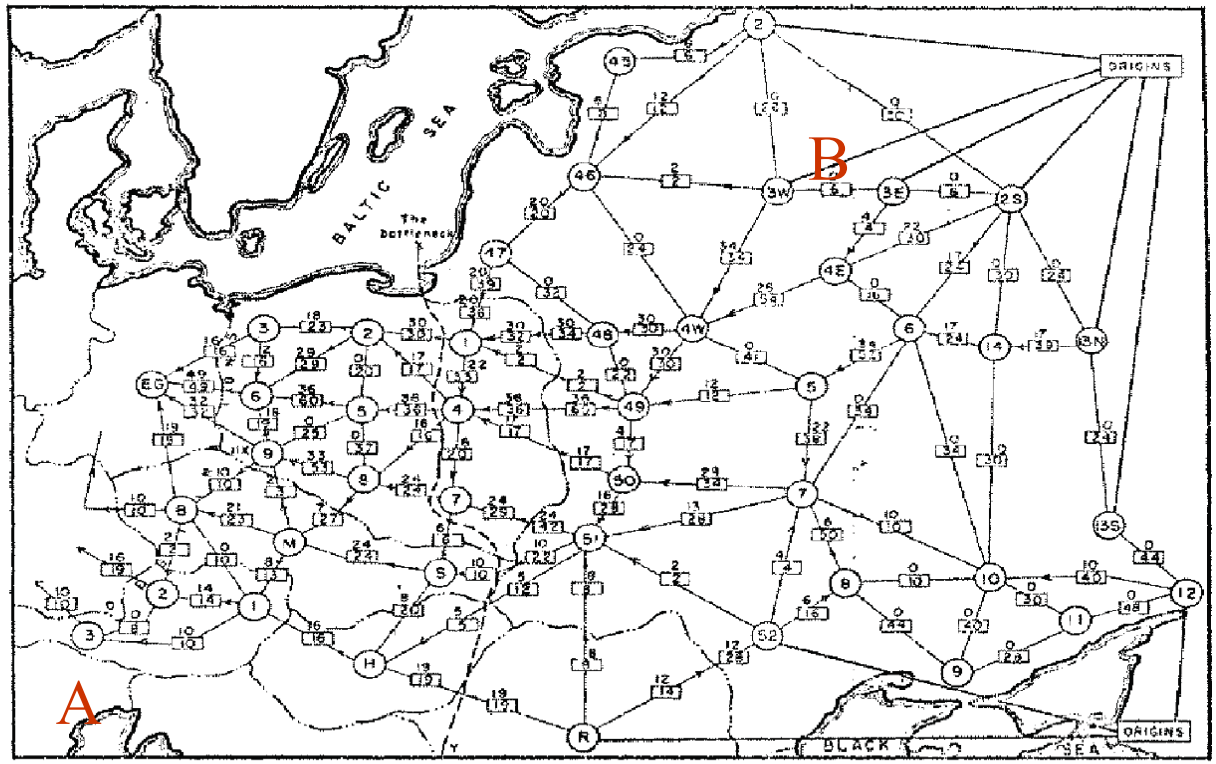
Another example:

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

-everybody can do it.

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.

A Real-Time Driver's Direction System

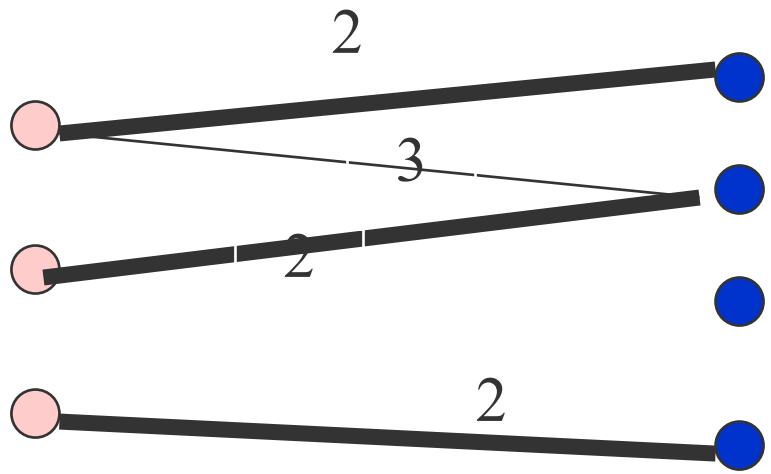
Given an electronic map (stored on a computer), the position of your car (provided by GPS), and the destination,

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.
- 400 US$ each.

On-line Auction: maximize the benefit

There are a set of customers and a set of products.



- Every customer can buy only one.
- Decide: which product goes to which customer.
- Maximum Matching for Bipartite Graph

Descriptions of Algorithms

- Flow chart
- Pseudo-code
- Programs
- Natural languages

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,
   - 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

What We Cover(continued):

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

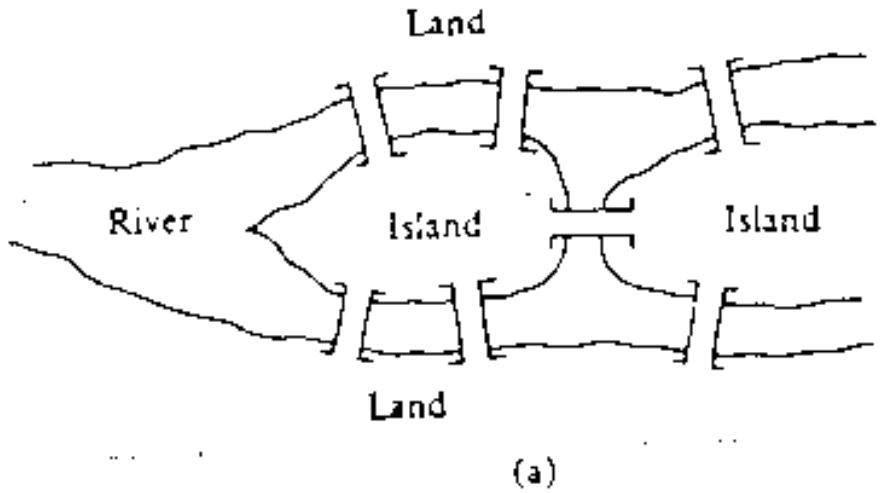
Why You have to take this course

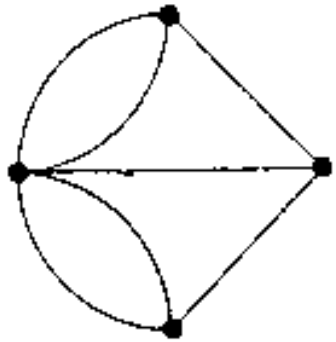
- You can apply learned techniques to solve various problems
- Have a sense of complexities of various problems in different domains
- College graduates vs. University graduates
- Supervisor v.s. low level working force

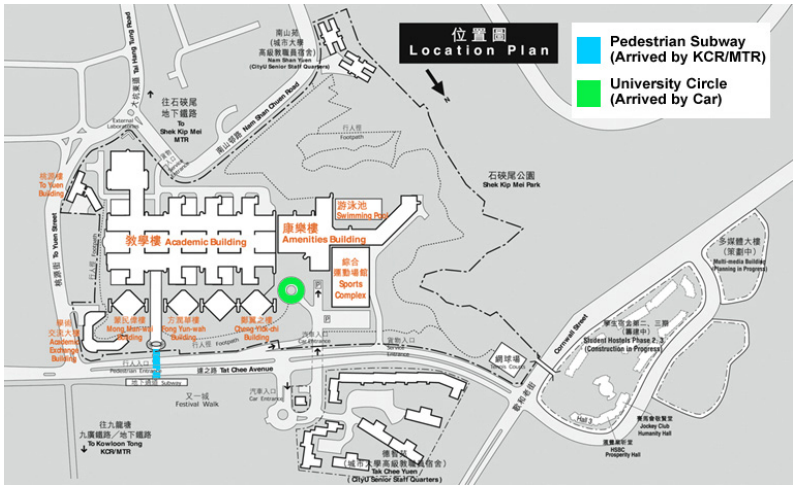
A joke:

- 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

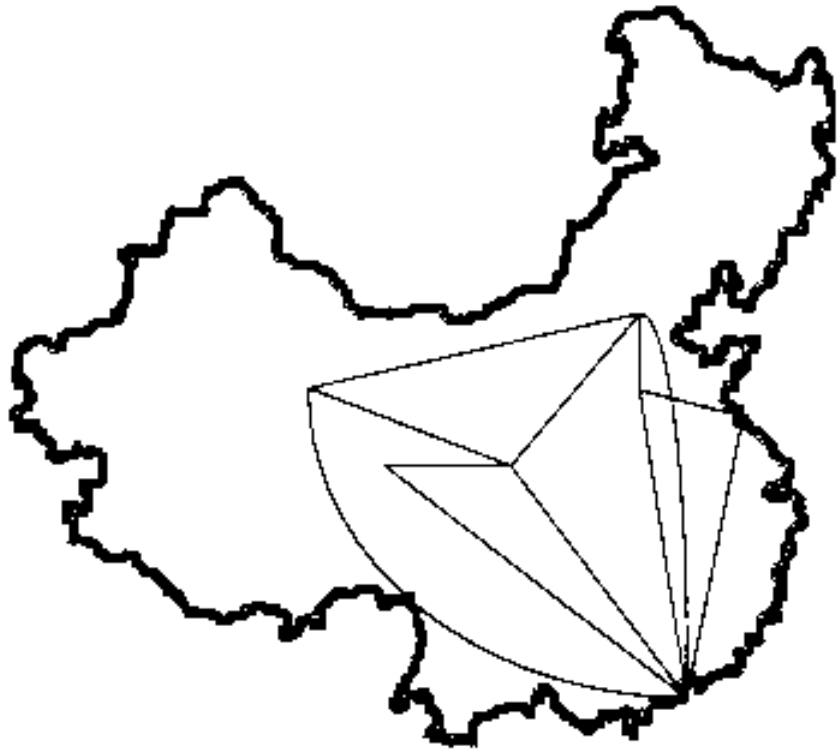






Königsberg graph

Hamilton Circuit

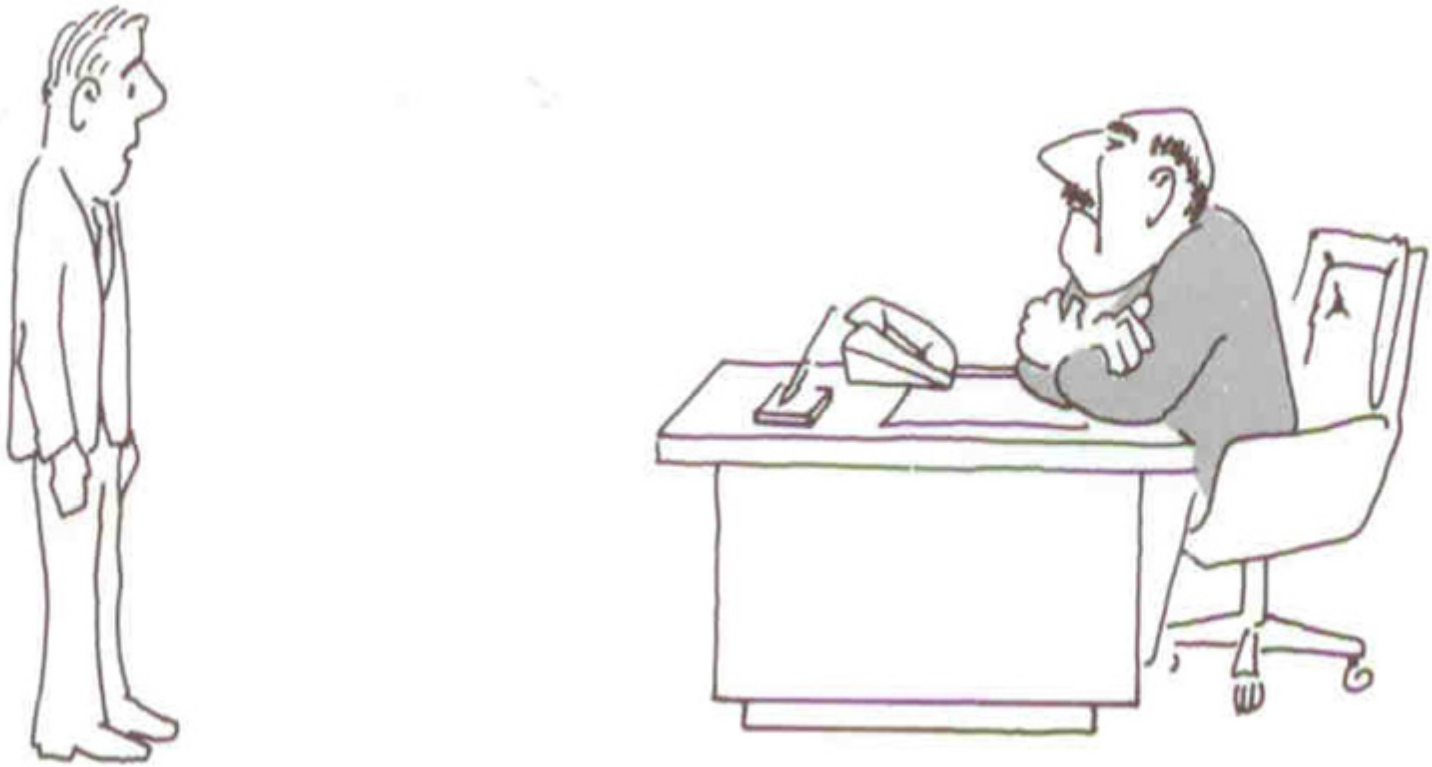


Traveling salesman problem (TSP),

A joke (continued):

- **Why?** no body in the company has taken CS3335.
- **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.

"I can't find an efficient algorithm, but neither can all these famous people."



"I can't find an efficient algorithm, I guess I'm just too dumb."

"I can't find an efficient algorithm, because no such algorithm is possible!"

Evaluation of the Course

- Course work: 30%
  - Four assignments
    - 6 points for each of the first **three** assignments
    - 12 points for the last assignment
    - Working load is Not heavy

- A final exam: 70%;

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 part -- just illustrate that some interesting things
   can be done if one works very hard.
   -- useful knowledge that will not be tested.
   -- I will give some challenging problems for fun.

Challenging Problems

For those who have ***extra energy***, we have challenging problems.

• Challenging problems will be distributed at the end of ***some*** lectures.

•No mark will be given for those challenge problems.

•I will record who completely solved the problem. (The records will be used to decide the boundary cases.)

•Good Training for solving new problems

- •Have high chance to solve the hard problems in the final exam.
- •Helpful for getting good marks in the final exam.

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.

The Process of Design an Algorithm

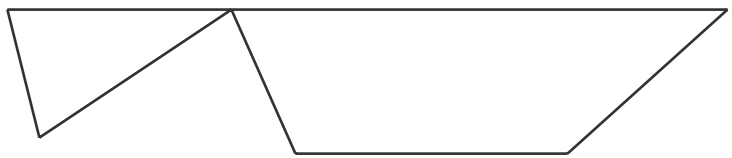
- Formulating the problem
  - with enough mathematical precision
  - we can ask a concrete question
  - start to solve it.
- Design the algorithm
  - list the "precise" steps. (an expert can translate the algorithm into a computer program.)
- Analyze the algorithm
  - prove that it is correct
  - establish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

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 $v_1 = v_k$

.

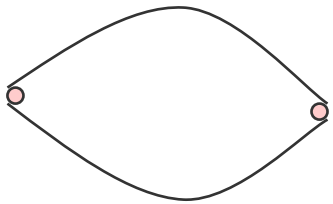


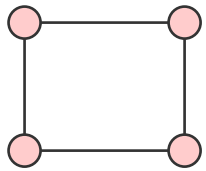
Degree of a vertex: number of edges incident to the vertex.

Euler circuit

- 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.

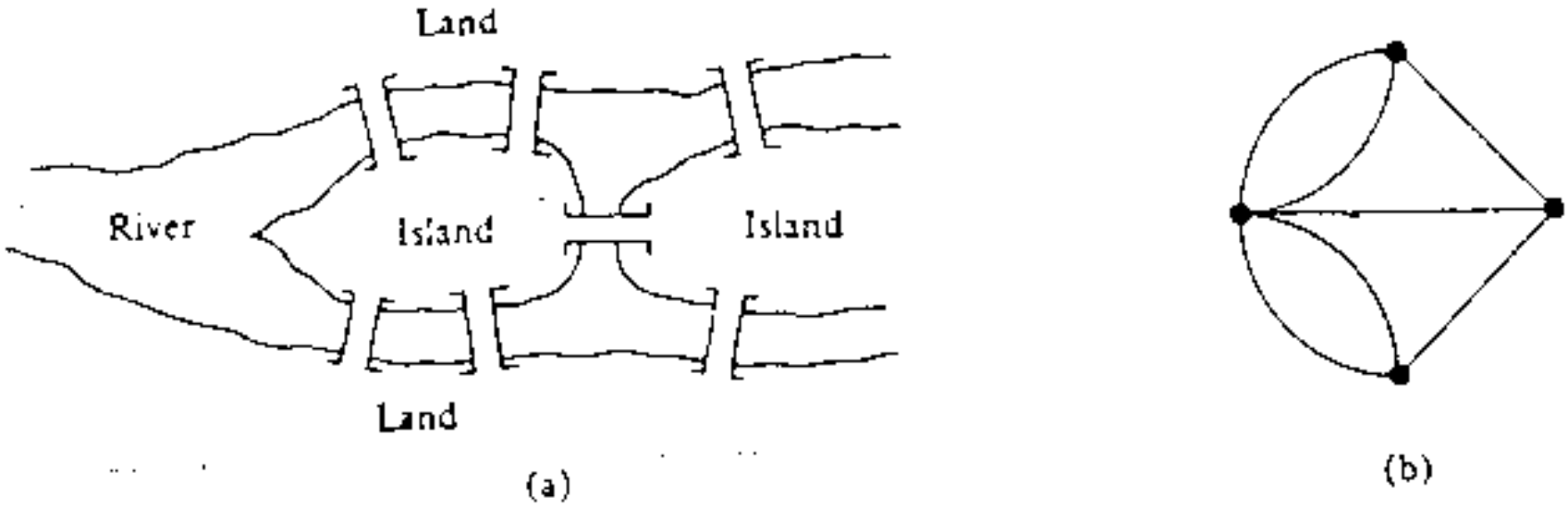




Story:

- The problem is called Konigsberg bridge problem
  - it asks if it is possible to take a walk in the town shown in Figure 1 (a) crossing each bridge exactly once and returning home.
- solved by Leonhard Euler [pronounced OIL-er] (1736)
- The first problem solved by using graph theory
- A graph is constructed to describe the town.
- (See Figure 1 (b).)

The original Konigsberg bridge (Figure 1)



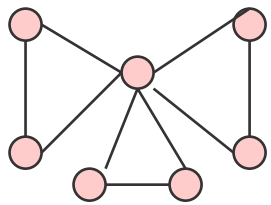
Königsberg graph

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 degree.*

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.



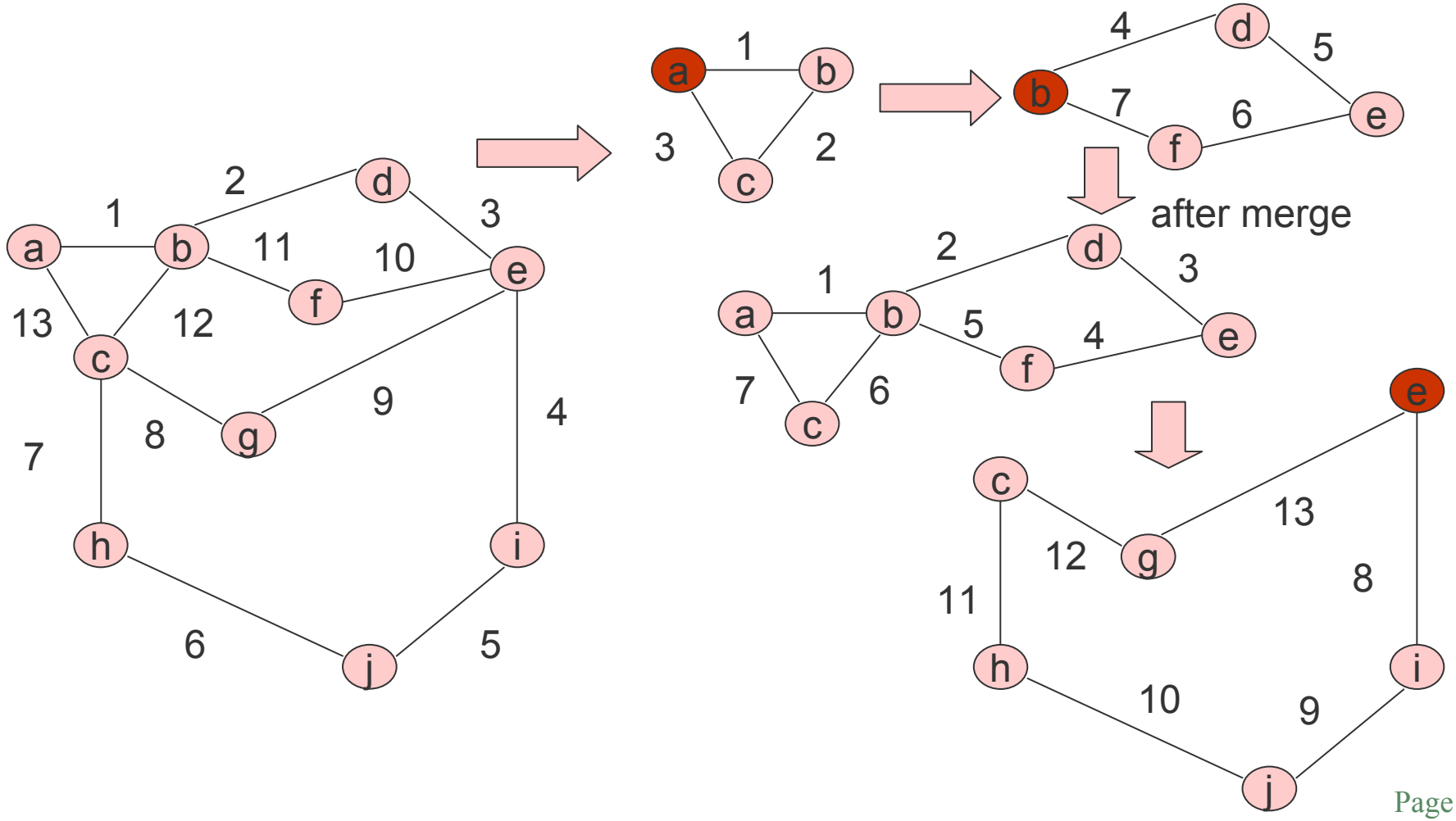
Proof of Theorem 1: if the degree of every node is even, then there is an Euler circuit.

We give 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 1:

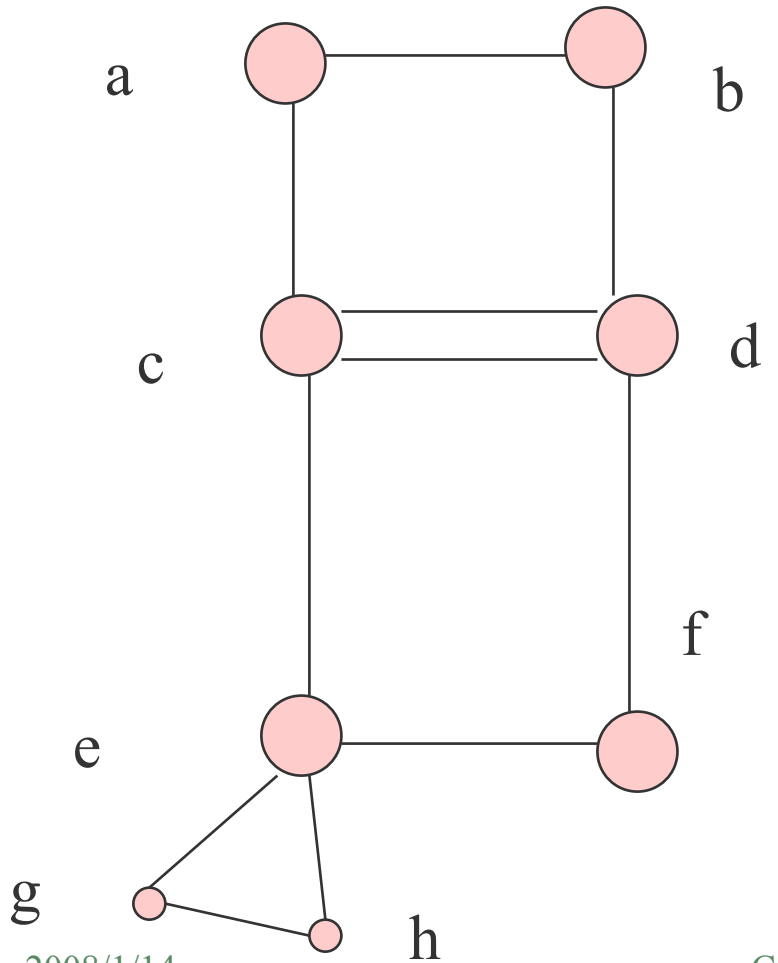
C1:abca, C2:bdefb, =>C3:abdefbca.
C4:eijhcge. C3+C4=>abdeijhcgefbca



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.

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

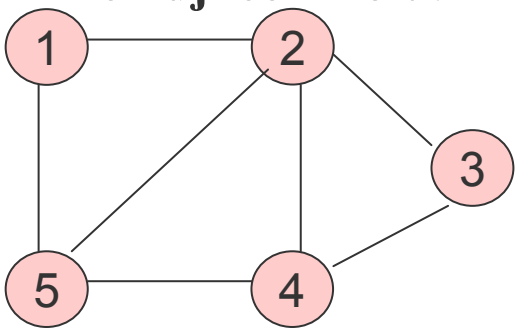
Note: There are multiple edges.

Representations of Graphs

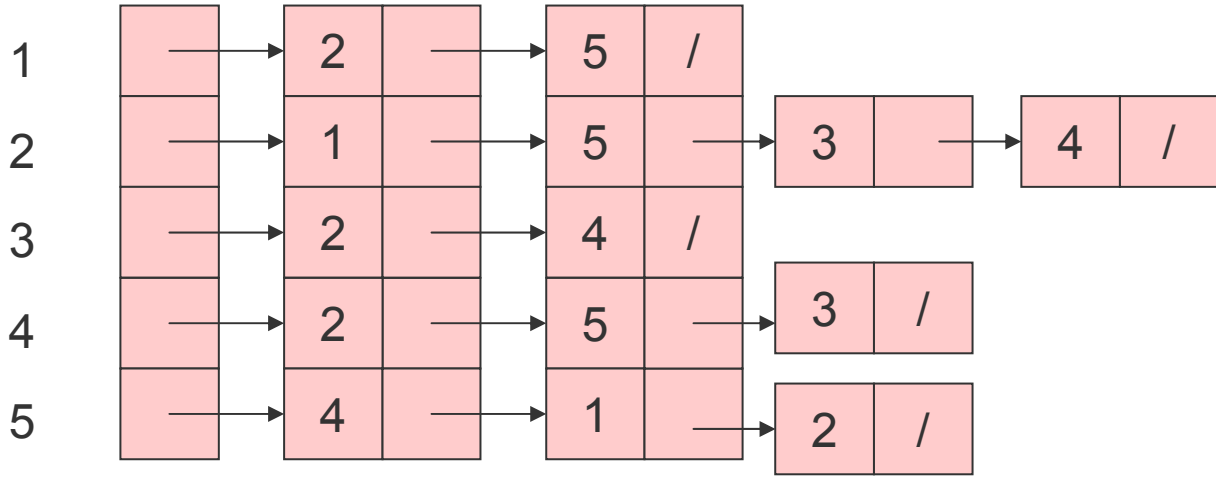
- Two standard ways
  - Adjacency-list representation
    - Space required $O(|E|)$
  - Adjacency-matrix representation
    - Space required $O(n^2)$.
- Depending on problems, both representations are useful.

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.

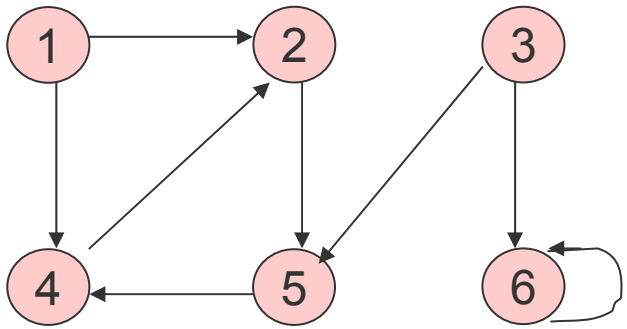


(a)

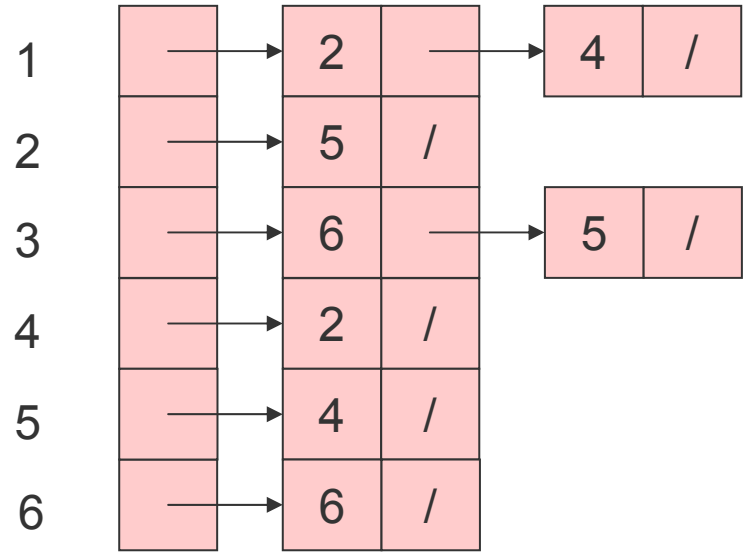


(b)

Adjacency-list representation for directed graph



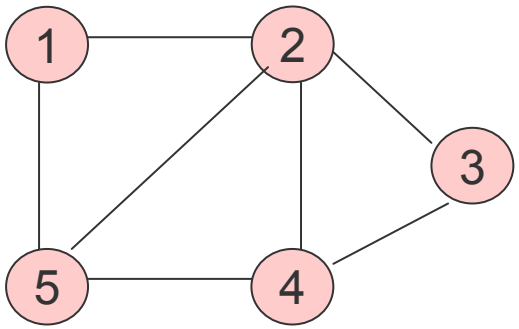
(a)



(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 $a_{ij}= 0$.

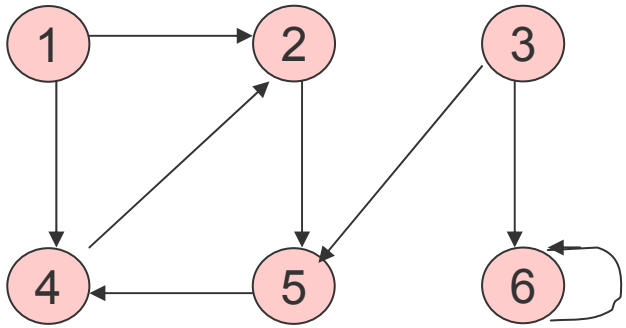


(a)

	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 node u.

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.
3. *If the adjacency-list for v is empty, DELETE the list.*

In Step 3: if all the adjacency-list is empty, stop. (Note we did 3).

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

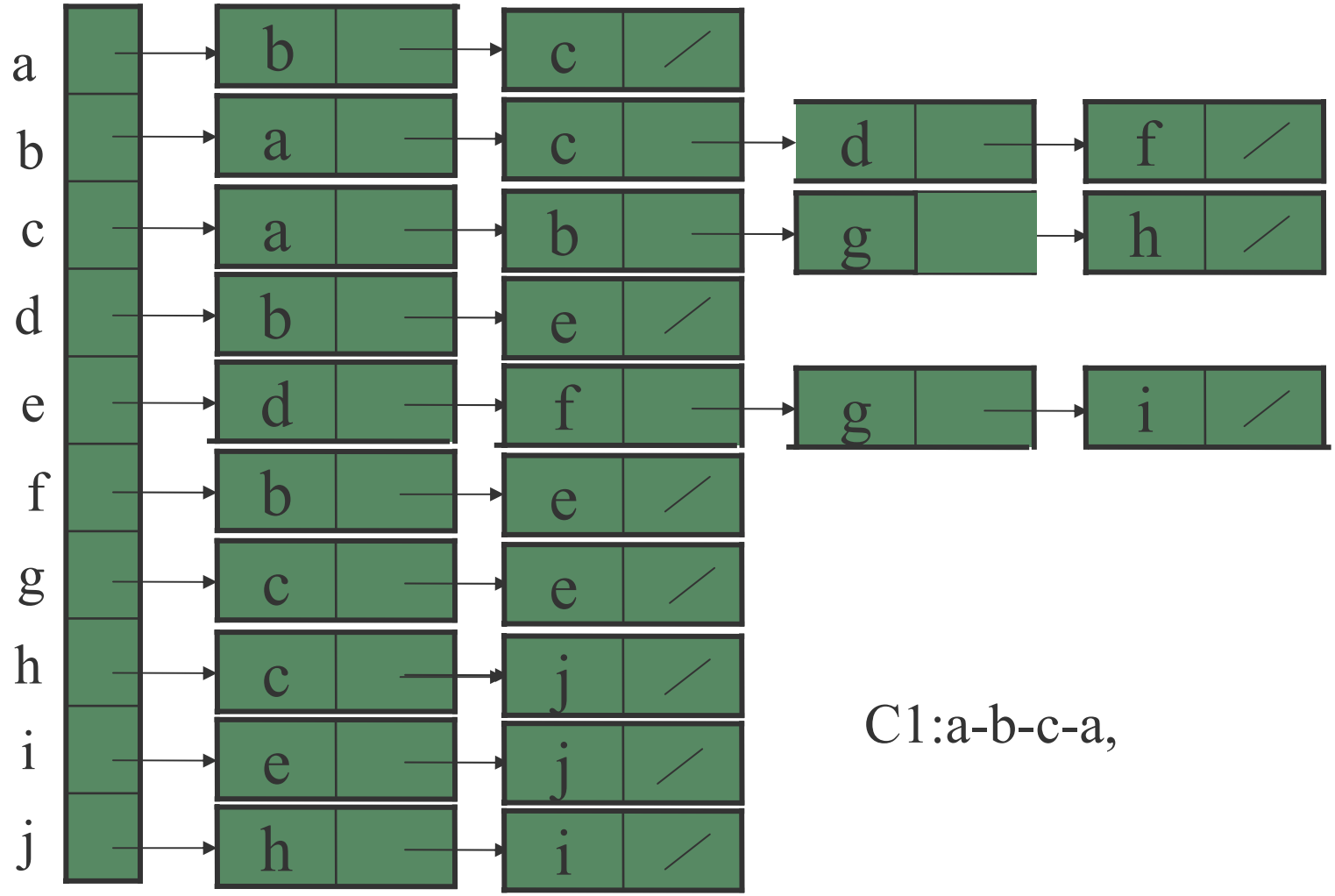
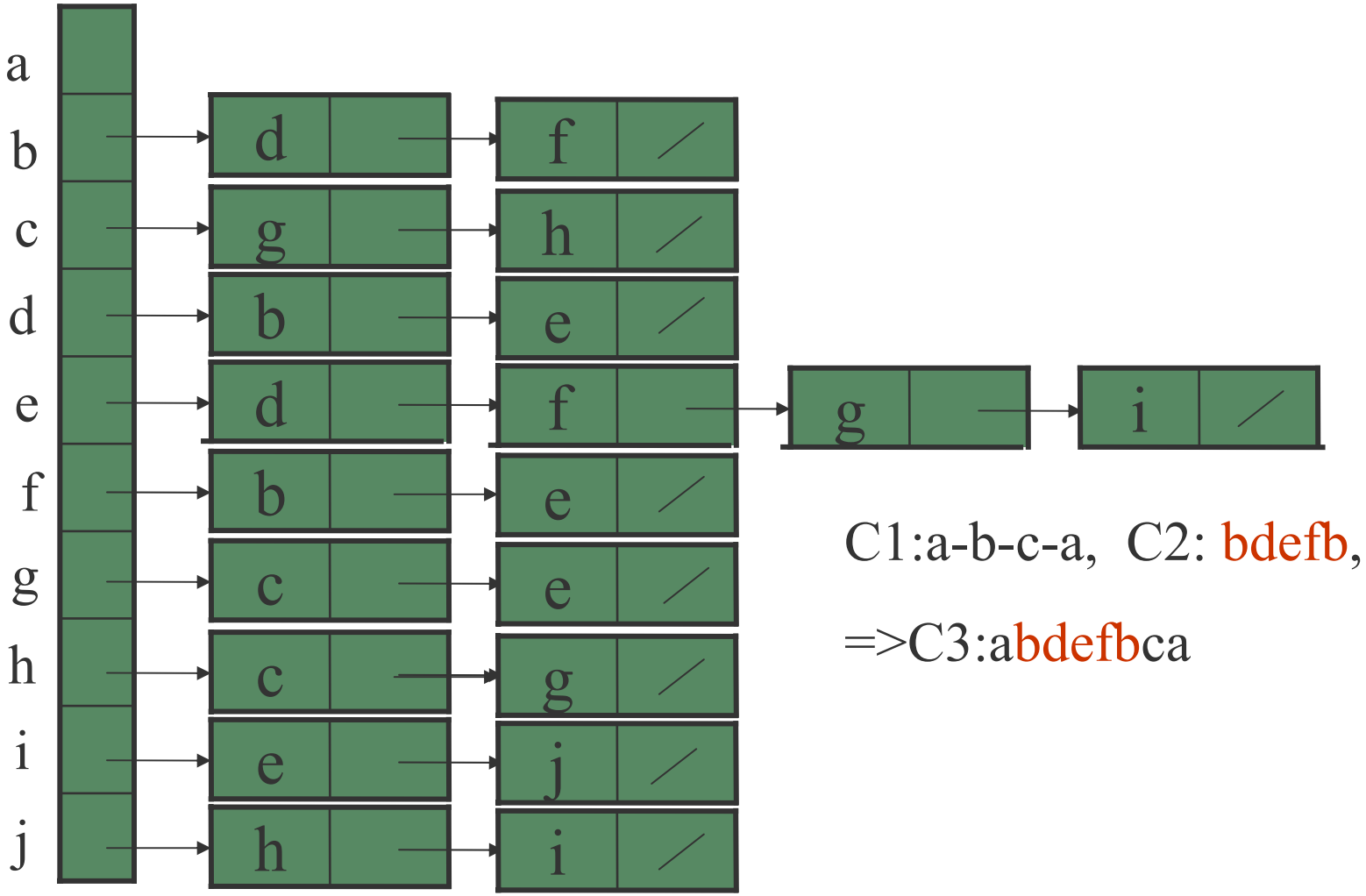
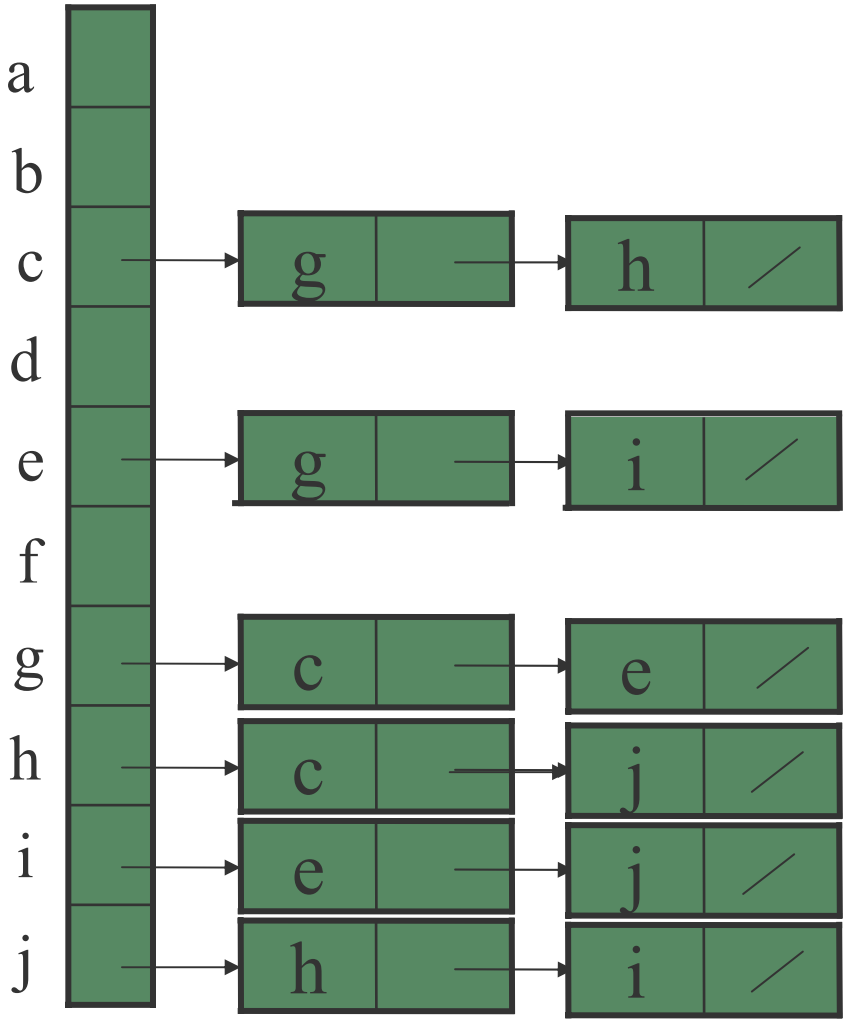


Figure1: The adjacency-list representation of Example 1(Slide 25)



After edge (a,b), (b, c) and (c, d) are used.



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

After edges (b, d) (d, e), (e, f), (f, b) are used.

Time complexity

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

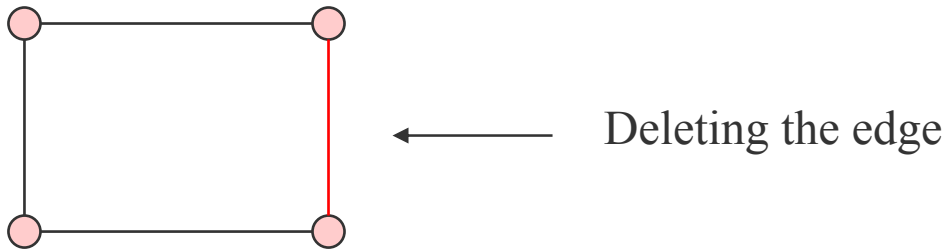
This can be done by using an adjacency list.

- The implementation is *tricky*
- Even O(|V||E|) algorithm is not trivial.
- Do not be disappointment 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.

Euler Path

- A path which contains all edges in a graph G is called an *Euler path* of G.
- Deleting an edge in a graph having an Euler Circuit, we obtain an Euler path in the new graph.



Corollary: A graph G=(V,E) which has an Euler path has 2 vertices of odd degree.

Proof of the Corollary

- Suppose that a graph which has an Euler path starting at u and ending at v, where $u \neq v$.
- Creating a new edge e joining u and v, we have an Euler circuit for the new graph $G'=(V, E \cup \{e\})$.

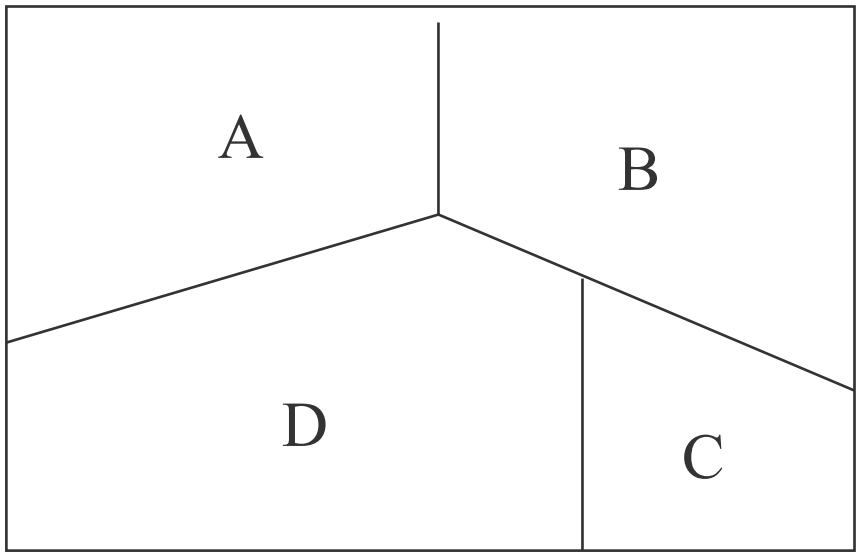
- From Theorem 1, all the vertices in G' have even degree. Remove e.

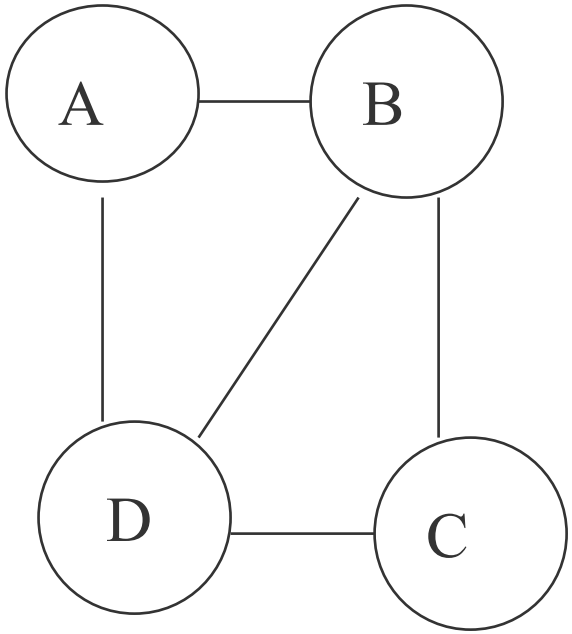
- Then u and v are the only vertices of odd degree in G.

(Nice argument, not required for exam.)

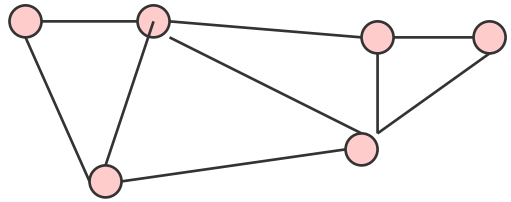
Application 1:

- 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.

Problem Solving: (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.

Special Arrangements:

- **We will have tutorials in Week 14 to makeup the tutorials in week 1 (if cancelled).**
  - To answer question for people from ALL groups.