Chapter 10
Graph connectivity

Discrete Structures for Computing on January 4, 2023

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai
Faculty of Computer Science and Engineering
University of Technology - VNUHCM
trtanh@hcmut.edu.vn

Acknowledgement

Some slides about Euler and Hamilton circuits are created by Chung Ki-hong and Hur Joon-seok from KAIST.

Course outcomes

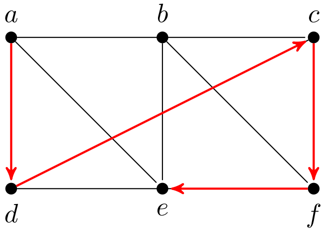
	Course learning outcomes
L.O.1	Understanding of logic and discrete structures
	L.O.1.1 – Describe definition of propositional and predicate logic L.O.1.2 – Define basic discrete structures: set, mapping, graphs
L.O.2	Represent and model practical problems with discrete structures
	L.O.2.1 – Logically describe some problems arising in Computing L.O.2.2 – Use proving methods: direct, contrapositive, induction L.O.2.3 – Explain problem modeling using discrete structures
L.O.3	Understanding of basic probability and random variables
	L.O.3.1 – Define basic probability theory L.O.3.2 – Explain discrete random variables
L.O.4	Compute quantities of discrete structures and probabilities
	L.O.4.1 – Operate (compute/ optimize) on discrete structures L.O.4.2 – Compute probabilities of various events, conditional ones, Bayes theorem

Contents

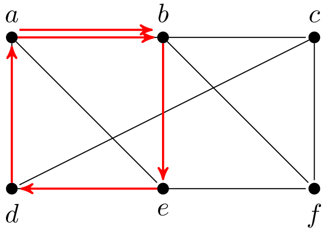
Path and Circuits

Definition (in undirected graph)

- Path (*đường đi*) of length n from u to v: a sequence of n edges $\{x_0, x_1\}, \{x_1, x_2\}, \ldots, \{x_{n-1}, x_n\}$, where $x_0 = u$ and $x_n = v$.
- A path is a circuit (*chu trình*) if it begins and ends at the same vertex, $u = v$.
- A path or circuit is simple (*đơn*) if it does not contain the same edge more than once.

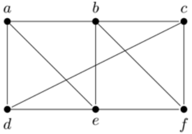


Simple path



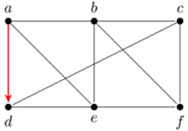
Not simple path

Paths and Circuits

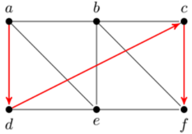


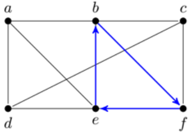
Original Graph

Simple path
(length 1)



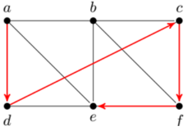
Simple path
(length 3)

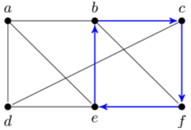




Simple Circuit
(length 3)

Simple path
(length 4)



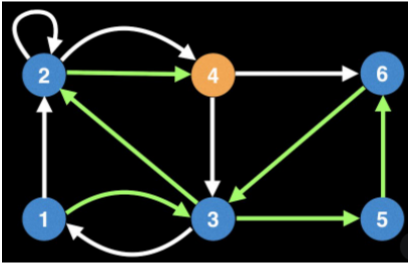


Simple Circuit
(length 4)

Path and Circuits

Definition (in directed graphs)

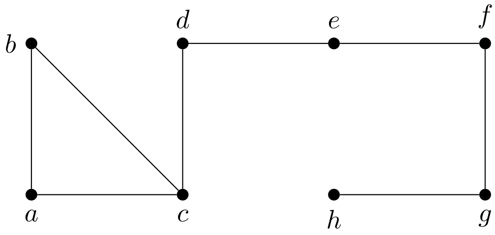
Path is a sequence of $(x_0, x_1), (x_1, x_2), \ldots, (x_{n-1}, x_n)$, where $x_0 = u$ and $x_n = v$.



Connectedness in Undirected Graphs

Definition

- An undirected graph is called connected (*liên thông*) if there is a path between every pair of distinct vertices of the graph.
- There is a simple path between every pair of distinct vertices of a connected undirected graph.

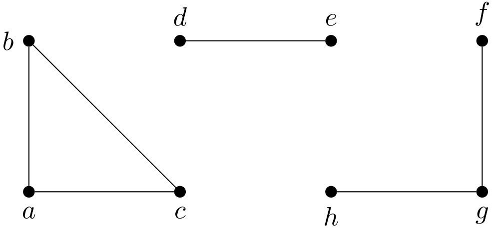


Connected graph

Connectedness in Undirected Graphs

Definition

- An undirected graph is called connected (*liên thông*) if there is a path between every pair of distinct vertices of the graph.
- There is a simple path between every pair of distinct vertices of a connected undirected graph.



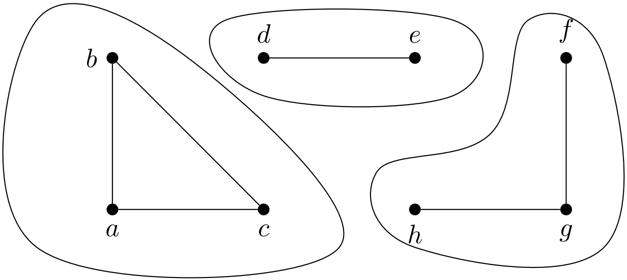
Disconnected graph

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Connectedness in Undirected Graphs

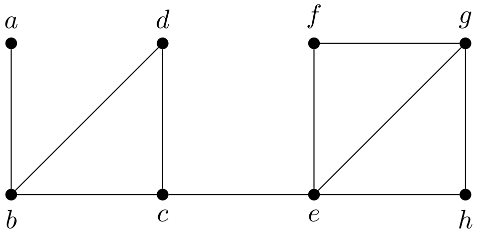
Definition

- An undirected graph is called connected (*liên thông*) if there is a path between every pair of distinct vertices of the graph.
- There is a simple path between every pair of distinct vertices of a connected undirected graph.



Connected components (*thành phần liên thông*)

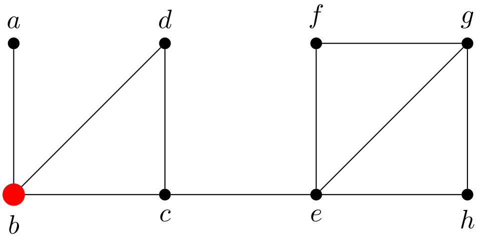
How Connected is a Graph?



Definition

- b is a cut vertex (*đỉnh cắt*) or articulation point (*điểm khớp*).

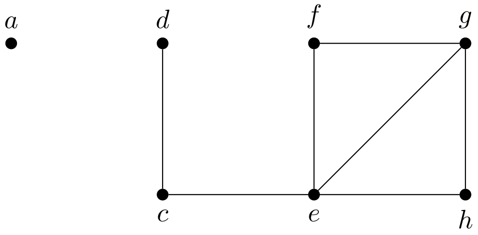
How Connected is a Graph?



Definition

- b is a cut vertex (*đỉnh cắt*) or articulation point (*điểm khớp*).

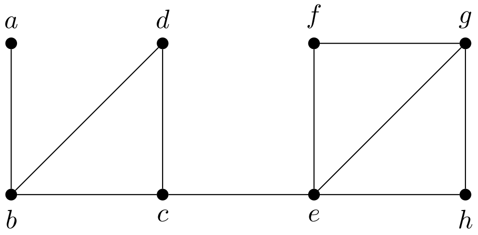
How Connected is a Graph?



Definition

- b is a cut vertex (*đỉnh cắt*) or articulation point (*điểm khớp*).

How Connected is a Graph?

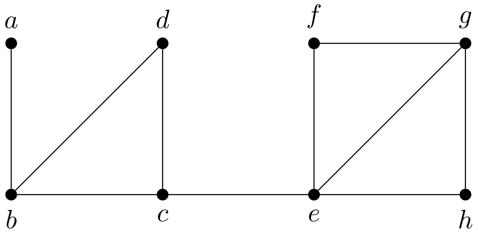


Definition

- b is a cut vertex (*đỉnh cắt*) or articulation point (*điểm khớp*). What else?

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

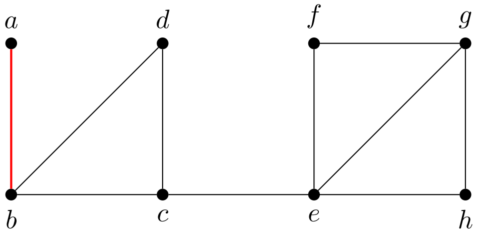
How Connected is a Graph?



Definition

- b is a cut vertex (*đỉnh cắt*) or articulation point (*điểm khớp*). What else?
- $\{a, b\}$ is a cut edge (*cạnh cắt*) or bridge (*cầu*).

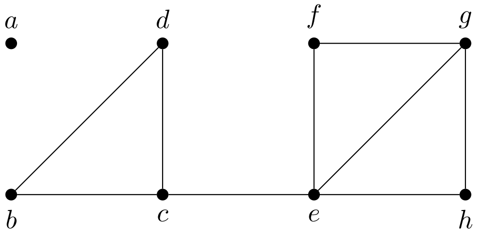
How Connected is a Graph?



Definition

- b is a cut vertex (*đỉnh cắt*) or articulation point (*điểm khớp*). What else?
- $\{a, b\}$ is a cut edge (*cạnh cắt*) or bridge (*cầu*).

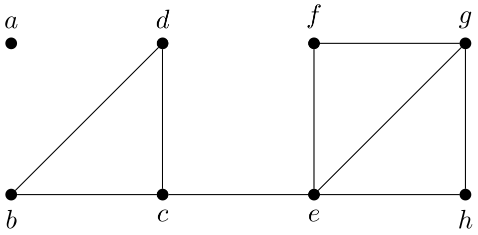
How Connected is a Graph?



Definition

- b is a cut vertex (*đỉnh cắt*) or articulation point (*điểm khớp*). What else?
- $\{a, b\}$ is a cut edge (*cạnh cắt*) or bridge (*cầu*).

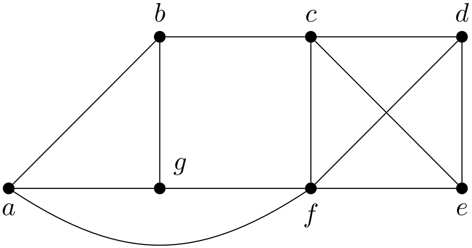
How Connected is a Graph?



Definition

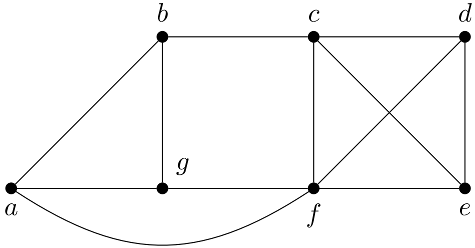
- b is a cut vertex (*đỉnh cắt*) or articulation point (*điểm khớp*). What else?
- $\{a, b\}$ is a cut edge (*cạnh cắt*) or bridge (*cầu*). What else?

How Connected is a Graph?



Definition

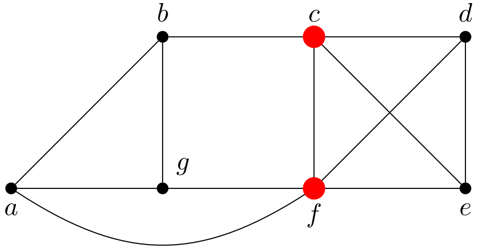
How Connected is a Graph?



Definition

- This graph doesn't have cut vertices: nonseparable graph (*đồ thị không thể phân tách*)

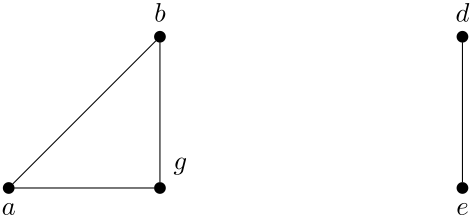
How Connected is a Graph?



Definition

- This graph doesn't have cut vertices: nonseparable graph (*đồ thị không thể phân tách*)

How Connected is a Graph?

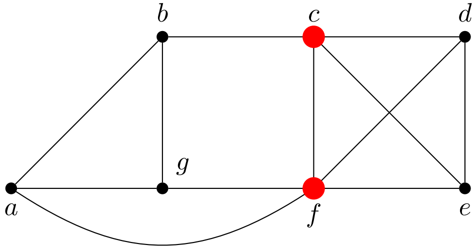


Definition

- This graph doesn't have cut vertices: nonseparable graph (*đồ thị không thể phân tách*)

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

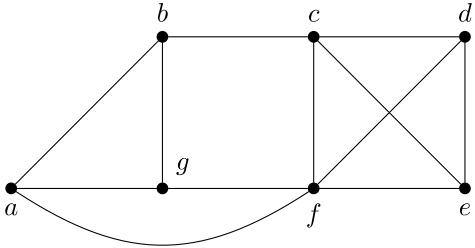
How Connected is a Graph?



Definition

- This graph doesn't have cut vertices: nonseparable graph (*đồ thị không thể phân tách*)
- The vertex cut is $\{c, f\}$, so the minimum number of vertices in a vertex cut, vertex connectivity (*liên thông đỉnh*) $\kappa(G) = 2$.

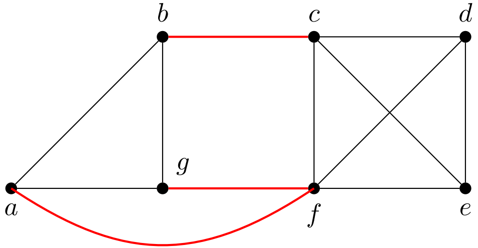
How Connected is a Graph?



Definition

- This graph doesn't have cut vertices: nonseparable graph (*đồ thị không thể phân tách*)
- The vertex cut is $\{c, f\}$, so the minimum number of vertices in a vertex cut, vertex connectivity (*liên thông đỉnh*) $\kappa(G) = 2$.

How Connected is a Graph?

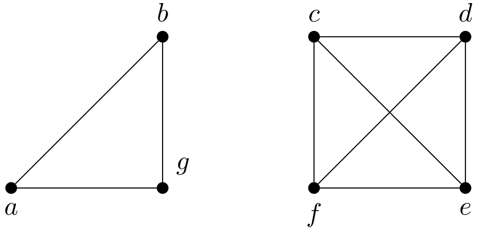


Definition

- This graph doesn't have cut vertices: nonseparable graph (*đồ thị không thể phân tách*)
- The vertex cut is $\{c, f\}$, so the minimum number of vertices in a vertex cut, vertex connectivity (*liên thông đỉnh*) $\kappa(G) = 2$.

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

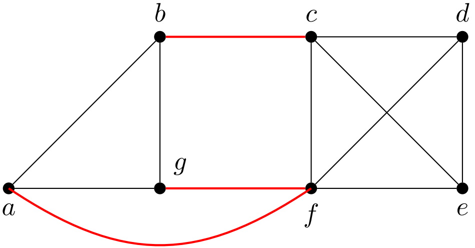
How Connected is a Graph?



Definition

- This graph doesn't have cut vertices: nonseparable graph (*đồ thị không thể phân tách*)
- The vertex cut is $\{c, f\}$, so the minimum number of vertices in a vertex cut, vertex connectivity (*liên thông đỉnh*) $\kappa(G) = 2$.

How Connected is a Graph?



Definition

- This graph doesn't have cut vertices: nonseparable graph (*đồ thị không thể phân tách*)
- The vertex cut is $\{c, f\}$, so the minimum number of vertices in a vertex cut, vertex connectivity (*liên thông đỉnh*) $\kappa(G) = 2$.
- The edge cut is $\{\{b, c\}, \{a, f\}, \{f, g\}\}$, the minimum number of edges in an edge cut, edge connectivity (*liên thông cạnh*) $\lambda(G) = 3$.

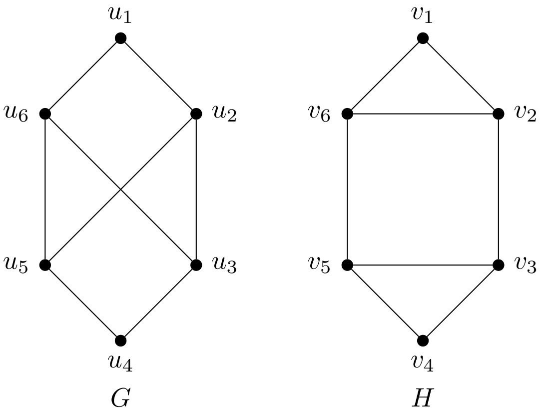
Applications of Vertex and Edge Connectivity

- Reliability of networks
  - Minimum number of routers that disconnect the network
  - Minimum number of fiber optic links that can be down to disconnect the network
- Highway network
  - Minimum number of intersections that can be closed
  - Minimum number of roads that can be closed

Applications

Example

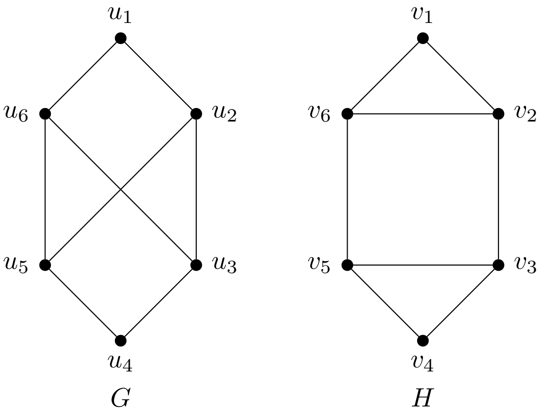
Determine whether the graphs below are isomorphic.



Applications

Example

Determine whether the graphs below are isomorphic.



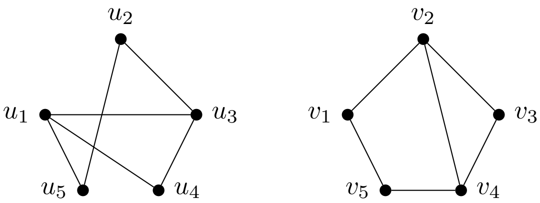
Solution

H has a simple circuit of length three, not G.

Applications

Example

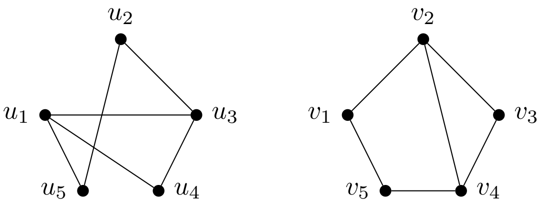
Determine whether the graphs below are isomorphic.



Applications

Example

Determine whether the graphs below are isomorphic.



Solution

Both graphs have the same vertices, edges, degrees, circuits. They may be isomorphic.
To find a possible isomorphism, we can follow paths that go through all vertices so that the corresponding vertices in the two graphs have the same degrees.

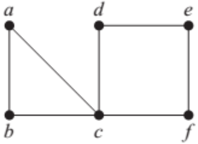
Exercise

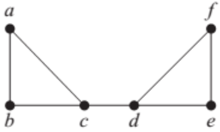
Find all the cut vertices, cut edges of the graphs

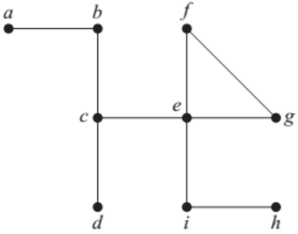
a) C_n, where $n \geq 3$

b) W_n where $n \geq 3$

c) $K_{m,n}$ where $m \geq 2, n \geq 2$

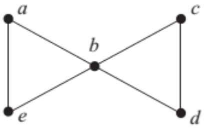


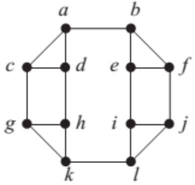




Exercise

For each of these graphs, find $\kappa(G), \lambda(G)$

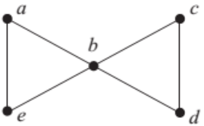
a) 

b) 

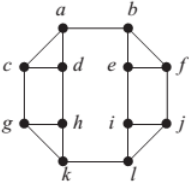
Exercise

For each of these graphs, find $\kappa(G), \lambda(G)$

a)



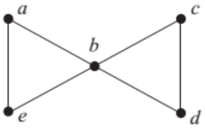
b)

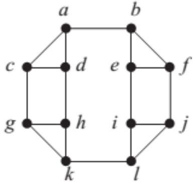


Construct a graph G with $\kappa(G) = 1, \lambda(G) = 2$, and $\min_{v \in V} deg(v) = 3$.

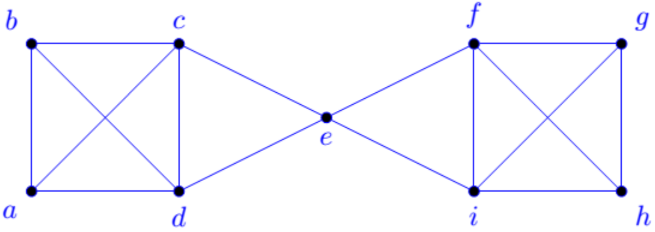
Exercise

For each of these graphs, find $\kappa(G), \lambda(G)$

a) 

b) 

Construct a graph G with $\kappa(G) = 1, \lambda(G) = 2$, and $\min_{v \in V} deg(v) = 3$.

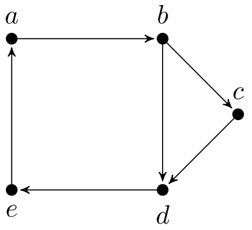


Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

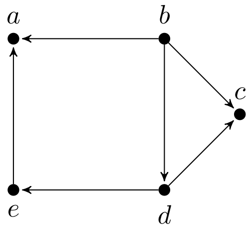
Connectedness in Directed Graphs

Definition

- An directed graph is strongly connected (*liên thông mạnh*) if there is a path between any two vertices in the graph (for both directions).
- An directed graph is weakly connected (*liên thông yếu*) if there is a path between any two vertices in the underlying undirected graph.

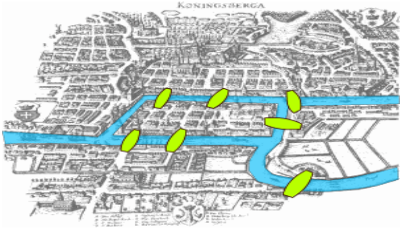


Strongly connected



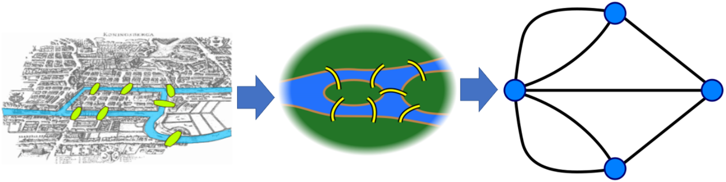
Weakly connected

The Famous Problem of Seven Bridges of Königsberg



- Is there a route that a person crosses all the seven bridges once?

Euler Solution

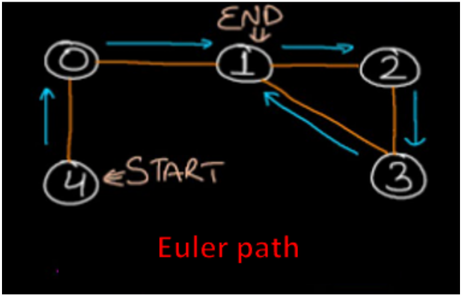


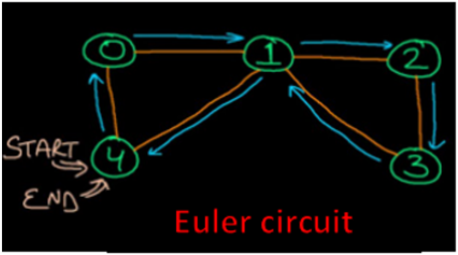
- Euler gave the solution: It is not possible to cross all the bridges exactly once.

What is Euler Path and Circuit?

- **Euler Path** (*đường đi Euler*) is a path in the graph that passes each edge only once.
  The problem of Seven Bridges of Königsberg can be also stated: Does Euler Path exist in the graph?
- **Euler Circuit** (*chu trình Euler*) is a path in the graph that passes each edge only once and return back to its original position.
  From Definition, Euler Circuit is a subset of Euler Path.

Examples of Euler Path and Circuit



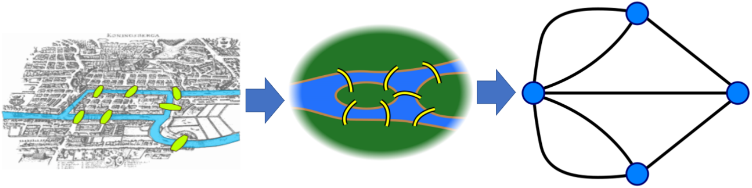


Conditions for Existence

In a connected multigraph,

- Euler Circuit existence: **no odd-degree nodes exist** in the graph.
- Euler Path existence: **2 or no odd-degree nodes exist** in the graph.

Back to the Seven Bridges Problem



- Four vertices of odd degree
- No Euler circuit $\rightarrow$ cannot cross each bridge exactly once, and return to starting point
- No Euler path, either

Searching Euler Circuits and Paths – Fleury's Algorithm

- Choose a random vertex (if circuit) or an odd degree vertex (if path)
- Pick an edge joined to another vertex so that it is not a cut edge unless there is no alternative.
- Remove the chosen edge. The above procedure is repeated until all edges are covered.

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Searching Euler Circuits and Paths – Hierholzer's Algorithm

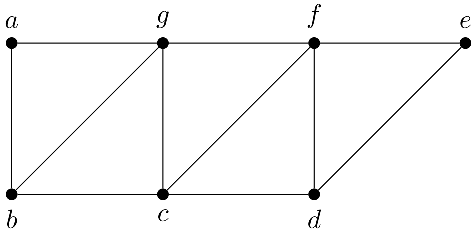
- Choose a starting vertex and find a circuit
- As long as there exists a vertex v that belongs to the current tour but that has adjacent edges not part of the tour, start another circuit from v

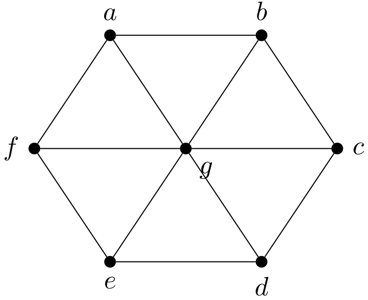
More efficient algorithm, $O(n)$.

Exercise

Example

Are these following graph Euler path (circuit)? If yes, find one.

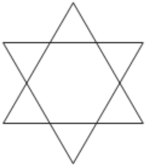




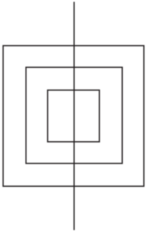
Exercise

Determine whether the picture shown can be drawn with a pencil in a continuous motion without lifting the pencil or retracing part of the picture.

a)

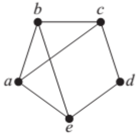


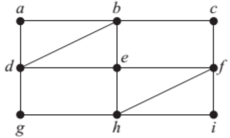
b)

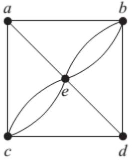


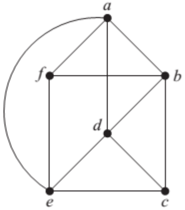
Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

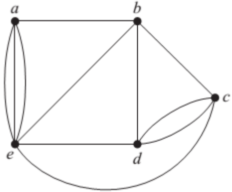
Exercise - Euler path & circuit

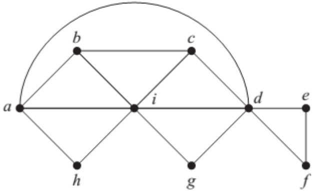
a) 

b) 

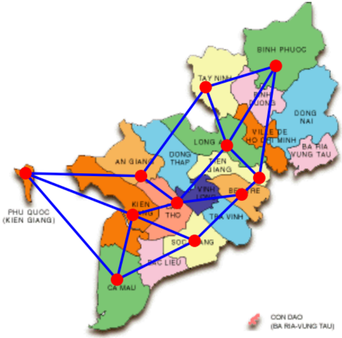
c) 

d) 

e) 

f) 

Traveling Salesman Problem



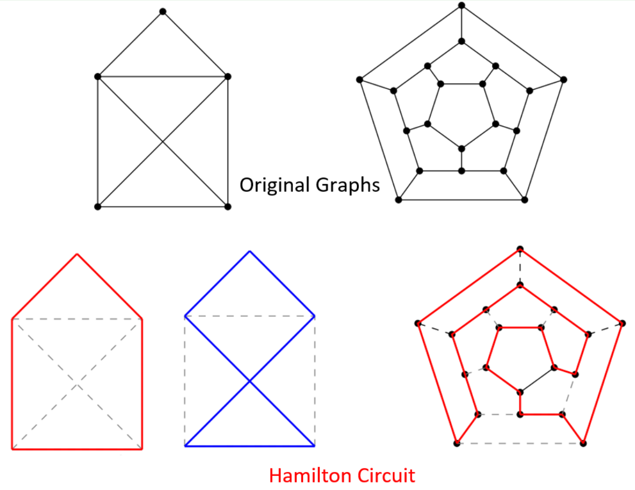
Is there the possible tour that visits each city exactly once?

What Is A Hamilton Circuit?

Definition

The circuit that visit each vertex in a graph once.

Example



Rules of Hamilton Circuits

$\deg(v) = 2$ for $\forall v$ in Hamilton circuit!

Rules of Hamilton Circuits

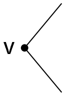
$\deg(v) = 2$ for $\forall v$ in Hamilton circuit!

Rule 1 if $\deg(v) = 2$, both edge must be used.

Rules of Hamilton Circuits

$\deg(v) = 2$ for $\forall v$ in Hamilton circuit!

Rule 1 if $\deg(v) = 2$, both edge must be used.

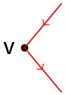


Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Rules of Hamilton Circuits

$\deg(v) = 2$ for $\forall v$ in Hamilton circuit!

Rule 1 if $\deg(v) = 2$, both edge must be used.

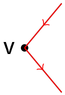


Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Rules of Hamilton Circuits

$$\deg(v) = 2 \text{ for } \forall v \text{ in Hamilton circuit!}$$

Rule 1 if $\deg(v) = 2$, both edge must be used.

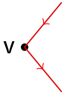


Rule 2 No subcircuit (*chu trình con*) can be formed.

Rules of Hamilton Circuits

$\deg(v) = 2$ for $\forall v$ in Hamilton circuit!

Rule 1 if $\deg(v) = 2$, both edge must be used.



Rule 2 No subcircuit (*chu trình con*) can be formed.

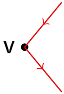
Rule 3 Once two edges at a vertex v is determined, all other edges incident at v must be removed.

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Rules of Hamilton Circuits

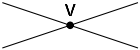
$$\deg(v) = 2 \text{ for } \forall v \text{ in Hamilton circuit!}$$

Rule 1 if $\deg(v) = 2$, both edge must be used.



Rule 2 No subcircuit (*chu trình con*) can be formed.

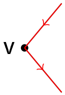
Rule 3 Once two edges at a vertex v is determined, all other edges incident at v must be removed.



Rules of Hamilton Circuits

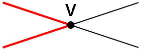
$\deg(v) = 2$ for $\forall v$ in Hamilton circuit!

Rule 1 if $\deg(v) = 2$, both edge must be used.



Rule 2 No subcircuit (*chu trình con*) can be formed.

Rule 3 Once two edges at a vertex v is determined, all other edges incident at v must be removed.

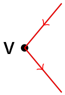


Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Rules of Hamilton Circuits

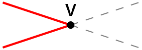
$$\deg(v) = 2 \text{ for } \forall v \text{ in Hamilton circuit!}$$

Rule 1 if $\deg(v) = 2$, both edge must be used.

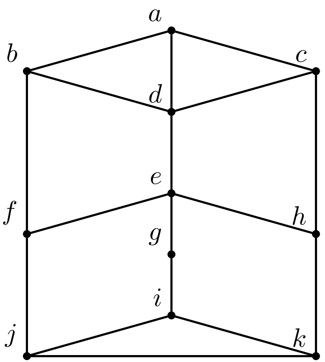


Rule 2 No subcircuit (*chu trình con*) can be formed.

Rule 3 Once two edges at a vertex v is determined, all other edges incident at v must be removed.



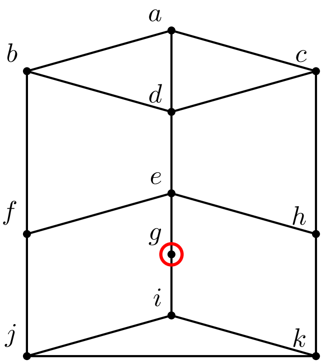
Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

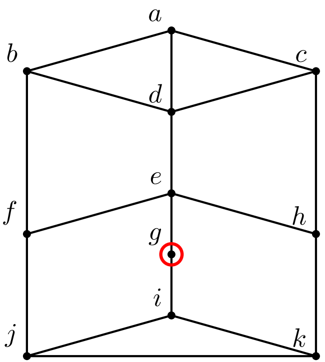
Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

Finding Hamilton Circuits



Vertices : cities

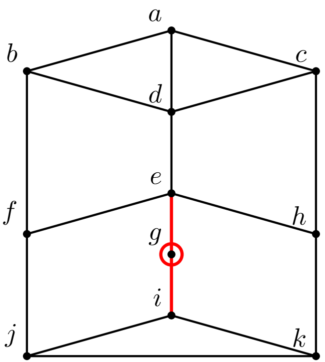
Edges : possible routes

Rule 1

$\deg(v) = 2$

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Finding Hamilton Circuits



Vertices : cities

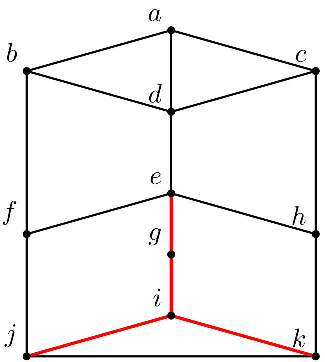
Edges : possible routes

Rule 1

$\deg(v) = 2$

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Finding Hamilton Circuits



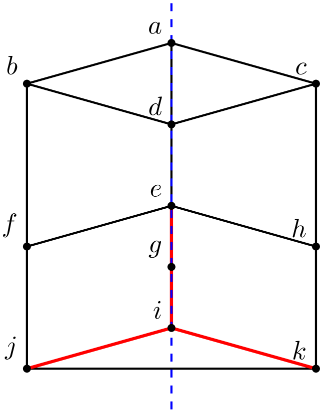
Vertices : cities

Edges : possible routes

Rule 1

$\deg(v) = 2$

Finding Hamilton Circuits



Vertices : cities

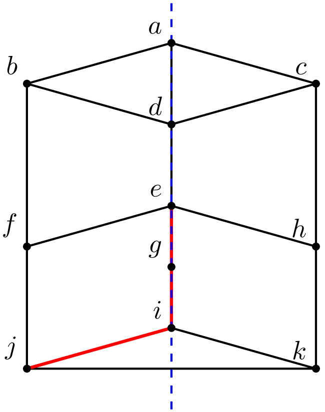
Edges : possible routes

Rule 1

$\deg(v) = 2$

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Finding Hamilton Circuits



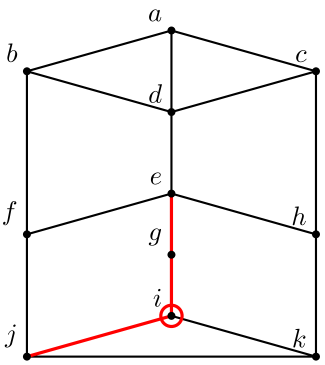
Vertices : cities

Edges : possible routes

Rule 1

$\deg(v) = 2$

Finding Hamilton Circuits



Vertices : cities

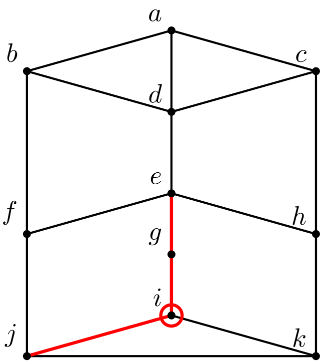
Edges : possible routes

Rule 1

$\deg(v) = 2$

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

Rule 1

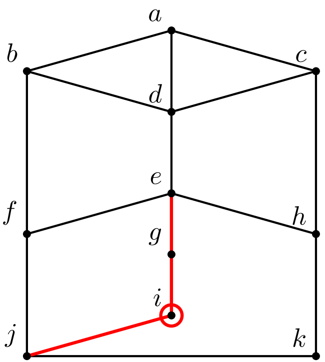
$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

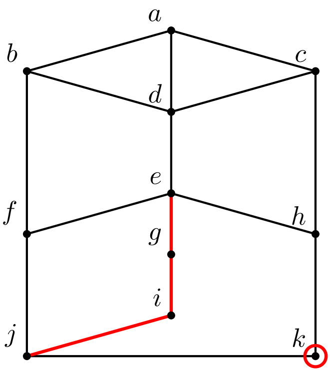
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

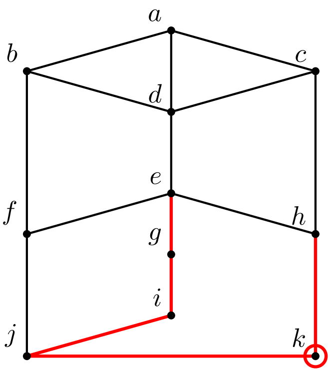
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



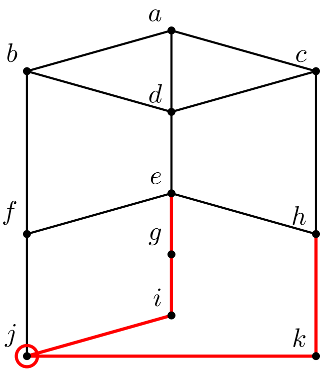
Vertices : cities

Edges : possible routes

Rule 1
$\deg(v) = 2$

Rule 3
Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

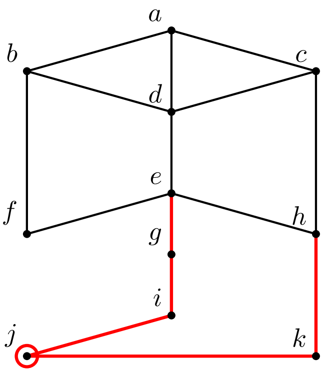
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

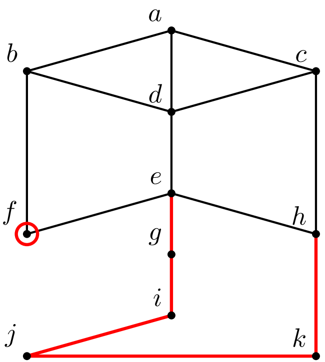
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



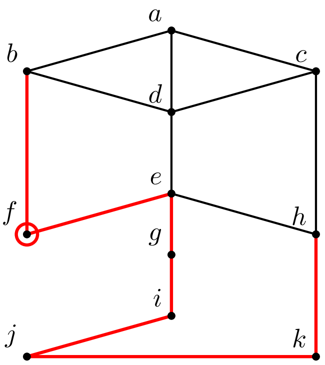
Vertices : cities

Edges : possible routes

Rule 1
$\deg(v) = 2$

Rule 3
Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

Rule 1

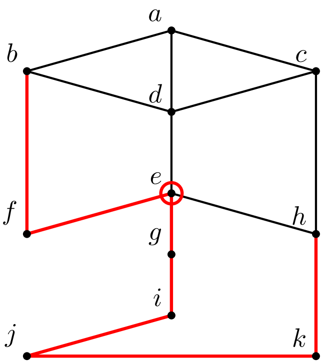
$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

Rule 1

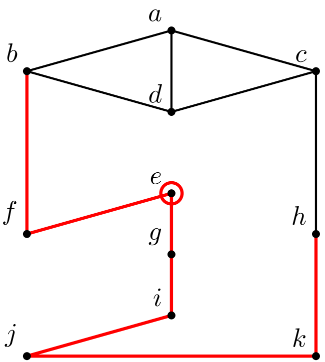
$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

Rule 1

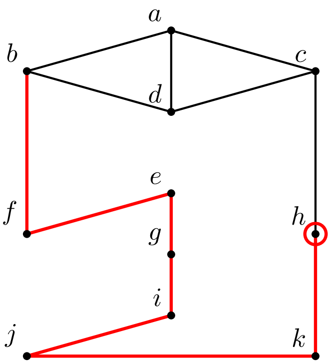
$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Finding Hamilton Circuits



Vertices : cities

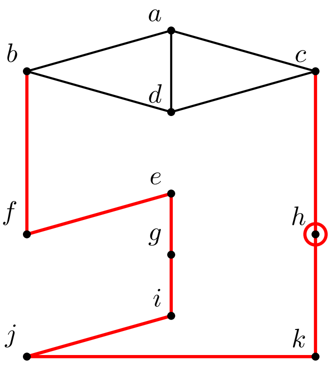
Edges : possible routes

Rule 1
$\deg(v) = 2$

Rule 3
Once two edges are determined, other edges must be removed

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Finding Hamilton Circuits



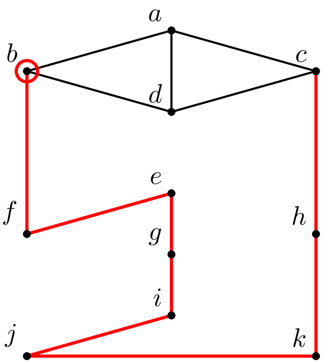
Vertices : cities

Edges : possible routes

Rule 1
$\deg(v) = 2$

Rule 3
Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

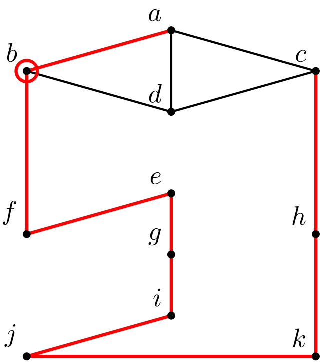
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

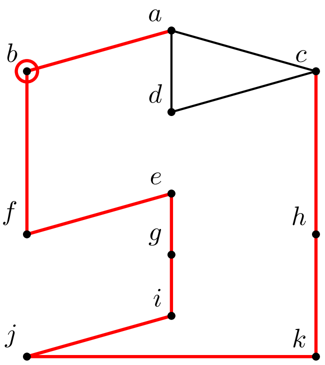
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

Rule 1

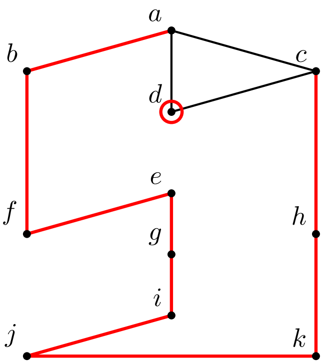
$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

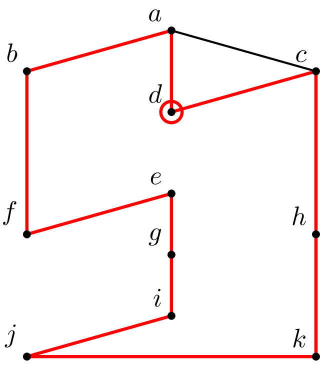
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

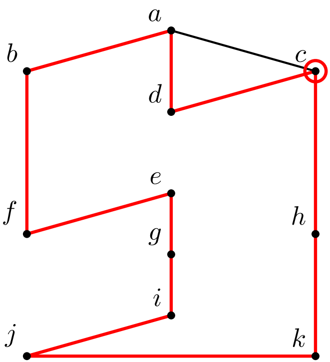
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

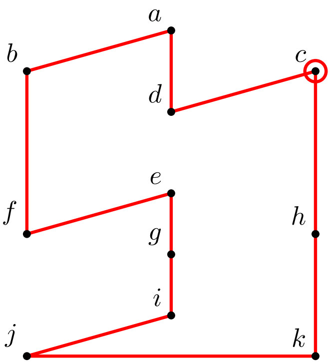
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



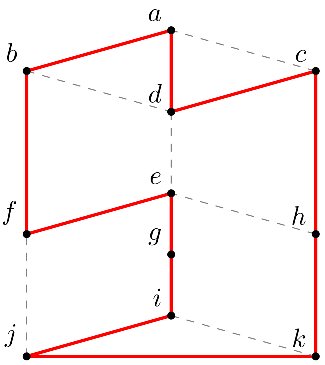
Vertices : cities

Edges : possible routes

Rule 1
$\deg(v) = 2$

Rule 3
Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



We get Hamilton circuit!

Vertices : cities

Edges : possible routes

Rule 1

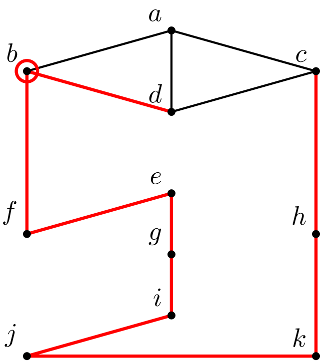
$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Finding Hamilton Circuits



Vertices : cities

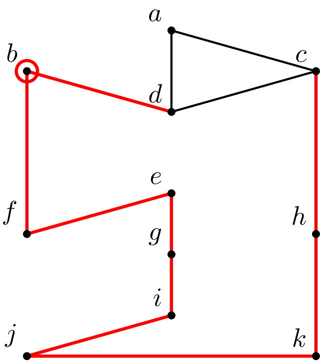
Edges : possible routes

Rule 1
$\deg(v) = 2$

Rule 3
Once two edges are determined, other edges must be removed

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

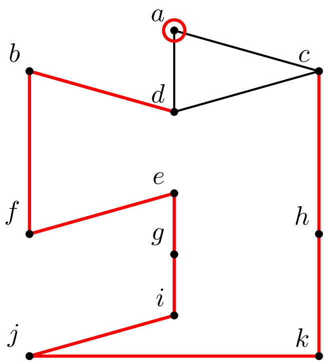
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

Rule 1

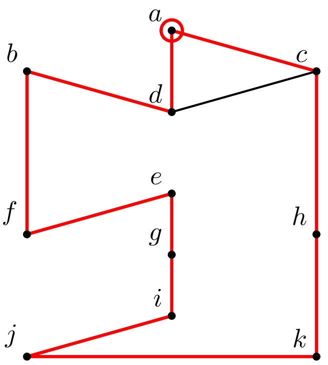
$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Finding Hamilton Circuits



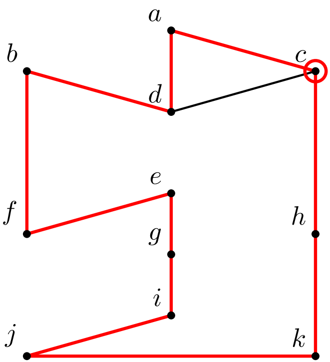
Vertices : cities

Edges : possible routes

Rule 1
$\deg(v) = 2$

Rule 3
Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

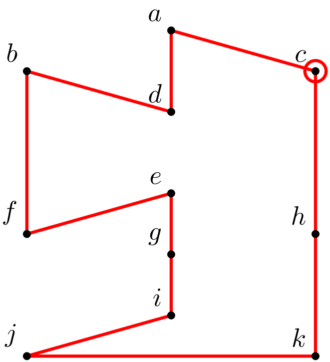
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

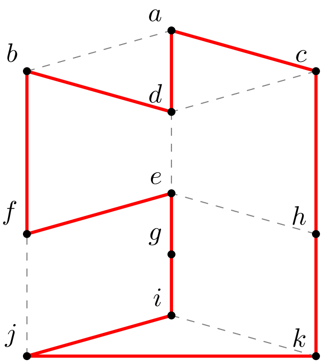
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



We get Hamilton circuit!

Vertices : cities

Edges : possible routes

Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Existence of Hamilton Circuit

Hamilton circuit does not exist for all graph. But, there is no specific way to find whether Hamilton circuit exists or not.

Simple check by rules of Hamilton circuit

Existence of Hamilton Circuit

Hamilton circuit does not exist for all graph. But, there is no specific way to find whether Hamilton circuit exists or not.

Simple check by rules of Hamilton circuit

Existence of Hamilton Circuit

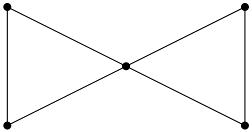
Hamilton circuit does not exist for all graph. But, there is no specific way to find whether Hamilton circuit exists or not.

Simple check by rules of Hamilton circuit

Existence of Hamilton Circuit

Hamilton circuit does not exist for all graph. But, there is no specific way to find whether Hamilton circuit exists or not.

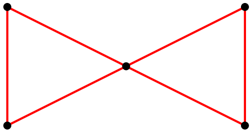
Simple check by rules of Hamilton circuit



Existence of Hamilton Circuit

Hamilton circuit does not exist for all graph. But, there is no specific way to find whether Hamilton circuit exists or not.

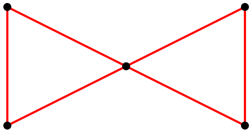
Simple check by rules of Hamilton circuit



Existence of Hamilton Circuit

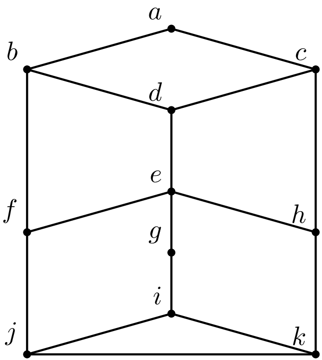
Hamilton circuit does not exist for all graph. But, there is no specific way to find whether Hamilton circuit exists or not.

Simple check by rules of Hamilton circuit

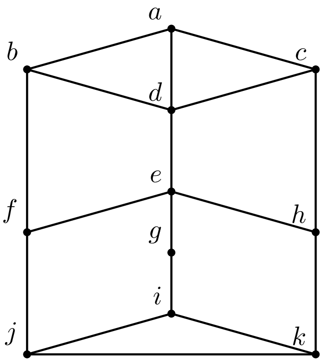


Violates Rule 2! (No subcircuit)

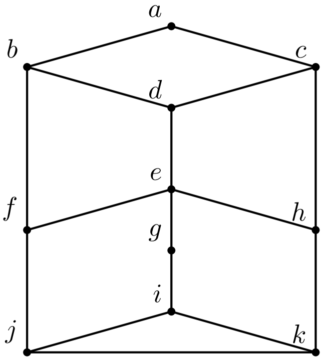
We can verify nonexistence of the graph during find Hamilton circuit.



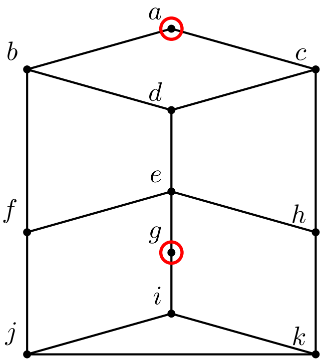
We can verify nonexistence of the graph during find Hamilton circuit.



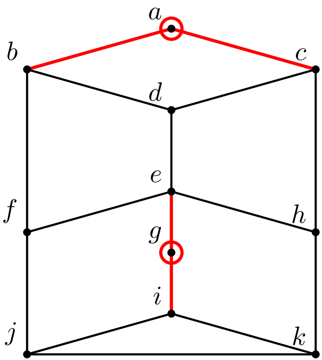
We can verify nonexistence of the graph during find Hamilton circuit.



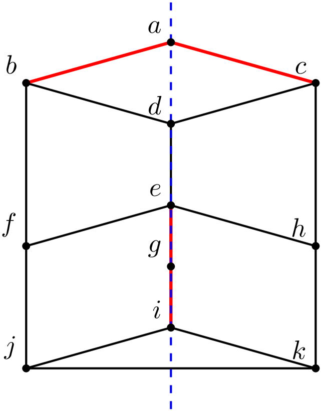
We can verify nonexistence of the graph during find Hamilton circuit.



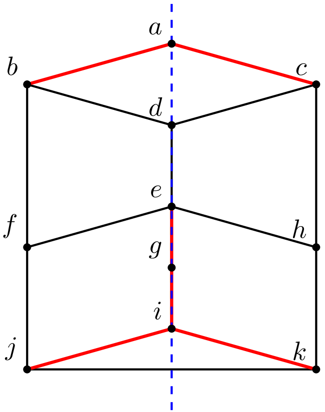
We can verify nonexistence of the graph during find Hamilton circuit.



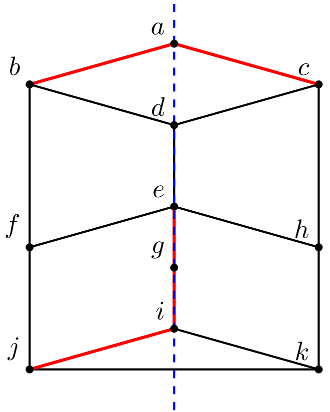
We can verify nonexistence of the graph during find Hamilton circuit.



We can verify nonexistence of the graph during find Hamilton circuit.

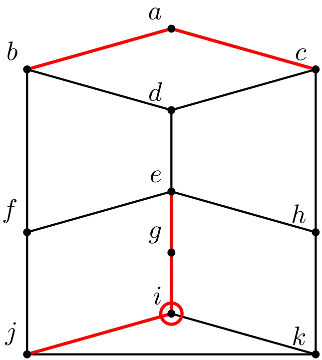


We can verify nonexistence of the graph during find Hamilton circuit.

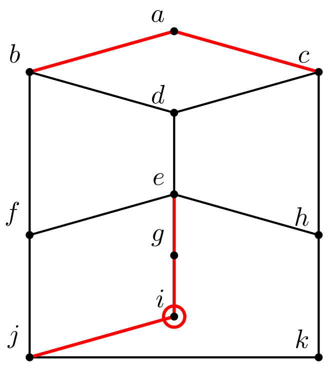


Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

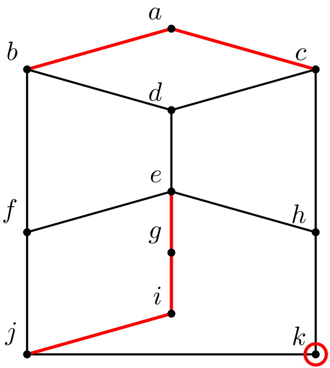
We can verify nonexistence of the graph during find Hamilton circuit.



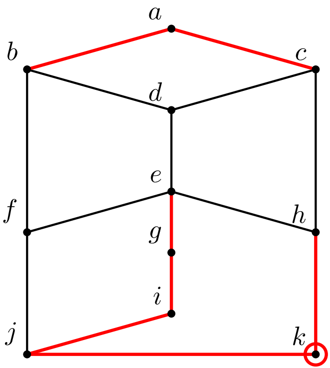
We can verify nonexistence of the graph during find Hamilton circuit.



We can verify nonexistence of the graph during find Hamilton circuit.

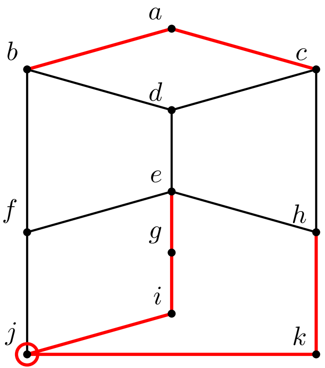


We can verify nonexistence of the graph during find Hamilton circuit.

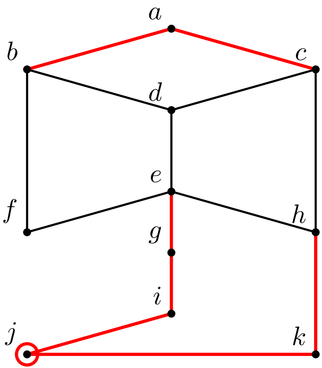


Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

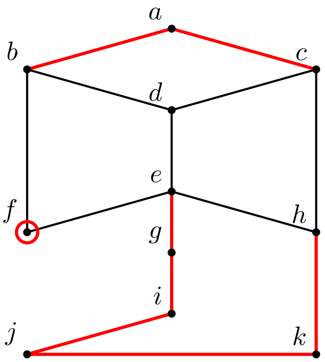
We can verify nonexistence of the graph during find Hamilton circuit.



We can verify nonexistence of the graph during find Hamilton circuit.

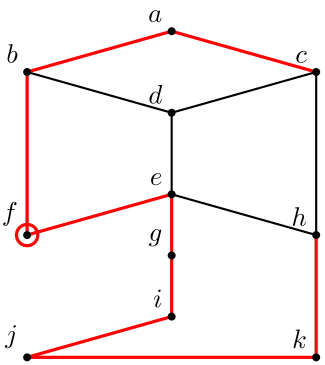


We can verify nonexistence of the graph during find Hamilton circuit.

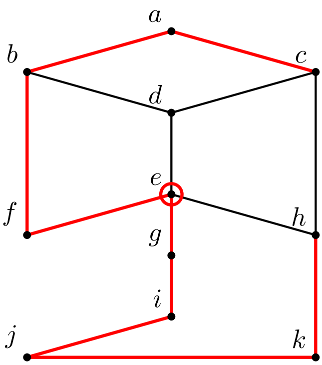


Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

We can verify nonexistence of the graph during find Hamilton circuit.

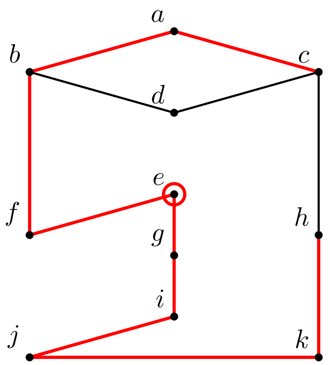


We can verify nonexistence of the graph during find Hamilton circuit.

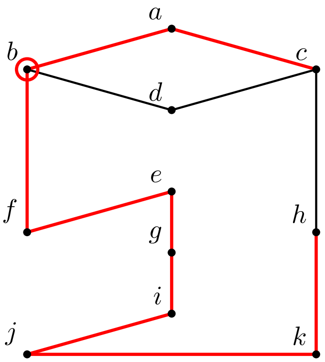


Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

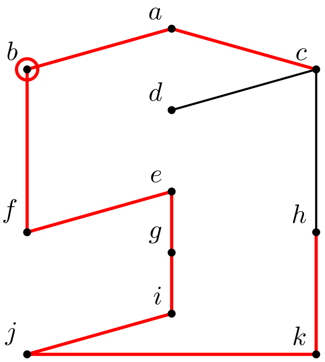
We can verify nonexistence of the graph during find Hamilton circuit.



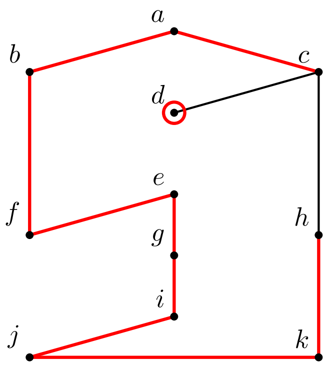
We can verify nonexistence of the graph during find Hamilton circuit.



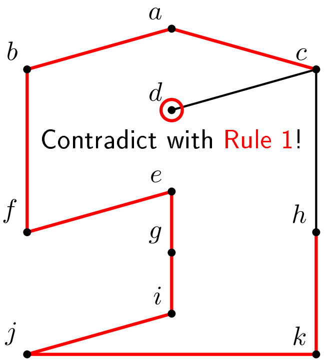
We can verify nonexistence of the graph during find Hamilton circuit.



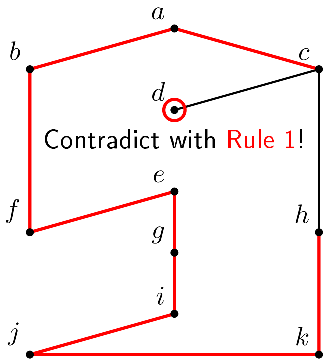
We can verify nonexistence of the graph during find Hamilton circuit.



We can verify nonexistence of the graph during find Hamilton circuit.



We can verify nonexistence of the graph during find Hamilton circuit.



Hamilton circuit doesn't exist!

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Application – Gray Code

Definition

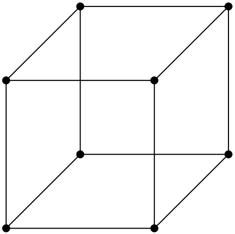
The binary sequence that express consecutive numbers by differing just one position of sequence.

Decimal number		Binary number	Gray code
1	$=$	001	000
2	$=$	010	100
3	$=$	011	110
4	$=$	100	010
5	$=$	101	011
$\vdots$		$\vdots$	$\vdots$

Used at digital communication for reduce the effect of noise; it prevents serious changes of information by noise.

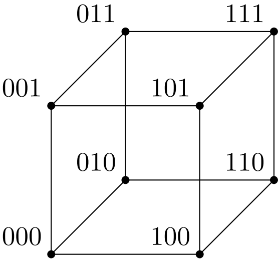
Gray Code

n-digit gray code can be generated by finding Hamilton circuits of n-dimensional hypercube!
Consider the case $n = 3$.



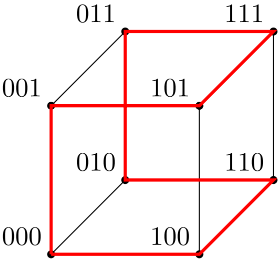
Gray Code

n-digit gray code can be generated by finding Hamilton circuits of n-dimensional hypercube!
Consider the case $n = 3$.



Gray Code

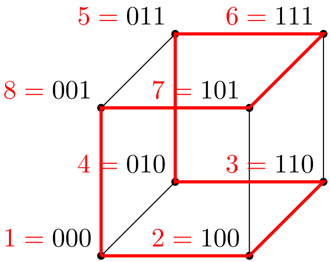
n-digit gray code can be generated by finding Hamilton circuits of n-dimensional hypercube!
Consider the case $n = 3$.



Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

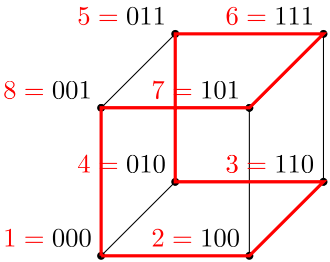
Gray Code

n-digit gray code can be generated by finding Hamilton circuits of n-dimensional hypercube!
Consider the case $n = 3$.



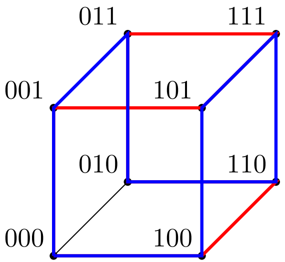
Gray Code

n-digit gray code can be generated by finding Hamilton circuits of n-dimensional hypercube!
Consider the case $n = 3$.



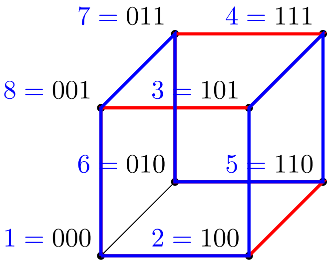
Gray Code

n-digit gray code can be generated by finding Hamilton circuits of n-dimensional hypercube!
Consider the case $n = 3$.



Gray Code

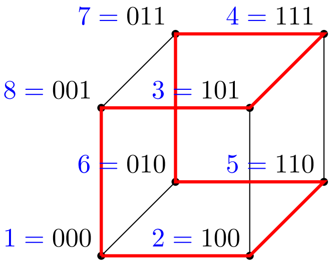
n-digit gray code can be generated by finding Hamilton circuits of n-dimensional hypercube!
Consider the case $n = 3$.



Gray Code

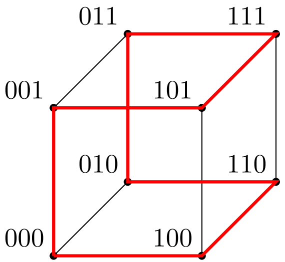
n-digit gray code can be generated by finding Hamilton circuits of n-dimensional hypercube!

Consider the case $n = 3$.



Gray Code

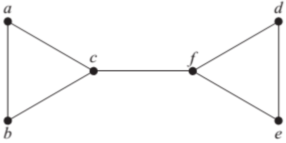
n-digit gray code can be generated by finding Hamilton circuits of n-dimensional hypercube!
Consider the case $n = 3$.



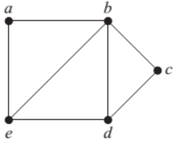
Coordinate of each vertex is 3-digit binary sequences.
Coordinates of adjacent vertices differ in just on place.
Hamilton circuits of a cubic graph makes the order of binary sequences!

Exercise - Hamilton path & circuit

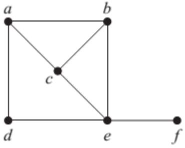
a)



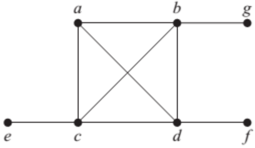
b)



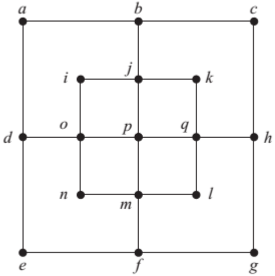
c)



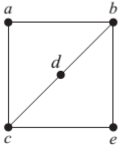
d)



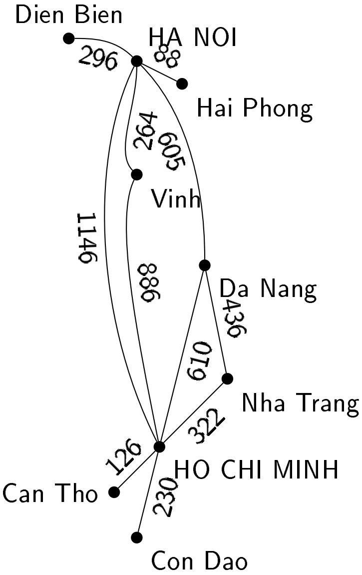
e)



f)



Weighted Graphs



Problem

The problem is also sometimes called the single-pair shortest path problem, to distinguish it from the following generalizations:

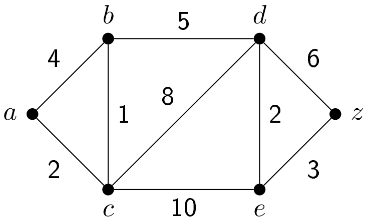
- The single-source shortest path problem, in which we have to find shortest paths from a source vertex v to all other vertices in the graph.
- The single-destination shortest path problem, in which we have to find shortest paths from all vertices in the graph to a single destination vertex v. This can be reduced to the single-source shortest path problem by reversing the edges in the graph.
- The all-pairs shortest path problem, in which we have to find shortest paths between every pair of vertices v, v' in the graph.

These generalizations have significantly more efficient algorithms than the simplistic approach of running a single-pair shortest path algorithm on all relevant pairs of vertices.

Dijkstra's Algorithm

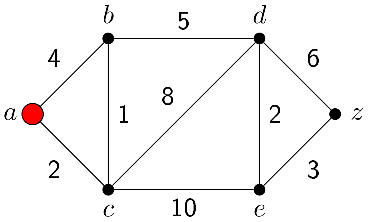
```
procedure Dijkstra(G,a)
// Initialization Step
  forall vertices v
     Label[v] := ∞
     Prev[v] := -1
  endfor
  Label(a) := 0 // a is the source node
  S := ∅

// Iteration Step
  while z ∉ S
     u := a vertex not in S with minimal Label
     S := S ∪ {u}
     forall vertices v not in S
       if (Label[u] + Wt(u,v)) < Label(v)
         then begin
                Label[v] := Label[u] + Wt(u,v)
                Pred[v] := u
              end
  endwhile
```

Example



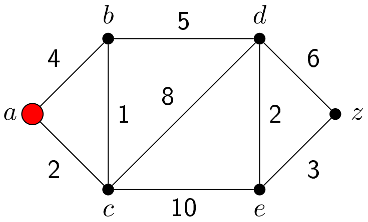
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞

Example



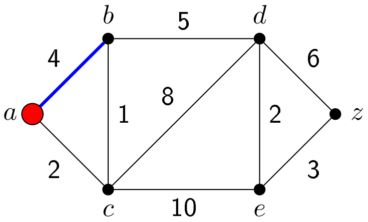
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞

Example



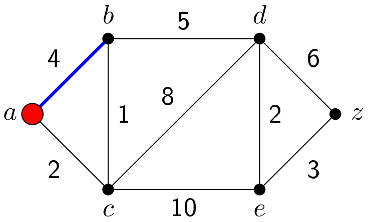
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0					

Example



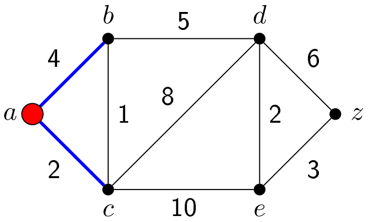
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0					

Example



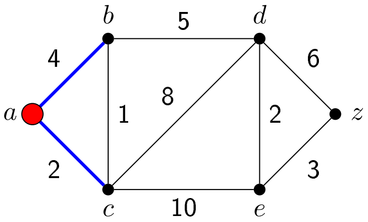
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4				

Example



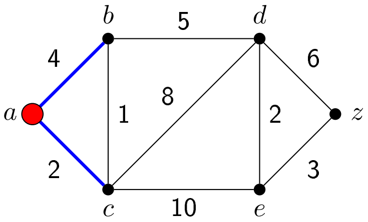
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4				

Example



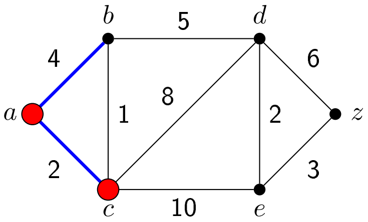
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	2			

Example



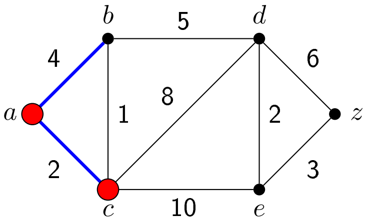
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	2	∞	∞	∞

Example



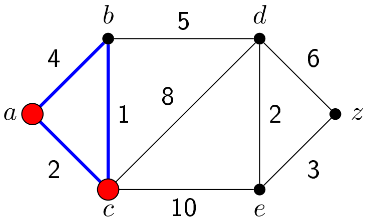
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞

Example



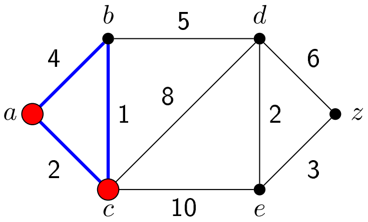
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0		2			

Example



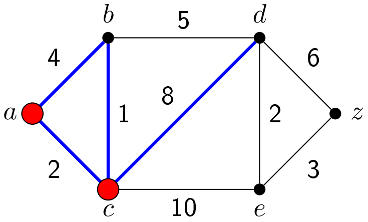
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0		2			

Example



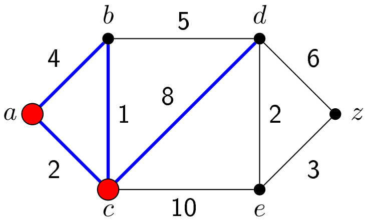
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	3	2			

Example



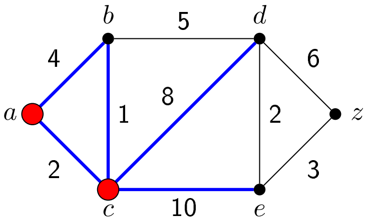
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	3	2			

Example



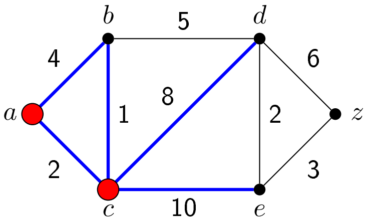
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	3	2	10		

Example



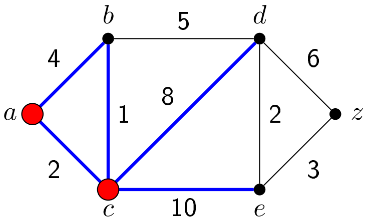
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	3	2	10		

Example



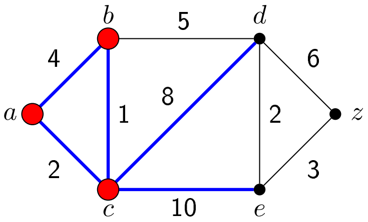
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	3	2	10	12	

Example



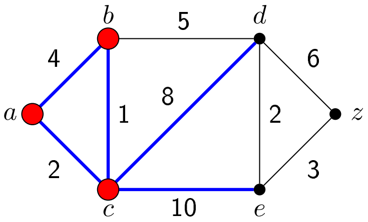
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	3	2	10	12	∞

Example



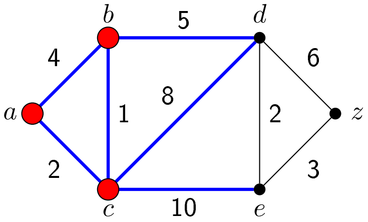
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞

Example



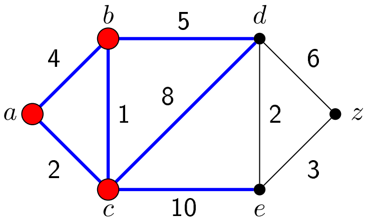
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2			

Example



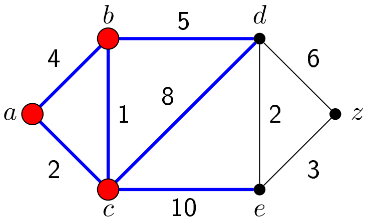
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2			

Example



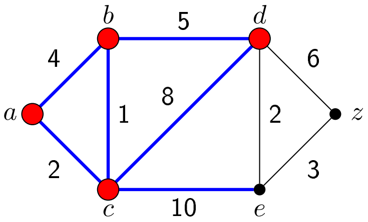
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2	8		

Example



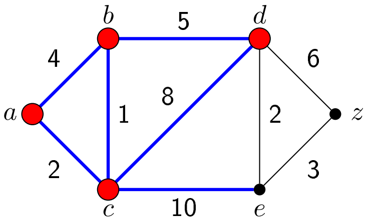
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2	8	12	∞

Example



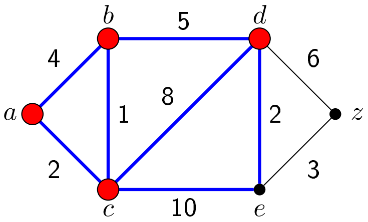
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2	**8**	12	∞

Example



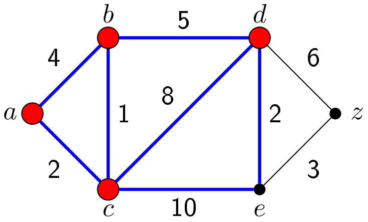
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2	**8**	12	∞
d	0	3	2	8		

Example



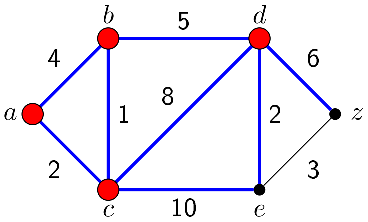
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2	**8**	12	∞
d	0	3	2	8		

Example



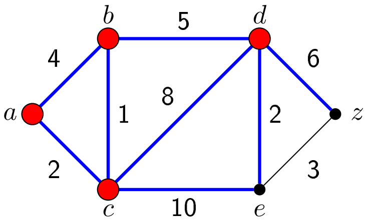
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2	**8**	12	∞
d	0	3	2	8	10	

Example



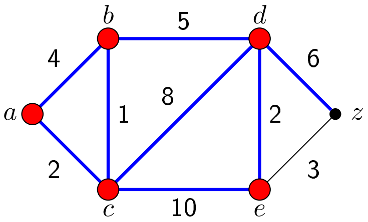
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2	**8**	12	∞
d	0	3	2	8	10	

Example



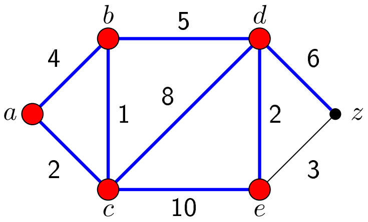
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2	**8**	12	∞
d	0	3	2	8	10	14

Example



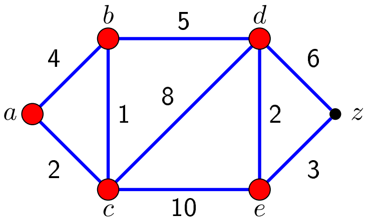
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2	**8**	12	∞
d	0	3	2	8	**10**	14

Example



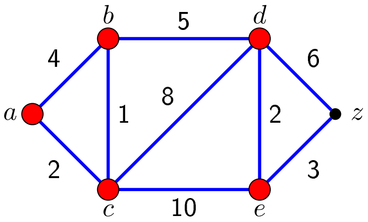
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2	**8**	12	∞
d	0	3	2	8	**10**	14
e	0	3	2	8	10	

Example



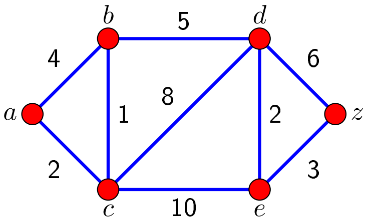
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2	**8**	12	∞
d	0	3	2	8	**10**	14
e	0	3	2	8	10	

Example



S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2	**8**	12	∞
d	0	3	2	8	**10**	14
e	0	3	2	8	10	13

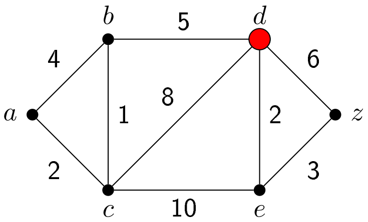
Example



S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2	**8**	12	∞
d	0	3	2	8	**10**	14
e	0	3	2	8	10	**13**

Back tracking procedure

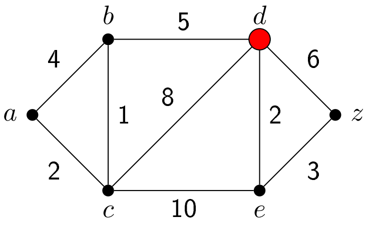
How to determine shortest path from a to d according to Dijkstra's algorithm?



S	a	b	c	d	e	z
$\emptyset$	$\underline{0}$	∞	∞	∞	∞	∞
a	0	4	$\underline{2}$	∞	∞	∞
c	0	$\underline{3}$	2	10	12	∞
b	0	3	2	$\underline{8}$	12	∞
d	0	3	2	8	$\underline{10}$	14
e	0	3	2	8	10	$\underline{13}$

Back tracking procedure

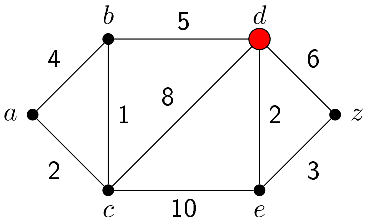
How to determine shortest path from a to d according to Dijkstra's algorithm?



S	a	b	c	d	e	z
$\emptyset$	$\underline{0}$	∞	∞	∞	∞	∞
a	0	4	$\underline{2}$	∞	∞	∞
c	0	$\underline{3}$	2	10	12	∞
b	0	3	2	$\underline{8}$	12	∞
d	0	3	2	8	$\underline{10}$	14
e	0	3	2	**8**	10	$\underline{13}$

Back tracking procedure

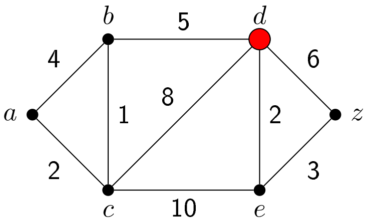
How to determine shortest path from a to d according to Dijkstra's algorithm?



S	a	b	c	d	e	z
$\emptyset$	$\underline{0}$	∞	∞	∞	∞	∞
a	0	4	$\underline{2}$	∞	∞	∞
c	0	$\underline{3}$	2	10	12	∞
b	0	3	2	$\underline{8}$	12	∞
d	0	3	2	**8**	$\underline{10}$	14
e	0	3	2	**8**	10	$\underline{13}$

Back tracking procedure

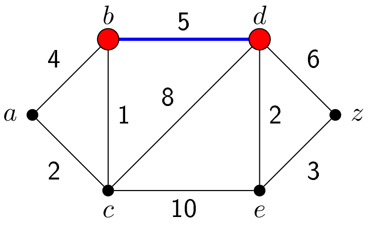
How to determine shortest path from a to d according to Dijkstra's algorithm?



S	a	b	c	d	e	z
$\emptyset$	$\underline{0}$	∞	∞	∞	∞	∞
a	0	4	$\underline{2}$	∞	∞	∞
c	0	$\underline{3}$	2	10	12	∞
b	0	3	2	$\underline{\mathbf{8}}$	12	∞
d	0	3	2	**8**	$\underline{10}$	14
e	0	3	2	**8**	10	$\underline{13}$

Back tracking procedure

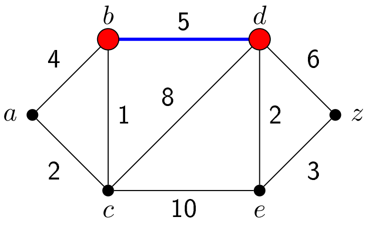
How to determine shortest path from a to d according to Dijkstra's algorithm?



S	a	b	c	d	e	z
$\emptyset$	$\underline{0}$	∞	∞	∞	∞	∞
a	0	4	$\underline{2}$	∞	∞	∞
c	0	$\underline{3}$	2	10	12	∞
b	0	3	2	$\underline{\mathbf{8}}$	12	∞
d	0	3	2	**8**	$\underline{10}$	14
e	0	3	2	**8**	10	$\underline{13}$

Back tracking procedure

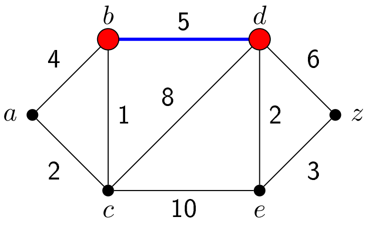
How to determine shortest path from a to d according to Dijkstra's algorithm?



S	a	b	c	d	e	z
$\emptyset$	$\underline{0}$	∞	∞	∞	∞	∞
a	0	4	$\underline{2}$	∞	∞	∞
c	0	$\underline{3}$	2	10	12	∞
b	0	**3**	2	$\underline{\mathbf{8}}$	12	∞
d	0	3	2	**8**	$\underline{10}$	14
e	0	3	2	**8**	10	$\underline{13}$

Back tracking procedure

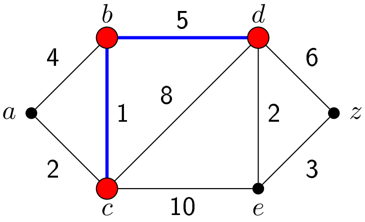
How to determine shortest path from a to d according to Dijkstra's algorithm?



S	a	b	c	d	e	z
$\emptyset$	$\underline{0}$	∞	∞	∞	∞	∞
a	0	4	$\underline{2}$	∞	∞	∞
c	0	$\underline{\mathbf{3}}$	2	10	12	∞
b	0	**3**	2	$\underline{\mathbf{8}}$	12	∞
d	0	3	2	**8**	$\underline{10}$	14
e	0	3	2	**8**	10	$\underline{13}$

Back tracking procedure

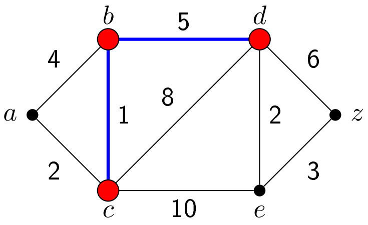
How to determine shortest path from a to d according to Dijkstra's algorithm?



S	a	b	c	d	e	z
$\emptyset$	$\underline{0}$	∞	∞	∞	∞	∞
a	0	4	$\underline{2}$	∞	∞	∞
c	0	$\underline{\mathbf{3}}$	2	10	12	∞
b	0	**3**	2	$\underline{\mathbf{8}}$	12	∞
d	0	3	2	**8**	$\underline{10}$	14
e	0	3	2	**8**	10	$\underline{13}$

Back tracking procedure

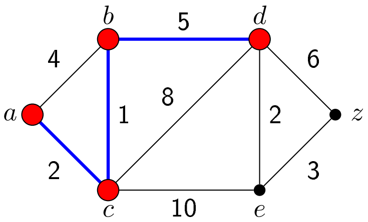
How to determine shortest path from a to d according to Dijkstra's algorithm?



S	a	b	c	d	e	z
$\emptyset$	$\underline{0}$	∞	∞	∞	∞	∞
a	0	4	$\underline{\mathbf{2}}$	∞	∞	∞
c	0	$\underline{\mathbf{3}}$	**2**	10	12	∞
b	0	**3**	2	$\underline{\mathbf{8}}$	12	∞
d	0	3	2	**8**	$\underline{10}$	14
e	0	3	2	**8**	10	$\underline{13}$

Back tracking procedure

How to determine shortest path from a to d according to Dijkstra's algorithm?



S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	**0**	4	**2**	∞	∞	∞
c	0	**3**	**2**	10	12	∞
b	0	**3**	2	**8**	12	∞
d	0	3	2	**8**	10	14
e	0	3	2	**8**	10	13

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Dijkstra's Algorithm

Property

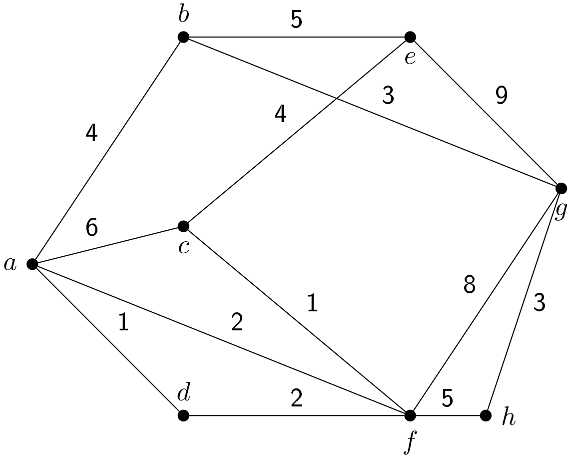
Applicable for any G, any length $\ell(v_i) \geq 0$, $\forall i$; one-to-all; complexity $O(|V|^2)$.

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Exercise

Example

Find the shortest path from a to other vertices using Dijkstra's algorithm.

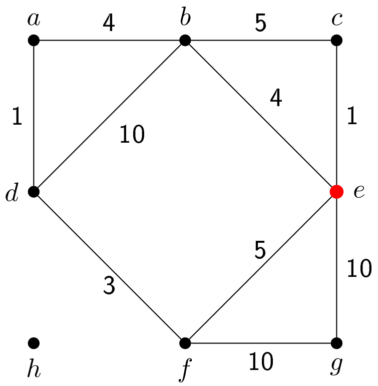


Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Exercise

Example

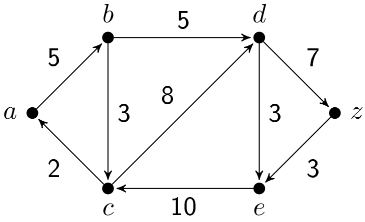
Find the shortest path from e to other vertices using Dijkstra's algorithm.



Exercise

Example

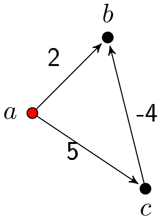
Find the shortest path from a to other vertices using Dijkstra's algorithm.



Dijkstra's Algorithm Flaw

Can Dijkstra's Algorithm be used on...

- ...digraph?
  - Yes!
- ...negative weighted graph?
  - No! Why?



Bellman-Ford Algorithm

```
procedure BellmanFord(G,a)
// Initialization Step
  forall vertices v
     Label[v] := ∞
     Prev[v] := -1
  Label(a) := 0 // a is the source node

// Iteration Step
  for i from 1 to size(vertices)-1
    forall vertices v
      if (Label[u] + Wt(u,v)) < Label[v]
        then
          Label[v] := Label[u] + Wt(u,v)
          Prev[v] := u

// Check circuit of negative weight
     forall vertices v
       if (Label[u] + Wt(u,v)) < Label(v)
         error "Contains circuit of negative weight"
```

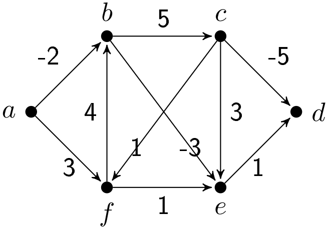
Property

any G, any weighted; one-to-all; detect whether there exists a circle of negative length; complexity $O(|V| \times |E|)$.

Example

Example

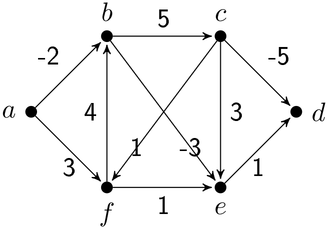
Step	a	b	c	d	e	f



Example

Example

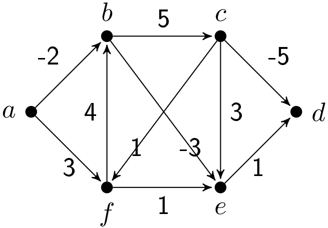
Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞



Example

Example

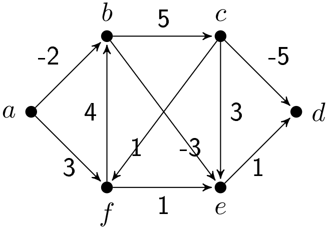
Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	-2a	∞	∞	∞	3a



Example

Example

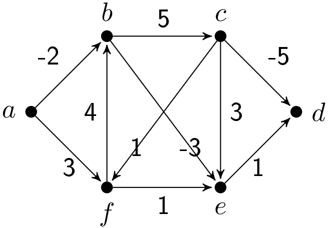
Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	-2a	∞	∞	∞	3a
2	0	-2	3b	∞	-5b	3



Example

Example

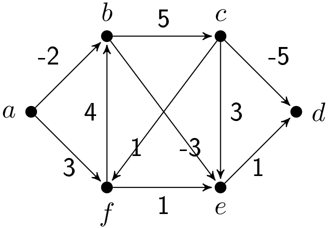
Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	-2a	∞	∞	∞	3a
2	0	-2	3b	∞	-5b	3
3	0	-2	3	-4e	-5	3



Example

Example

Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	-2a	∞	∞	∞	3a
2	0	-2	3b	∞	-5b	3
3	0	-2	3	-4e	-5	3
4	0	-2	3	-4	-5	3

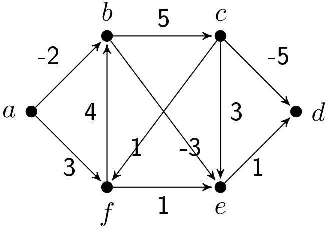


Example

Example

Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	-2a	∞	∞	∞	3a
2	0	-2	3b	∞	-5b	3
3	0	-2	3	-4e	-5	3
4	0	-2	3	-4	-5	3

Stop since Step 4 = Step 3.

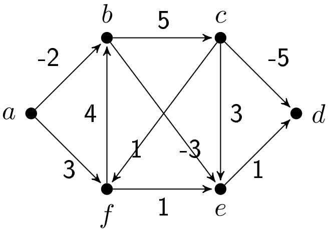


Backtracking procedure

Example

Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	-2a	∞	∞	∞	3a
2	0	-2	3b	∞	-5b	3
3	0	-2	3	-4e	-5	3
4	0	-2	3	-4	-5	3

Stop since Step 4 = Step 3.



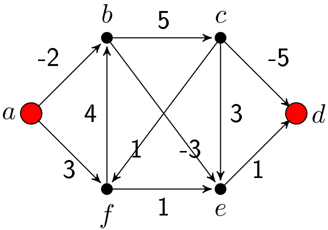
Backtracking procedure

Example

Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	-2a	∞	∞	∞	3a
2	0	-2	3b	∞	-5b	3
3	0	-2	3	-4e	-5	3
4	0	-2	3	-4	-5	3

Stop since Step 4 = Step 3.

How to find shortest path from a to d? $a \rightarrow$ d



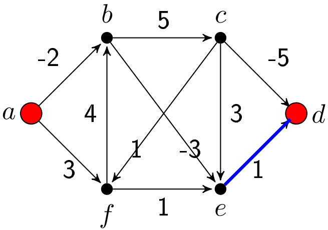
Backtracking procedure

Example

Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	-2a	∞	∞	∞	3a
2	0	-2	3b	∞	-5b	3
3	0	-2	3	**-4e**	-5	3
4	0	-2	3	**-4**	-5	3

Stop since Step 4 = Step 3.

How to find shortest path from a to d? $a \to \quad e \to d$



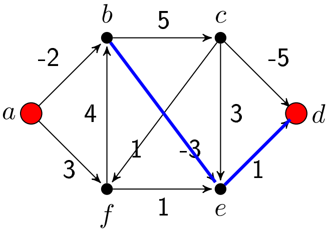
Backtracking procedure

Example

Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	-2a	∞	∞	∞	3a
2	0	-2	3b	∞	**-5b**	3
3	0	-2	3	**-4e**	**-5**	3
4	0	-2	3	**-4**	-5	3

Stop since Step 4 = Step 3.

How to find shortest path from a to d? $a \to b \to e \to d$



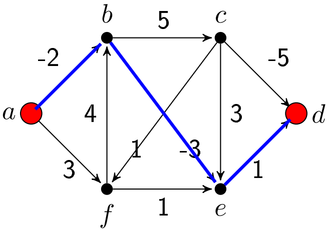
Backtracking procedure

Example

Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	**-2a**	∞	∞	∞	3a
2	0	**-2**	3b	∞	**-5b**	3
3	0	-2	3	**-4e**	**-5**	3
4	0	-2	3	**-4**	-5	3

Stop since Step 4 = Step 3.

How to find shortest path from a to d? $a \to b \to e \to d$

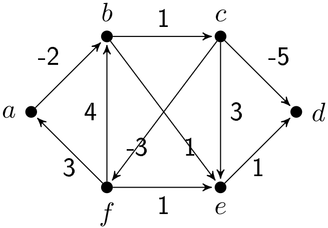


Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Example

Example

Step	a	b	c	d	e	f

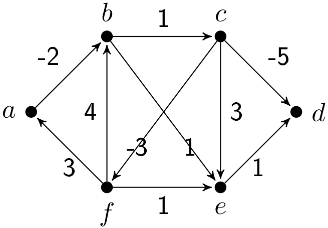


Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Example

Example

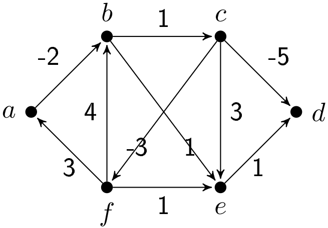
Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞



Example

Example

Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	-2a	∞	∞	∞	∞

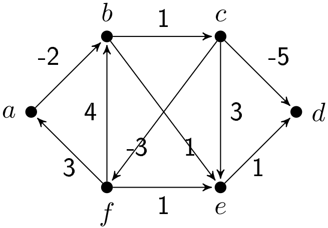


Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Example

Example

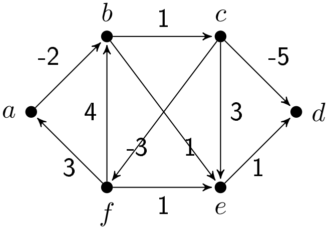
Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	-2a	∞	∞	∞	∞
2	0	-2	-1b	∞	-1b	∞



Example

Example

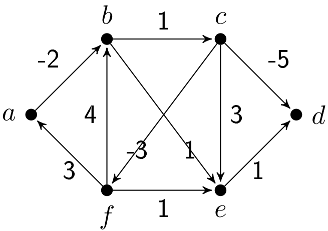
Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	-2a	∞	∞	∞	∞
2	0	-2	-1b	∞	-1b	∞
3	0	-2	-1	-6c	-1	-4c



Example

Example

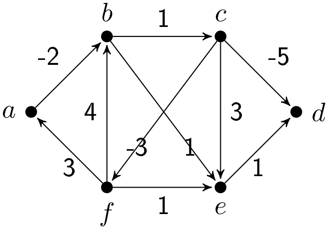
Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	-2a	∞	∞	∞	∞
2	0	-2	-1b	∞	-1b	∞
3	0	-2	-1	-6c	-1	-4c
4	-1f	-2	-1	-6	-3f	-4



Example

Example

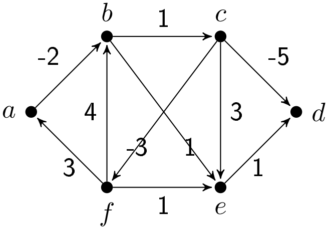
Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	-2a	∞	∞	∞	∞
2	0	-2	-1b	∞	-1b	∞
3	0	-2	-1	-6c	-1	-4c
4	-1f	-2	-1	-6	-3f	-4
5	-1	-3a	-1	-6	-3	-4



Example

Example

Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	-2a	∞	∞	∞	∞
2	0	-2	-1b	∞	-1b	∞
3	0	-2	-1	-6c	-1	-4c
4	-1f	-2	-1	-6	-3f	-4
5	-1	-3a	-1	-6	-3	-4
6	-1	-3	-2b	-6	-3	-4

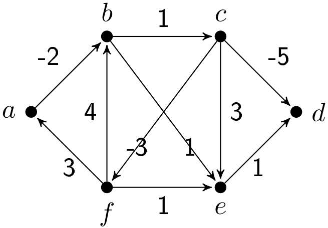


Example

Example

Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	-2a	∞	∞	∞	∞
2	0	-2	-1b	∞	-1b	∞
3	0	-2	-1	-6c	-1	-4c
4	-1f	-2	-1	-6	-3f	-4
5	-1	-3a	-1	-6	-3	-4
6	-1	-3	-2b	-6	-3	-4

There exists a circle of negative length since Step 6 $\neq$ Step 5.

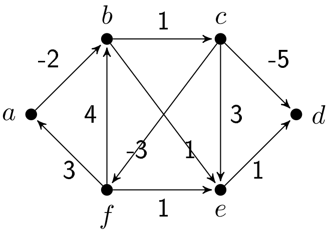


Example

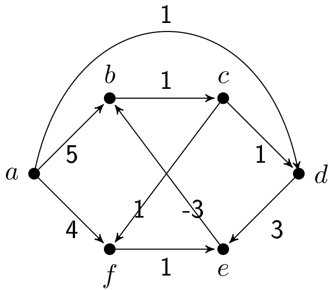
Example

Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	-2a	∞	∞	∞	∞
2	0	-2	-1b	∞	-1b	∞
3	0	-2	-1	-6c	-1	-4c
4	-1f	-2	-1	-6	-3f	-4
5	-1	-3a	-1	-6	-3	-4
6	-1	-3	-2b	-6	-3	-4
7	-1	-3	-2	-7c	-3	-4

There exists a circle of negative length since Step 6 $\neq$ Step 5.



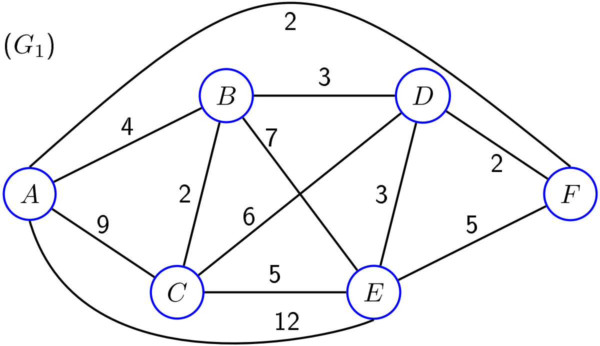
Example



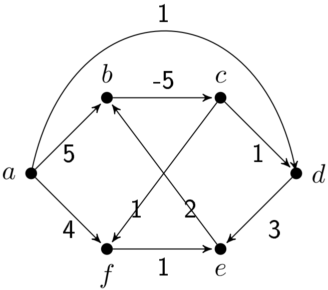
Example

Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	5a	∞	1a	∞	4a
2	0	5a	6b	1a	4d	4a
3	0	1e	6b	1a	4d	4a
4	0	1e	2b	1a	4d	4a
5	0	1e	2b	1a	4d	3c
6	0	1e	2b	1a	4d	3c

Exercise



Exercise



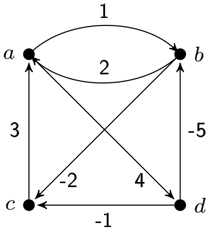
Floyd-Warshall Algorithm [1962]

```
procedure FloydWarshall ()
   for k := 1 to n
      for i := 1 to n
         for j := 1 to n
            path[i,j] = min (path[i,j],
                             path[i,k]+path[k,j]);
```

Property

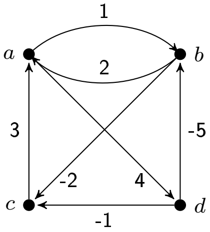
any G, any weighted; all-to-all; this is an software algorithm; complexity $O(|V|^3)$.

Example



$$L^{(0)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & \infty_0 \\ 3_0 & \infty_0 & 0_0 & \infty_0 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

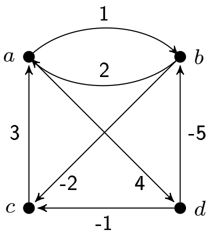
Example



$$L^{(0)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & \infty_0 \\ 3_0 & \infty_0 & 0_0 & \infty_0 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

$$L^{(1)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & & & \\ 3_0 & & & \\ \infty_0 & & & \end{pmatrix}$$

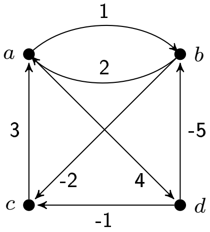
Example



$$L^{(0)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & \infty_0 \\ 3_0 & \infty_0 & 0_0 & \infty_0 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

$$L^{(1)} = \begin{pmatrix} \color{red}{0_0} & \color{red}{1_0} & \color{red}{\infty_0} & \color{red}{4_0} \\ \color{red}{2_0} & 0_0 & -2_0 & \color{blue}{6_1} \\ \color{red}{3_0} & \color{blue}{4_1} & 0_0 & \color{blue}{7_1} \\ \color{red}{\infty_0} & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

Example



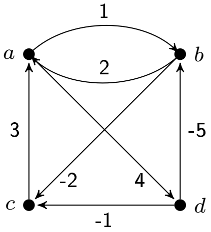
$$L^{(0)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & \infty_0 \\ 3_0 & \infty_0 & 0_0 & \infty_0 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

$$L^{(1)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & 6_1 \\ 3_0 & 4_1 & 0_0 & 7_1 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

$$L^{(2)} = \begin{pmatrix} & 1_0 & & \\ 2_0 & 0_0 & -2_0 & 6_1 \\ & 4_1 & & \\ & -5_0 & & \end{pmatrix}$$

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Example

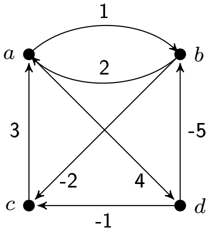


$$L^{(0)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & \infty_0 \\ 3_0 & \infty_0 & 0_0 & \infty_0 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

$$L^{(1)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & 6_1 \\ 3_0 & 4_1 & 0_0 & 7_1 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

$$L^{(2)} = \begin{pmatrix} 0_0 & 1_0 & -1_2 & 4_0 \\ 2_0 & 0_0 & -2_0 & 6_1 \\ 3_0 & 4_1 & 0_0 & 7_1 \\ -3_2 & -5_0 & -7_2 & 0_0 \end{pmatrix}$$

Example



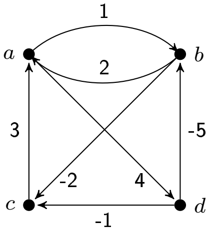
$$L^{(0)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & \infty_0 \\ 3_0 & \infty_0 & 0_0 & \infty_0 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

$$L^{(1)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & 6_1 \\ 3_0 & 4_1 & 0_0 & 7_1 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

$$L^{(2)} = \begin{pmatrix} 0_0 & 1_0 & -1_2 & 4_0 \\ 2_0 & 0_0 & -2_0 & 6_1 \\ 3_0 & 4_1 & 0_0 & 7_1 \\ -3_2 & -5_0 & -7_2 & 0_0 \end{pmatrix}$$

$$L^{(3)} = \begin{pmatrix} & & -1_2 & \\ & & -2_0 & \\ 3 & 4_1 & 0_0 & 7_1 \\ & & -7_2 & \end{pmatrix}$$

Example



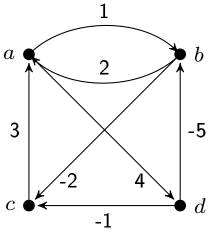
$$L^{(0)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & \infty_0 \\ 3_0 & \infty_0 & 0_0 & \infty_0 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

$$L^{(1)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & 6_1 \\ 3_0 & 4_1 & 0_0 & 7_1 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

$$L^{(2)} = \begin{pmatrix} 0_0 & 1_0 & -1_2 & 4_0 \\ 2_0 & 0_0 & -2_0 & 6_1 \\ 3_0 & 4_1 & 0_0 & 7_1 \\ -3_2 & -5_0 & -7_2 & 0_0 \end{pmatrix}$$

$$L^{(3)} = \begin{pmatrix} 0_0 & 1_0 & -1_2 & 4_0 \\ 1_3 & 0_0 & -2_0 & 5_3 \\ 3 & 4_1 & 0_0 & 7_1 \\ -4_3 & -5_0 & -7_2 & 0_0 \end{pmatrix}$$

Example



$$L^{(0)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & \infty_0 \\ 3_0 & \infty_0 & 0_0 & \infty_0 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

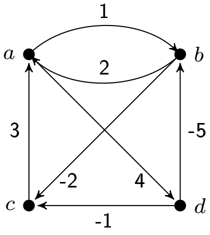
$$L^{(1)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & 6_1 \\ 3_0 & 4_1 & 0_0 & 7_1 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

$$L^{(2)} = \begin{pmatrix} 0_0 & 1_0 & -1_2 & 4_0 \\ 2_0 & 0_0 & -2_0 & 6_1 \\ 3_0 & 4_1 & 0_0 & 7_1 \\ -3_2 & -5_0 & -7_2 & 0_0 \end{pmatrix}$$

$$L^{(3)} = \begin{pmatrix} 0_0 & 1_0 & -1_2 & 4_0 \\ 1_3 & 0_0 & -2_0 & 5_3 \\ 3 & 4_1 & 0_0 & 7_1 \\ -4_3 & -5_0 & -7_2 & 0_0 \end{pmatrix}$$

$$L^{(4)} = \begin{pmatrix} & & & 4_0 \\ & & & 5_3 \\ & & & 7_1 \\ -4_3 & -5_0 & -7_2 & 0_0 \end{pmatrix}$$

Example



$$L^{(0)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & \infty_0 \\ 3_0 & \infty_0 & 0_0 & \infty_0 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

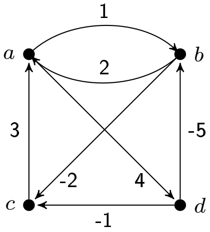
$$L^{(1)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & 6_1 \\ 3_0 & 4_1 & 0_0 & 7_1 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

$$L^{(2)} = \begin{pmatrix} 0_0 & 1_0 & -1_2 & 4_0 \\ 2_0 & 0_0 & -2_0 & 6_1 \\ 3_0 & 4_1 & 0_0 & 7_1 \\ -3_2 & -5_0 & -7_2 & 0_0 \end{pmatrix}$$

$$L^{(3)} = \begin{pmatrix} 0_0 & 1_0 & -1_2 & 4_0 \\ 1_3 & 0_0 & -2_0 & 5_3 \\ 3 & 4_1 & 0_0 & 7_1 \\ -4_3 & -5_0 & -7_2 & 0_0 \end{pmatrix}$$

$$L^{(4)} = \begin{pmatrix} 0_0 & -1_4 & -3_4 & 4_0 \\ 1_3 & 0_0 & -2_0 & 5_3 \\ 3_0 & 2_4 & 0_0 & 7_1 \\ -4_3 & -5_0 & -7_2 & 0_0 \end{pmatrix}$$

Example



Shortest path from b to d
(5_3 from $L^{(4)}$):

$$L^{(0)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & \infty_0 \\ 3_0 & \infty_0 & 0_0 & \infty_0 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

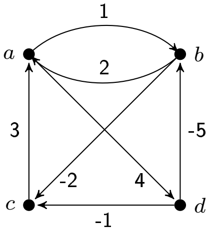
$$L^{(1)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & 6_1 \\ 3_0 & 4_1 & 0_0 & 7_1 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

$$L^{(2)} = \begin{pmatrix} 0_0 & 1_0 & -1_2 & 4_0 \\ 2_0 & 0_0 & -2_0 & 6_1 \\ 3_0 & 4_1 & 0_0 & 7_1 \\ -3_2 & -5_0 & -7_2 & 0_0 \end{pmatrix}$$

$$L^{(3)} = \begin{pmatrix} 0_0 & 1_0 & -1_2 & 4_0 \\ 1_3 & 0_0 & -2_0 & 5_3 \\ 3 & 4_1 & 0_0 & 7_1 \\ -4_3 & -5_0 & -7_2 & 0_0 \end{pmatrix}$$

$$L^{(4)} = \begin{pmatrix} 0_0 & -1_4 & -3_4 & 4_0 \\ 1_3 & 0_0 & -2_0 & 5_3 \\ 3_0 & 2_4 & 0_0 & 7_1 \\ -4_3 & -5_0 & -7_2 & 0_0 \end{pmatrix}$$

Example



Shortest path from b to d
(5_3 from $L^{(4)}$):
$bd = bc + cd$
($5_3 = -2_0 + 7_1$ from $L^{(3)}$)

$$L^{(0)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & \infty_0 \\ 3_0 & \infty_0 & 0_0 & \infty_0 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

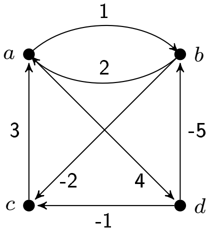
$$L^{(1)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & 6_1 \\ 3_0 & 4_1 & 0_0 & 7_1 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

$$L^{(2)} = \begin{pmatrix} 0_0 & 1_0 & -1_2 & 4_0 \\ 2_0 & 0_0 & -2_0 & 6_1 \\ 3_0 & 4_1 & 0_0 & 7_1 \\ -3_2 & -5_0 & -7_2 & 0_0 \end{pmatrix}$$

$$L^{(3)} = \begin{pmatrix} 0_0 & 1_0 & -1_2 & 4_0 \\ 1_3 & 0_0 & -2_0 & 5_3 \\ 3 & 4_1 & 0_0 & 7_1 \\ -4_3 & -5_0 & -7_2 & 0_0 \end{pmatrix}$$

$$L^{(4)} = \begin{pmatrix} 0_0 & -1_4 & -3_4 & 4_0 \\ 1_3 & 0_0 & -2_0 & 5_3 \\ 3_0 & 2_4 & 0_0 & 7_1 \\ -4_3 & -5_0 & -7_2 & 0_0 \end{pmatrix}$$

Example



Shortest path from b to d
(5_3 from $L^{(4)}$):
$bd = bc + cd$
($5_3 = -2_0 + 7_1$ from $L^{(3)}$)
$cd = ca + ad$
($7_1 = 3_0 + 4_0$ from $L^{(1)}$)

$$L^{(0)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & \infty_0 \\ 3_0 & \infty_0 & 0_0 & \infty_0 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

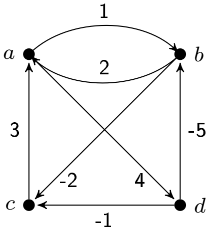
$$L^{(1)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & 6_1 \\ 3_0 & 4_1 & 0_0 & 7_1 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

$$L^{(2)} = \begin{pmatrix} 0_0 & 1_0 & -1_2 & 4_0 \\ 2_0 & 0_0 & -2_0 & 6_1 \\ 3_0 & 4_1 & 0_0 & 7_1 \\ -3_2 & -5_0 & -7_2 & 0_0 \end{pmatrix}$$

$$L^{(3)} = \begin{pmatrix} 0_0 & 1_0 & -1_2 & 4_0 \\ 1_3 & 0_0 & -2_0 & 5_3 \\ 3 & 4_1 & 0_0 & 7_1 \\ -4_3 & -5_0 & -7_2 & 0_0 \end{pmatrix}$$

$$L^{(4)} = \begin{pmatrix} 0_0 & -1_4 & -3_4 & 4_0 \\ 1_3 & 0_0 & -2_0 & 5_3 \\ 3_0 & 2_4 & 0_0 & 7_1 \\ -4_3 & -5_0 & -7_2 & 0_0 \end{pmatrix}$$

Example



Shortest path from b to d
(5_3 from $L^{(4)}$):
$bd = bc + cd$
($5_3 = -2_0 + 7_1$ from $L^{(3)}$)
$cd = ca + ad$
($7_1 = 3_0 + 4_0$ from $L^{(1)}$)
$\Rightarrow bd = bc + ca + ad$

$$L^{(0)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & \infty_0 \\ 3_0 & \infty_0 & 0_0 & \infty_0 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

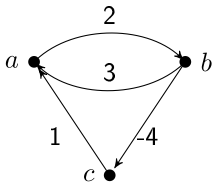
$$L^{(1)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & 6_1 \\ 3_0 & 4_1 & 0_0 & 7_1 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

$$L^{(2)} = \begin{pmatrix} 0_0 & 1_0 & -1_2 & 4_0 \\ 2_0 & 0_0 & -2_0 & 6_1 \\ 3_0 & 4_1 & 0_0 & 7_1 \\ -3_2 & -5_0 & -7_2 & 0_0 \end{pmatrix}$$

$$L^{(3)} = \begin{pmatrix} 0_0 & 1_0 & -1_2 & 4_0 \\ 1_3 & 0_0 & -2_0 & 5_3 \\ 3 & 4_1 & 0_0 & 7_1 \\ -4_3 & -5_0 & -7_2 & 0_0 \end{pmatrix}$$

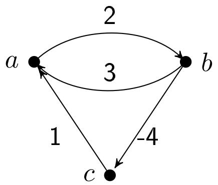
$$L^{(4)} = \begin{pmatrix} 0_0 & -1_4 & -3_4 & 4_0 \\ 1_3 & 0_0 & -2_0 & 5_3 \\ 3_0 & 2_4 & 0_0 & 7_1 \\ -4_3 & -5_0 & -7_2 & 0_0 \end{pmatrix}$$

Example



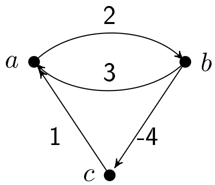
$$L^{(0)} = \begin{pmatrix} 0_0 & 2_0 & \infty_0 \\ 3_0 & 0_0 & -4_0 \\ 1_0 & \infty_0 & 0_0 \end{pmatrix}$$

Example



$$L^{(0)} = \begin{pmatrix} 0_0 & 2_0 & \infty_0 \\ 3_0 & 0_0 & -4_0 \\ 1_0 & \infty_0 & 0_0 \end{pmatrix} \quad L^{(1)} = \begin{pmatrix} 0_0 & 2_0 & \infty_0 \\ 3_0 & 0_0 & -4_0 \\ 1_0 & 3_1 & 0_0 \end{pmatrix}$$

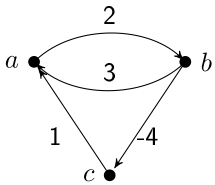
Example



$$L^{(0)} = \begin{pmatrix} 0_0 & 2_0 & \infty_0 \\ 3_0 & 0_0 & -4_0 \\ 1_0 & \infty_0 & 0_0 \end{pmatrix} \quad L^{(1)} = \begin{pmatrix} 0_0 & 2_0 & \infty_0 \\ 3_0 & 0_0 & -4_0 \\ 1_0 & 3_1 & 0_0 \end{pmatrix}$$

$$L^{(2)} = \begin{pmatrix} 0_0 & 2_0 & -2_0 \\ 3_0 & 0_0 & -4_0 \\ 1_0 & 3_1 & -1_2 \end{pmatrix}$$

Example

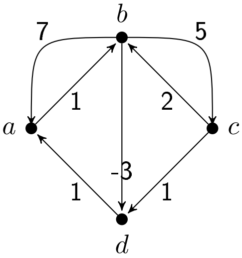


$$L^{(0)} = \begin{pmatrix} 0_0 & 2_0 & \infty_0 \\ 3_0 & 0_0 & -4_0 \\ 1_0 & \infty_0 & 0_0 \end{pmatrix} \quad L^{(1)} = \begin{pmatrix} 0_0 & 2_0 & \infty_0 \\ 3_0 & 0_0 & -4_0 \\ 1_0 & 3_1 & 0_0 \end{pmatrix}$$

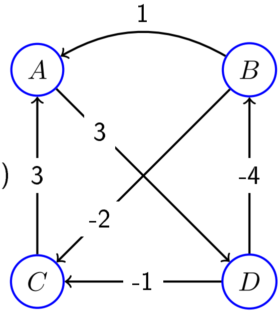
$$L^{(2)} = \begin{pmatrix} 0_0 & 2_0 & -2_0 \\ 3_0 & 0_0 & -4_0 \\ 1_0 & 3_1 & -1_2 \end{pmatrix}$$

STOP, there exists a circuit of negative length.

Exercise



Exercise



(G_2)

Ford's algorithm

$\pi(1) = 0$

For each $j \in V$ do

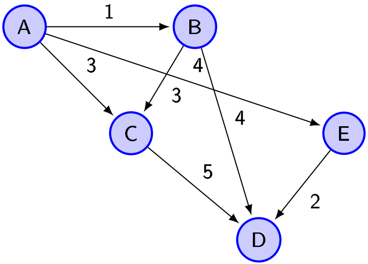
$\pi(j) = \min_{i \in \rho_j^{-1}(\pi(i)+\ell[i,j])}$

End

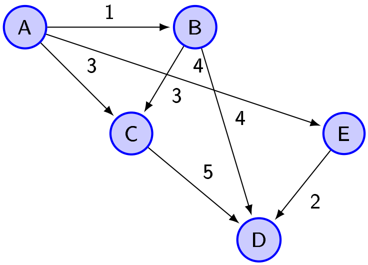
Property

G without circle, positive length; one-to-all; rank table definition; complexity $O(|V|)$.

Example



Example

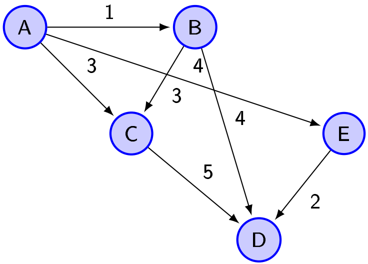


i	Γ_i^{-1}	rank(i)
A		
B		
C		
D		
E		

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

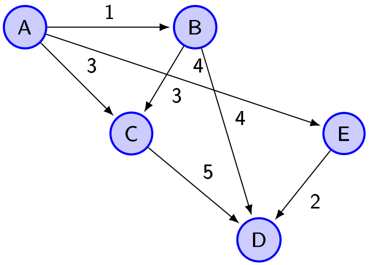
Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Example



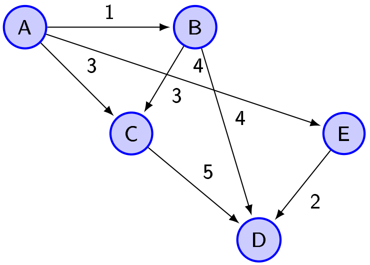
i	Γ_i^{-1}	rank(i)
A	-	
B	A	
C	A, B	
D	B, C , E	
E	A	

Example



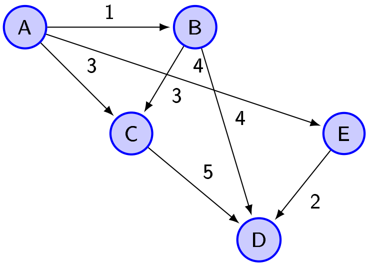
i	Γ_i^{-1}	rank(i)
A	-	0
B	A	
C	A, B	
D	B, C , E	
E	A	

Example



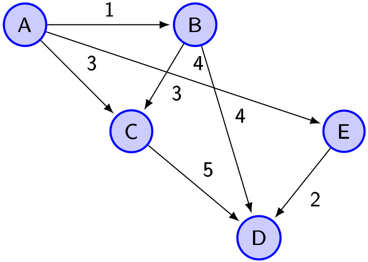
i	Γ_i^{-1}	rank(i)
A	-	0
B		1
C	B	
D	B, C , E	
E		1

Example



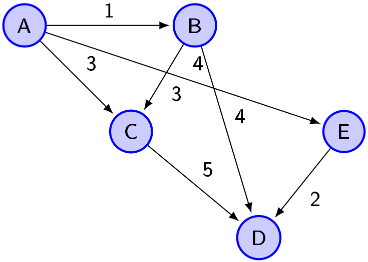
i	Γ_i^{-1}	rank(i)
A	-	0
B		1
C		2
D	C	
E		1

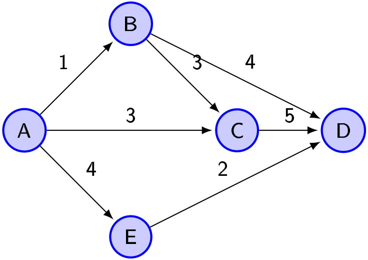
Example



i	Γ_i^{-1}	rank(i)
A	-	0
B		1
C		2
D		3
E		1

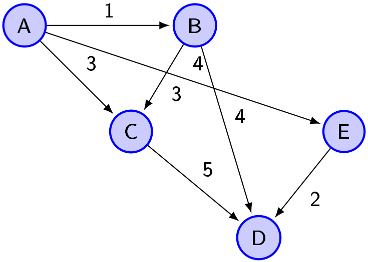
Example

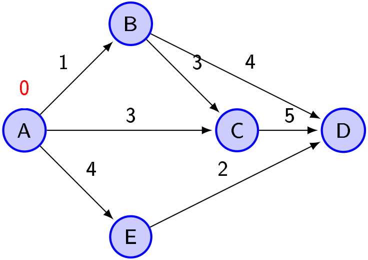




i	Γ_i^{-1}	rank(i)
A	-	0
B		1
C		2
D		3
E		1

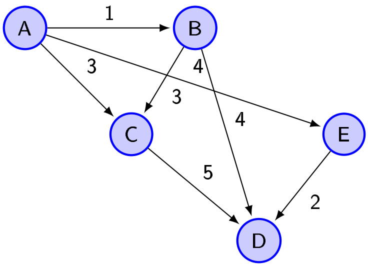
Example

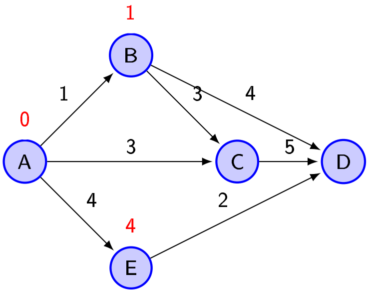




i	Γ_i^{-1}	rank(i)
A	-	0
B		1
C		2
D		3
E		1

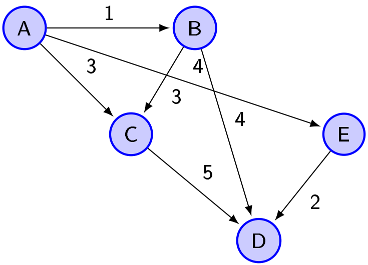
Example

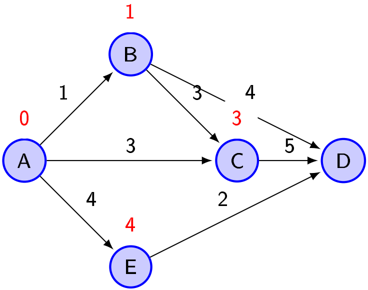




i	Γ_i^{-1}	rank(i)
A	-	0
B		1
C		2
D		3
E		1

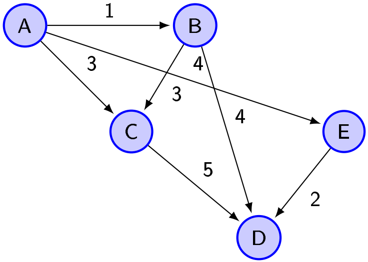
Example

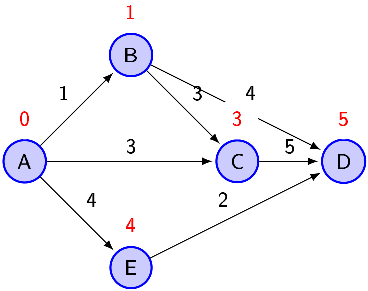




i	Γ_i^{-1}	rank(i)
A	-	0
B		1
C		2
D		3
E		1

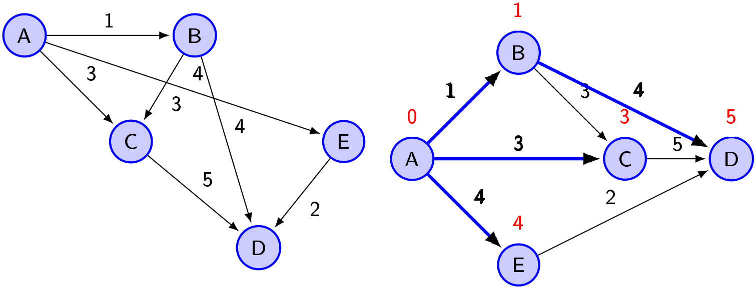
Example





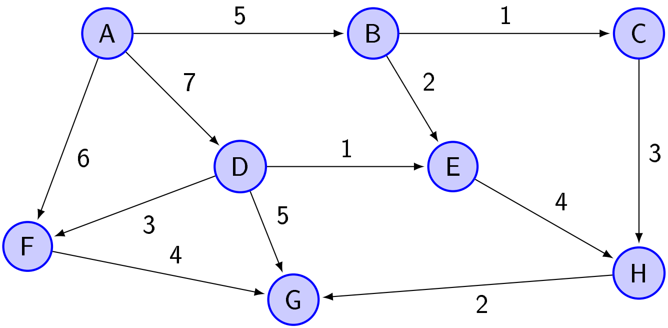
i	Γ_i^{-1}	rank(i)
A	-	0
B		1
C		2
D		3
E		1

Example



i	Γ_i^{-1}	rank(i)
A	-	0
B		1
C		2
D		3
E		1

Exercise



Application

Problem

A young professor in Hue is invited to teach some years in Ho Chi Minh university of technology. He decides to represent the diverse operations of his transfer by a graph and, in this purpose, establishes the list of following operations:

- A: Find a house in Ho Chi Minh city.
- B: Choose a removal man and sign a contract of move
- C: Make pack his furniture by the removal man
- D: Make transport his furniture towards Ho Chi Minh city
- E: Find an accommodation to HCM (from Hue)
- F: Transport his family to HCM
- G: Move into his new accommodation
- H: Register the children to their new school
- I: Look for a temporary work for his wife
- J: Fit out the new accommodation and pay this arrangement with the first treatment of his wife
- K: Find a small bar to celebrate in family the success of the move and express the enjoyment to live in a good accommodation arrangement

Application

Considering constraint of posteriority following: $A < F$; $B < C$; $C < D \wedge F$; $D < G$; $E < F$; $F < G \wedge H \wedge I$; $G < K$; $H < K$; $I < J$; $J < K$.

Approximated job processing times :

A	B	C	D	E	F	G	H	I	J	K
10	2	3	4	7	3	5	1	3	8	2

Application

Considering constraint of posteriority following: $A < F$; $B < C$; $C < D \wedge F$; $D < G$; $E < F$; $F < G \wedge H \wedge I$; $G < K$; $H < K$; $I < J$; $J < K$.

Approximated job processing times :

A	B	C	D	E	F	G	H	I	J	K
10	2	3	4	7	3	5	1	3	8	2

Question

- Determine a schedule of the 'movement' with minimal duration.
- What happens if his new accommodation is not available before date 20? In that case, of what margin we have to make the task J?

Question

How to determine a shortest path from u to v in graph G which traverses at most $\leq$ a given constant number of intermediate vertices.

Other shortest path problems

- multicriteria shortest path problem
  - linear combinaison
  - ε-constraint approach
  - lexico-graphical order
- k shortest paths problem
  - allowing loop
  - loppless
- multi-point shortest path
  - TSP, VRP

Top k shortest paths query

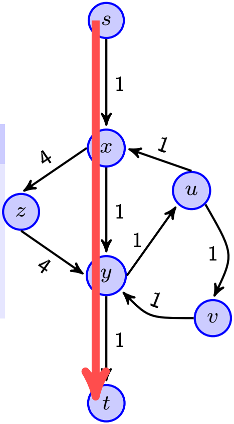
When the shortest path is not sufficient for application, top-k shortest paths are desired.

Top k shortest paths query

When the shortest path is not sufficient for application, top-k shortest paths are desired.

Top 3 general shortest paths (allowing loops)

- 1^{st}: $s \to x \to y \to t$, $l = 3$



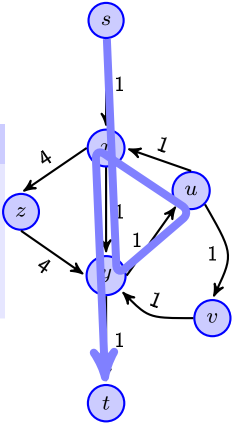
Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Top k shortest paths query

When the shortest path is not sufficient for application, top-k shortest paths are desired.

Top 3 general shortest paths (allowing loops)

- 1^{st}: $s \to x \to y \to t$, $l = 3$
- 2^{nd}:
  $s \to x \to y \to u \to x \to y \to t$, $l = 6$
  or $s \to x \to y \to u \to v \to y \to t$, $l = 6$



Top k shortest paths query

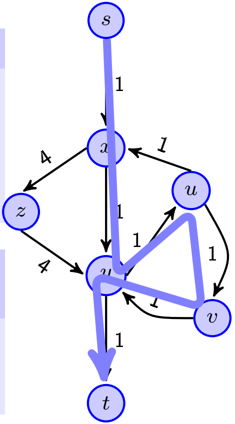
When the shortest path is not sufficient for application, top-k shortest paths are desired.

Top 3 general shortest paths (allowing loops)

- 1^{st}: $s \to x \to y \to t$, $l = 3$
- 2^{nd}:
  $s \to x \to y \to u \to x \to y \to t$, $l = 6$
  or $s \to x \to y \to u \to v \to y \to t$, $l = 6$

Top 2 elementary shortest paths (without loops)

- 1^{st}: $s \to x \to y \to t$, $l = 3$
- 2^{nd}: $s \to x \to z \to y \to t$, $l = 10$



Top k shortest paths query

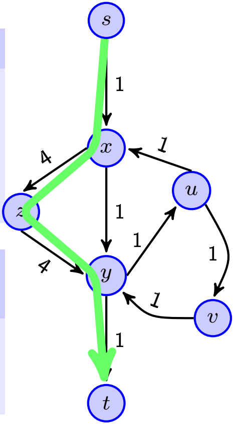
When the shortest path is not sufficient for application, top-k shortest paths are desired.

Top 3 general shortest paths (allowing loops)

- 1^{st}: $s \to x \to y \to t$, $l = 3$
- 2^{nd}:
  $s \to x \to y \to u \to x \to y \to t$, $l = 6$
  or $s \to x \to y \to u \to v \to y \to t$, $l = 6$

Top 2 elementary shortest paths (without loops)

- 1^{st}: $s \to x \to y \to t$, $l = 3$
- 2^{nd}: $s \to x \to z \to y \to t$, $l = 10$

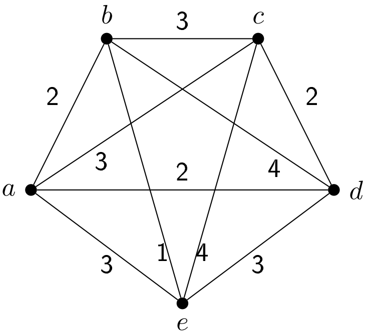


Traveling Salesman Problem (TSP)

Problem

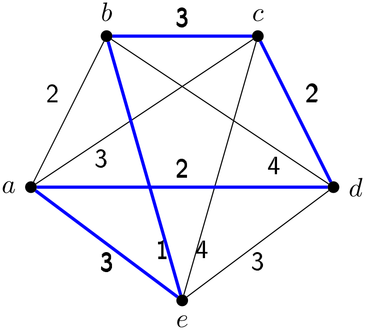
- Given a set of n customers located in n cities and distances for each pair of cities, the problem involves finding a round-trip with the minimum traveling cost.
- The vehicle must visit each customer exactly once and return to its point of origin also called depot.
- The objective function is the total cost of the tour.
- $\mathcal{NP}$-complete: all known techniques for obtaining an exact solution require an exponentially increasing number of steps (computing resources) as the problems become larger.
- **TSP is one of the most intensely studied problems in computational mathematics, yet no effective solution method.**

Traveling Salesman Problem



- The total number of possible Hamilton circuit is $(n-1)!/2$.
- For example, if there are 25 customers to visit, the total number of solutions is $24!/2 = 3.1 \times 10^{23}$.

Traveling Salesman Problem



- The total number of possible Hamilton circuit is $(n-1)!/2$.
- For example, if there are 25 customers to visit, the total number of solutions is $24!/2 = 3.1 \times 10^{23}$.
- If the depot is located at node 1, then the optimal tour is $1-5-2-3-4-1$ with total cost equal to 11.

Vehicle Routing Problem (VRP)

Problem

- The vehicle routing problem involves finding a set of trips, one for each vehicle, to deliver known quantities of goods to a set of customers.
- The objective is to minimize the travel costs of all trips combined.
- There may be upper bounds on the total load of each vehicle and the total duration of its trip.
- The most basic Vehicle Routing Problem (VRP) is the single-depot capacitate VRP.

Maps and Graphs

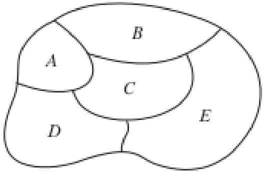
Definition

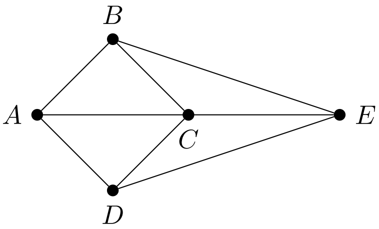
- Every map can be represented by a graph. We call it dual graph.
- Problem of coloring the regions of a map $\rightarrow$ coloring the vertices of the dual graph so that no two adjacent vertices have the same color.

Maps and Graphs

Definition

- Every map can be represented by a graph. We call it dual graph.
- Problem of coloring the regions of a map → coloring the vertices of the dual graph so that no two adjacent vertices have the same color.

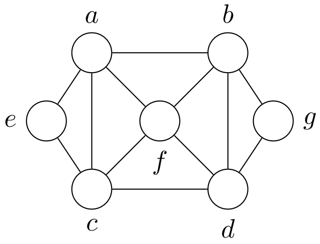




Graph coloring

Definition

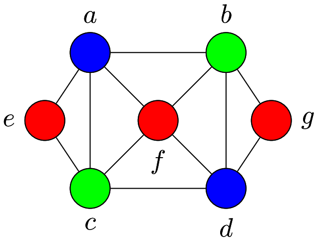
- A coloring (*tô màu*) of a simple graph is the assignment of a color to each vertex of the graph so that no two adjacent vertices are assigned the same color.



Graph coloring

Definition

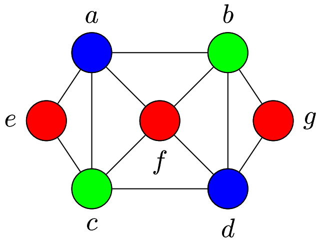
- A coloring (*tô màu*) of a simple graph is the assignment of a color to each vertex of the graph so that no two adjacent vertices are assigned the same color.
- The chromatic number (*số màu*) of a graph, denoted by $\chi(G)$, is the least number of colors needed for a coloring of this graph.

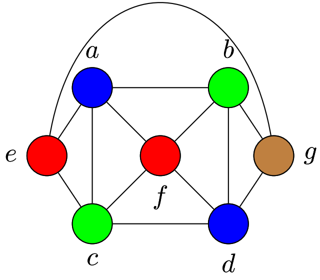


Graph coloring

Definition

- A coloring (*tô màu*) of a simple graph is the assignment of a color to each vertex of the graph so that no two adjacent vertices are assigned the same color.
- The chromatic number (*số màu*) of a graph, denoted by $\chi(G)$, is the least number of colors needed for a coloring of this graph.





Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Four color theorem

Theorem (Four color theorem)

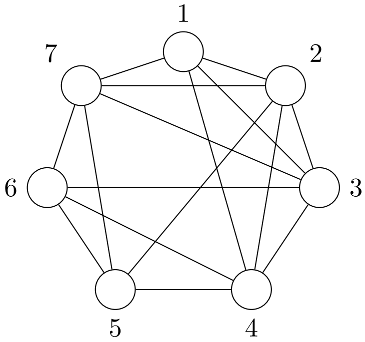
The chromatic number of a planar graph is no greater than four.

- Was a conjecture in the 1850s
- Was not proved completely until 1976 by Kenneth Appel and Wolfgang Haken, using computer
- No proof not relying on a computer has yet been found

Applications of Graph coloring

Scheduling Final Exam

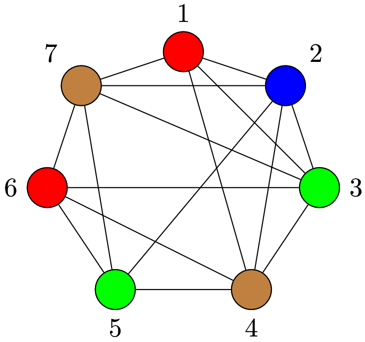
- How can the final exams at a university be scheduled so that no student has two exams at the same time?
- Suppose we have 7 finals, numbered 1 through 7.
- The pairs of courses have common students are depicted in the following graph



Applications of Graph coloring

Scheduling Final Exam

- How can the final exams at a university be scheduled so that no student has two exams at the same time?
- Suppose we have 7 finals, numbered 1 through 7.
- The pairs of courses have common students are depicted in the following graph



Applications of Graph Coloring

Other Applications

- Frequency Assignments: Television channels 2 through 12 are assigned to stations in North America so that no two stations within 150 miles can operate on the same channel. How can the assignment of channels be modeled by graph coloring?

Applications of Graph Coloring

Other Applications

- Frequency Assignments: Television channels 2 through 12 are assigned to stations in North America so that no two stations within 150 miles can operate on the same channel. How can the assignment of channels be modeled by graph coloring?
- Index Registers: In an execution of loop, the frequently used variables should be stored in index registers to speed up. How many index registers are needed?

Revision

Cho $G = (V, E)$ là một đồ thị đơn và vô hướng bất kỳ, có n đỉnh. Định nghĩa đồ thị bù của G là $G^c = (V, F)$ thỏa hai tính chất: $G \cup G^c = K_n$ và $E \cap F = \emptyset$.

Cho $H = (V, E)$ là một đồ thị đơn và vô hướng bất kỳ. Điều nào sau đây là đúng?

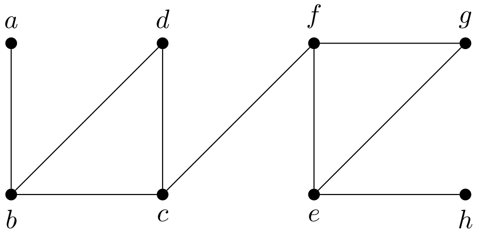
A) H và H^c là liên thông

B) H chứa đường đi Euler và H^c chứa đường đi Euler

C) H hoặc H^c là liên thông

D) H hoặc H^c chứa đường đi Hamilton

Revision



Chọn phát biểu đúng liên quan đến khái niệm đỉnh cắt (*cut vertex*) và cạnh cắt (*cut edge*) cho đồ thị trên.

A) Đồ thị có 2 đỉnh cắt.
B) Đồ thị có 4 đỉnh cắt.
C) Đồ thị có 1 cạnh cắt.
D) Đồ thị có 2 đỉnh cắt và 1 cạnh cắt.

Revision

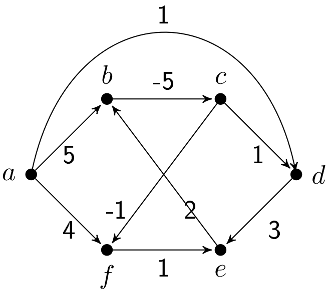
Chọn phát biểu đúng dưới đây về mối liên quan giữa cạnh cắt (cut edge) và đỉnh cắt (cut vertex)

A) Hai đầu mút của cạnh cắt phải là đỉnh cắt.
B) Hai đầu mút của cạnh cắt có thể không phải là đỉnh cắt.
C) Một trong hai đầu mút của cạnh cắt phải là đỉnh cắt.
D) Chỉ một trong hai đầu mút của cạnh cắt là đỉnh cắt.

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

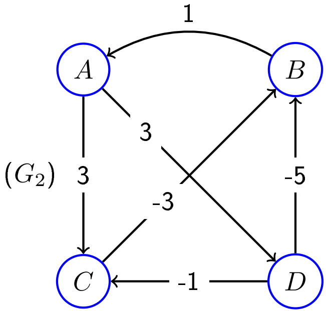
Revision

Determine a shortest path from a to other vertices in the following graph.



Revision

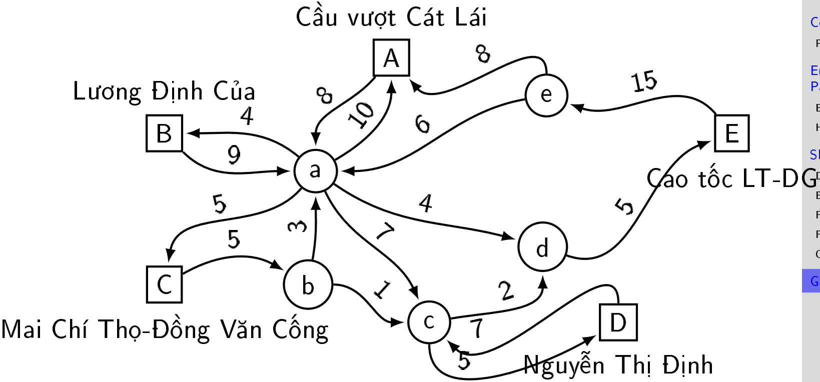
Determine a shortest path from any vertex to other vertex in the following graph (using Floyd-Warshall algorithm).



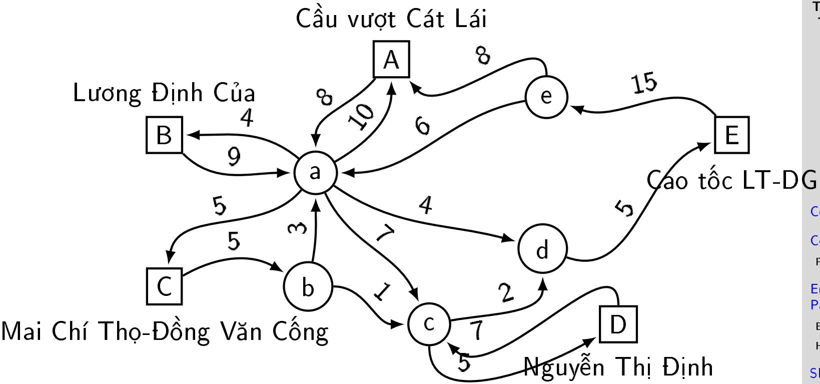
Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Revision

Các hướng giao thông ở giao lộ Mai Chí Thọ và Cao tốc Long Thành-Dầu Giây (LT-DG) được mô hình bằng một đồ thị đơn có hướng có trọng số như dưới đây. Các đỉnh hình tròn là những điểm giao cắt trong giao lộ, và những đỉnh hình vuông là những điểm vào giao lộ. Trọng số của đồ thị (nằm trên cạnh) thể hiện thời gian di chuyển (tính bằng giây) trên các cạnh tương ứng.



Revision



(Câu 1.) Thời gian di chuyển nhanh nhất giữa các cặp đỉnh A $\rightarrow$ B, C $\rightarrow$ E và D $\rightarrow$ A tương ứng là

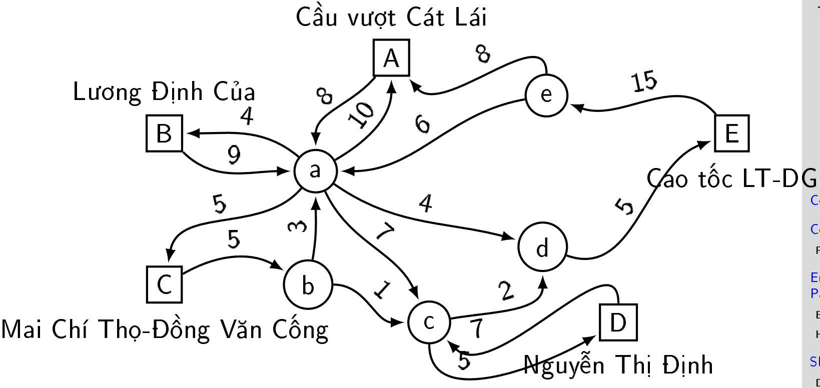
A) 19, 14 và 23

B) 12, 14 và 24

C) 12, 13 và 23

D) 12, 13 và 37

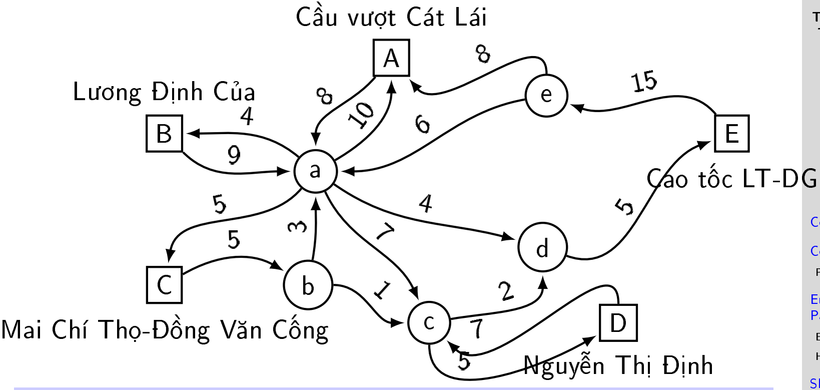
Revision



(Câu 2.) Cặp đỉnh vào và ra giao lộ nào có thời gian di chuyển lâu nhất?

A) E → A
B) D → A
C) D → C
D) Các đáp án khác đều sai.

Revision



(Câu 3.) Sở GTVT TP.HCM mong muốn giảm thời gian tối đa di chuyển qua giao lộ này xuống không lớn hơn 32 (với mọi cặp đỉnh vào giao lộ). Hãy cho biết nếu được phép tạo thêm 1 cạnh (có hướng) với trọng số là 13 thì phải thêm cạnh nào sau đây.

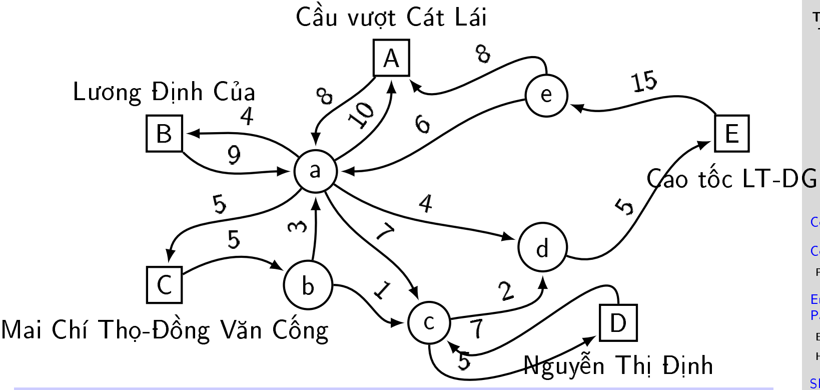
A) (a,b)

B) (c,a)

C) (d,e)

D) Không có cách nào.

Revision

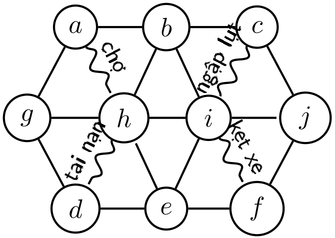


(Câu 4.) Các đỉnh giao cắt trong giao lộ là nguồn gốc của tắc đường do xung đột giữa các hướng lưu thông chéo nhau. Vì thế các đỉnh này thường được lắp các đèn điều khiển giao thông để xen kẽ cho phép các hướng di chuyển. Hãy cho biết những đỉnh nào cần lắp đèn điều khiển.

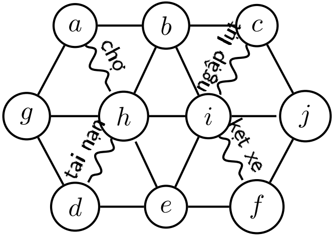
- A) a, b và c
- B) a và c
- C) a và d
- D) a, c, e và d

Revision

Để xây dựng hệ thống xe buýt đưa rước các em học sinh phổ thông, người ta cần xác định một tập các trạm dừng. Mỗi ngày, các em học sinh sẽ di chuyển từ nhà đến một trạm dừng đã được xác định sẵn trước, các em cần đứng chờ xe trước thời điểm xe đến.Xe sẽ đón các em tại các trạm này và đưa đến tận trường học, và sau khi kết thúc giờ học, xe sẽ đưa mỗi em từ trường về đến trạm dừng mà đã đón em đó vào buổi sáng.
Việc xác định các trạm này cần thỏa mãn điều kiện là **các trạm dừng không thể quá xa nhà của các em học sinh**.Và để tiện lợi, các trạm dừng chỉ được chọn trong số các địa chỉ nhà của học sinh.
Xét bản đồ gồm các địa chỉ nhà $a, b, c, d, e, f, g, h, i, j$ như bên dưới đây. Giả định rằng các em học sinh cư ngụ tại tất cả địa chỉ từ a đến j.
Các cạnh trong bản đồ đôi khi có nhãn dùng để lưu thông tin trạng thái của đường cần lưu ý theo số liệu thống kê. Có bốn loại nhãn: kẹt xe, tai nạn, ngập lụt, chợ. Các cạnh không có nhãn biểu diễn đường thông thoáng.



Revision



(Câu 1.) Đồ thị trong bản đồ có thể

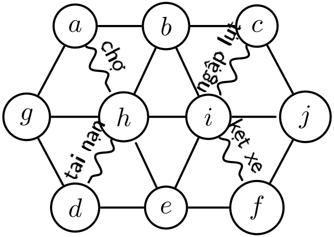
A) tồn tại chu trình Hamilton.

B) tồn tại chu trình Euler.

C) tồn tại đường đi Euler.

D) tồn tại nhiều đường đi Euler.

Revision

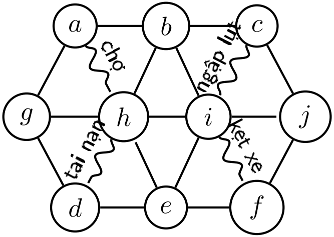


(Câu 2.) Khoảng cách không quá xa được xác định bằng tối đa một cạnh trong bản đồ, số trạm cần đặt để thỏa mãn các điều kiện trên là

- A) $\{1, \ldots, 4\}$
- B) $\{2, \ldots, 10\}$
- C) 4
- D) $\{1, \ldots, 10\}$

Nguyen An Khuong, Tran Tuan Anh, Nguyen Tien Thinh, Mai Xuan Toan, Tran Hong Tai

Revision



(Câu 3.) Để hệ thống vận hành tốt, ràng buộc về khoảng cách không quá xa nên được xác định bởi tối đa một cạnh *thông thoáng* trong bản đồ. Số trạm cần đặt để thỏa mãn các điều kiện là

- A) $\{1, \ldots, 4\}$
- B) $\{2, \ldots, 10\}$
- C) $\{3, \ldots, 10\}$
- D) $\{1, \ldots, 10\}$