Chapter 9
More About Graphs

Discrete Structures for Computing on 27 May 2014

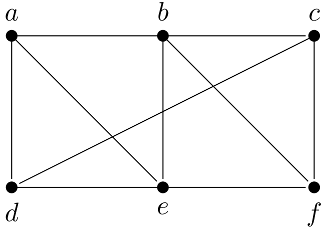
Huynh Tuong Nguyen, Tran Vinh Tan
Faculty of Computer Science and Engineering
University of Technology - VNUHCM

Acknowledgement

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

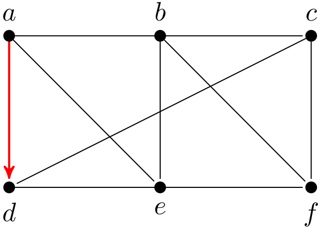
Contents

Paths and Circuits

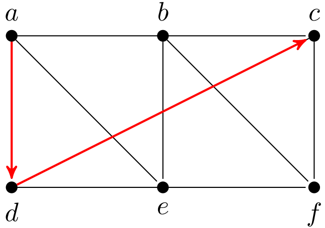


Huynh Tuong Nguyen, Tran Vinh Tan

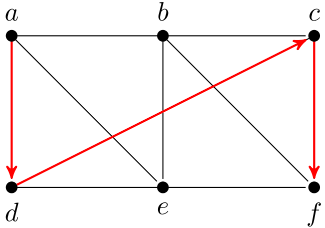
Paths and Circuits



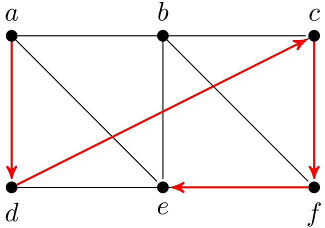
Paths and Circuits



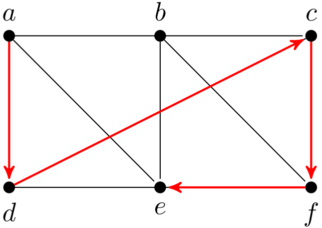
Paths and Circuits



Paths and Circuits

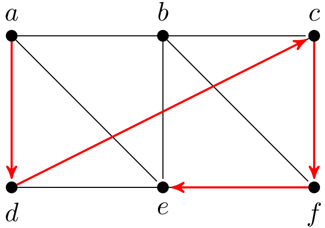


Paths and Circuits



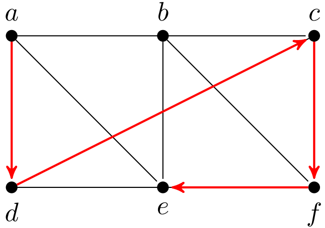
Simple path of length 4

Paths and Circuits

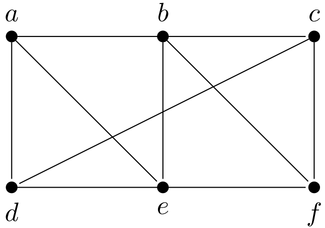


Simple path of length 4

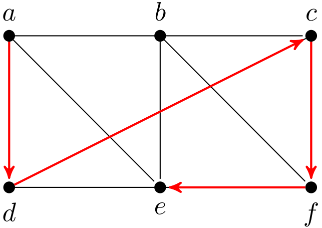
Paths and Circuits



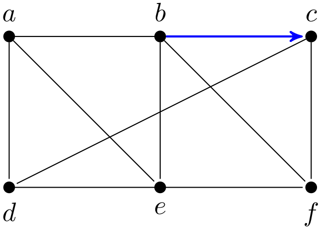
Simple path of length 4



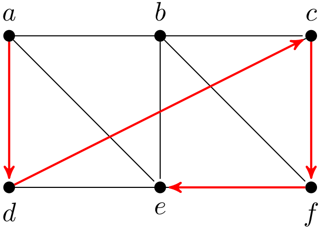
Paths and Circuits



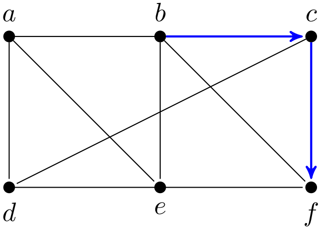
Simple path of length 4



Paths and Circuits

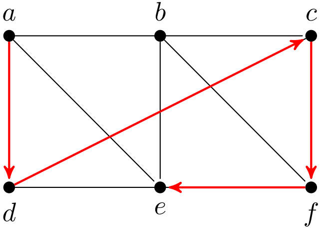


Simple path of length 4

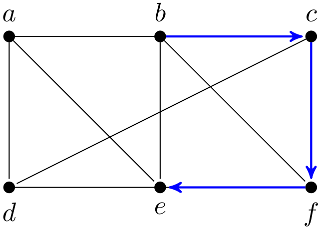


Huynh Tuong Nguyen, Tran Vinh Tan

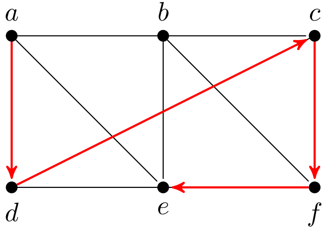
Paths and Circuits



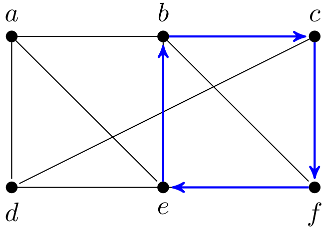
Simple path of length 4



Paths and Circuits

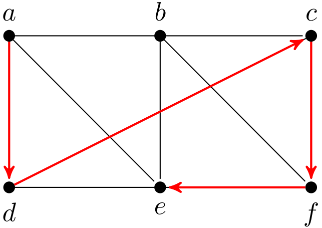


Simple path of length 4

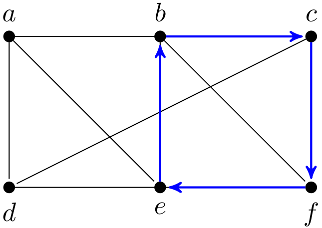


Huynh Tuong Nguyen, Tran Vinh Tan

Paths and Circuits



Simple path of length 4

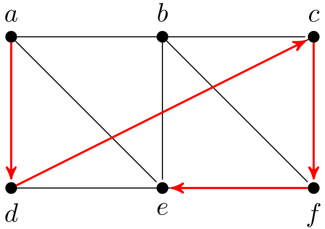


Circuit of length 4

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.

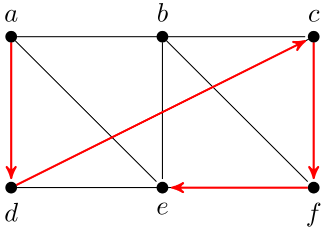


Simple path

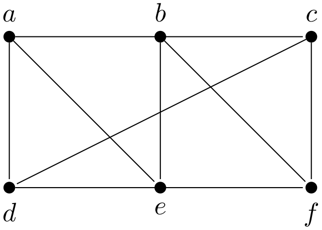
Path and Circuits

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

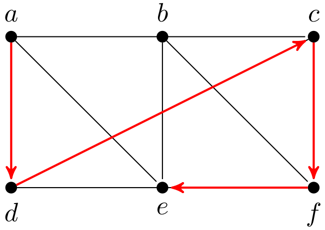


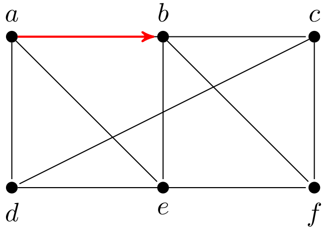
Contents

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.



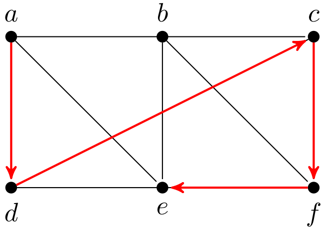


Simple path

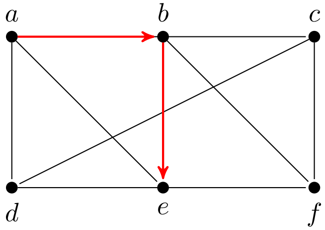
Path and Circuits

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.



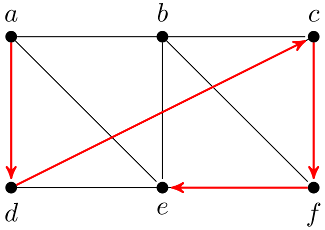
Simple path

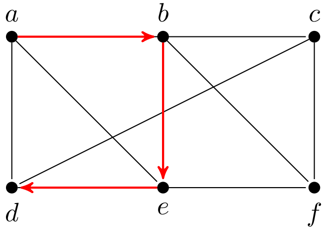


Path and Circuits

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.



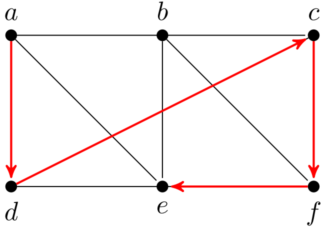


Simple path

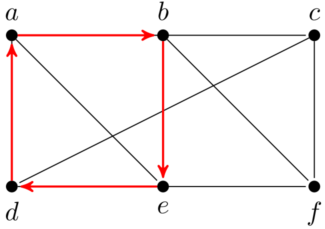
Path and Circuits

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.



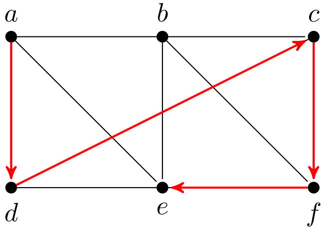
Simple path



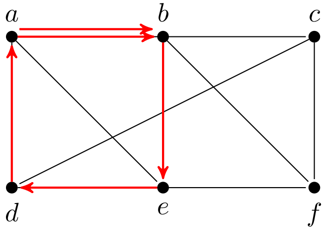
Path and Circuits

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.



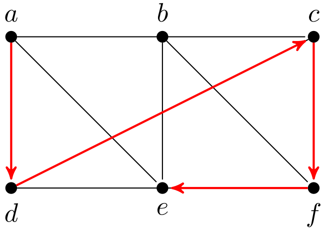
Simple path



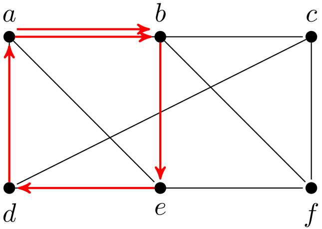
Path and Circuits

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

Path and Circuits

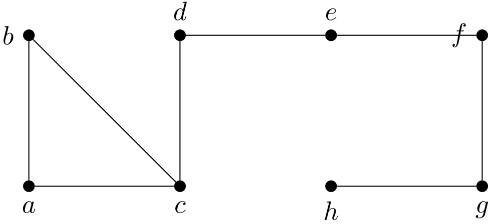
Definition (in directed graphs)

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.

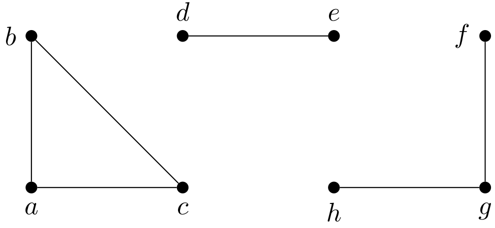


Connected graph

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.

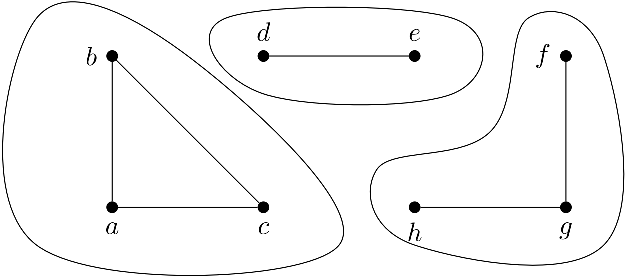


Disconnected graph

Connectedness in Undirected Graphs

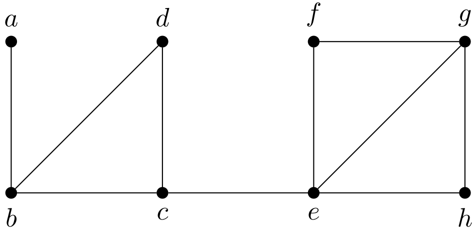
Definition

- 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*)

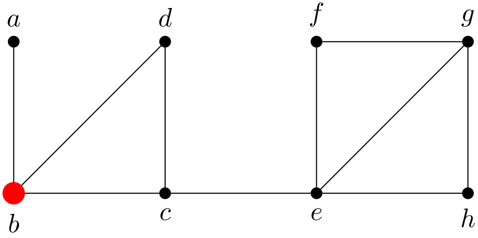
How Connected is a Graph?



Definition

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

How Connected is a Graph?

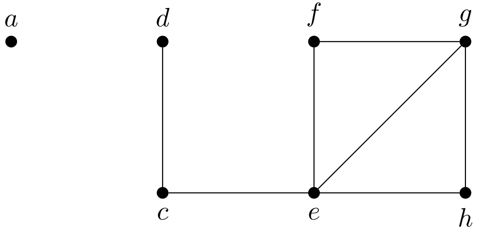


Definition

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

Huynh Tuong Nguyen, Tran Vinh Tan

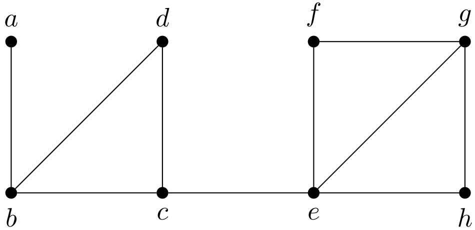
How Connected is a Graph?



Definition

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

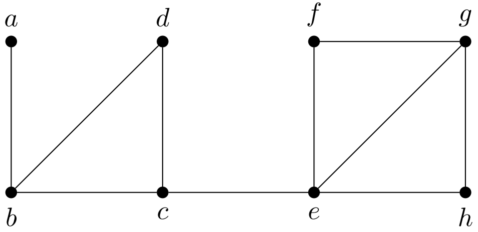
How Connected is a Graph?



Definition

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

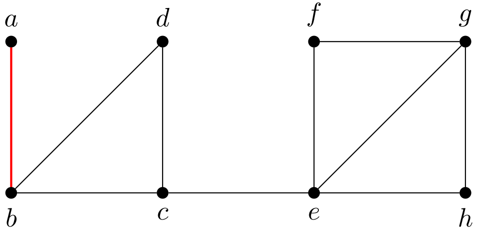
How Connected is a Graph?



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*).

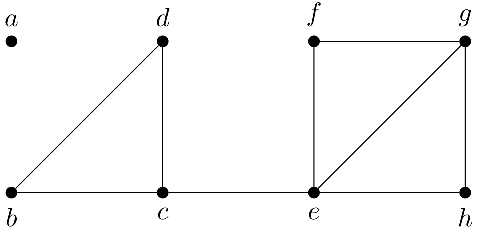
How Connected is a Graph?



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*).

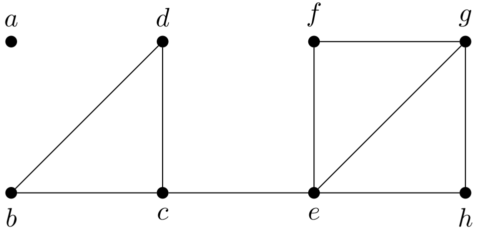
How Connected is a Graph?



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*).

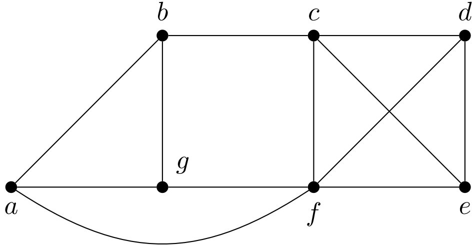
How Connected is a Graph?



Definition

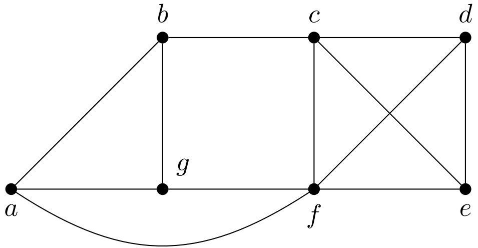
- b is a cut vertex (*đỉnh cắt*) or articulation point (*điểm khớp*). What else?
- $\{a, b\}$ is a cut edge (*cạnh cắt*) or bridge (*cầu*). What else?

How Connected is a Graph?



Definition

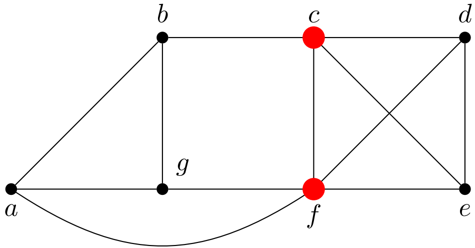
How Connected is a Graph?



Definition

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

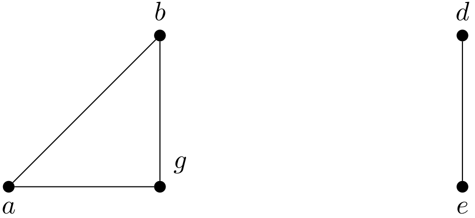
How Connected is a Graph?



Definition

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

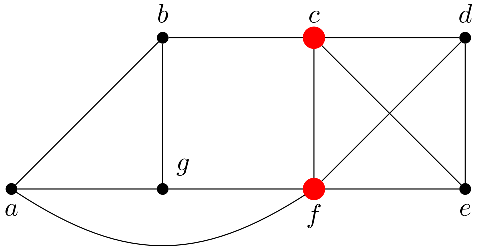
How Connected is a Graph?



Definition

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

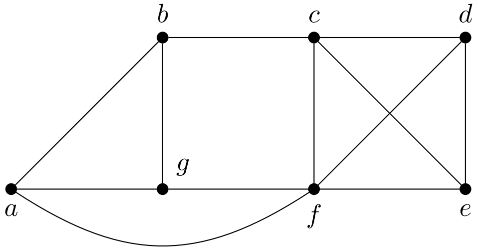
How Connected is a Graph?



Definition

- This graph don'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*) $\kappa(G) = 2$.

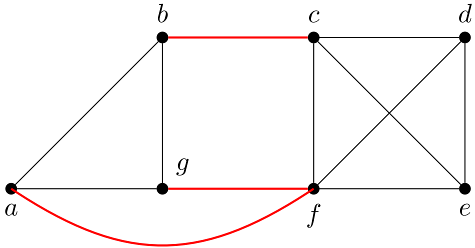
How Connected is a Graph?



Definition

- This graph don'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*) $\kappa(G) = 2$.

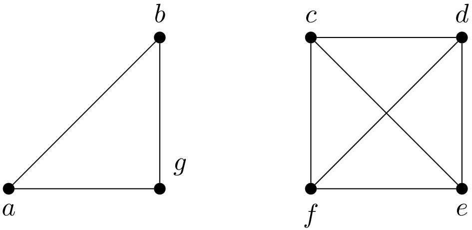
How Connected is a Graph?



Definition

- This graph don'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?

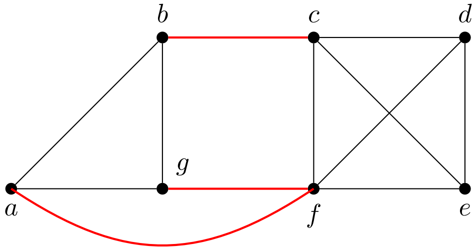


Definition

- This graph don'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*) $\kappa(G) = 2$.

Contents

How Connected is a Graph?



Definition

- This graph don'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*) $\lambda(G) = 3$.

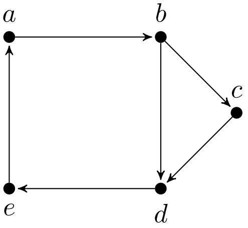
Applications of Vertex and Edge Connectivity

- 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

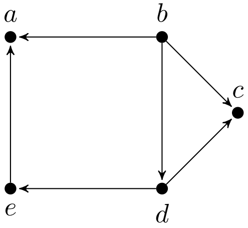
Connectedness in Directed Graphs

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

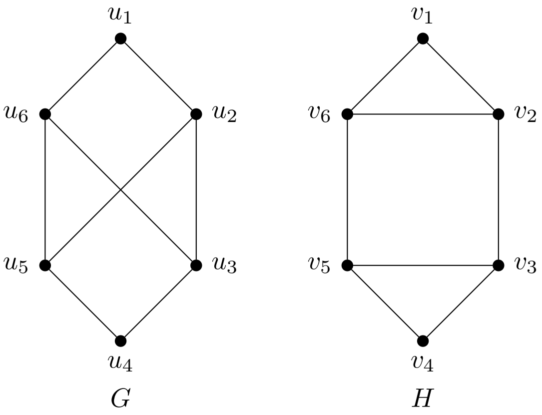


Weakly connected

Applications

Example

Determine whether the graphs below are isomorphic.



Solution

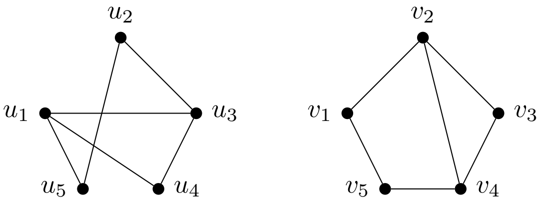
H has a simple circuit of length three, not G.

Huynh Tuong Nguyen, Tran Vinh Tan

Applications

Example

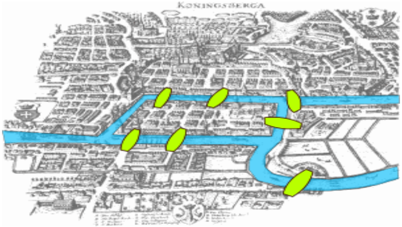
Determine whether the graphs below are isomorphic.



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.

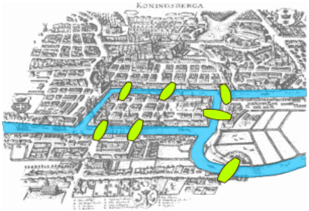
The Famous Problem of Seven Bridges of Königsberg



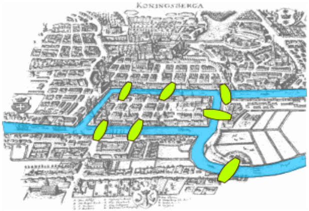
- Is there a route that a person crosses all the seven bridges once?

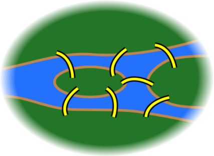
Euler Solution

Euler Solution

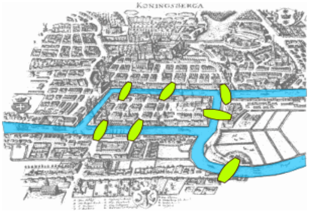


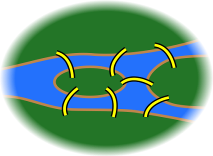
Euler Solution

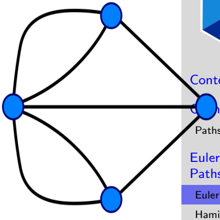




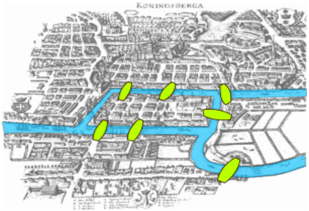
Euler Solution

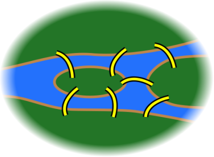


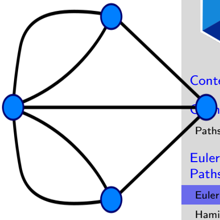




Euler Solution







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

What is Euler Path and Circuit?

What is Euler Path and Circuit?

- **Euler Path** (*đường đi Euler*) is a path in the graph that passes each edge only 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?

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.

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.

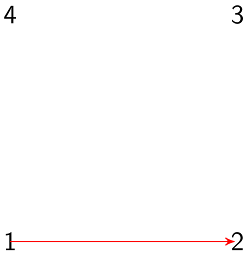
Examples of Euler Path and Circuit

Examples of Euler Path and Circuit

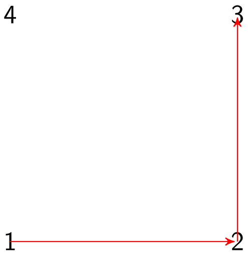
4 3

1 2

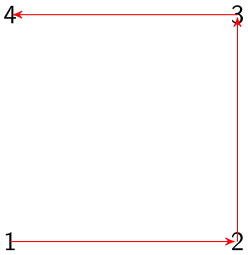
Examples of Euler Path and Circuit



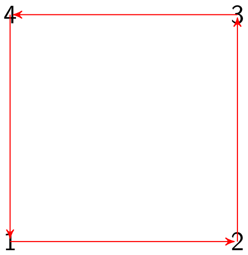
Examples of Euler Path and Circuit



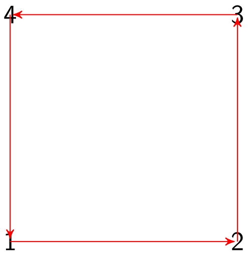
Examples of Euler Path and Circuit



Examples of Euler Path and Circuit

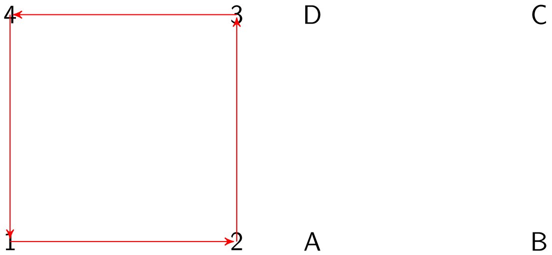


Examples of Euler Path and Circuit



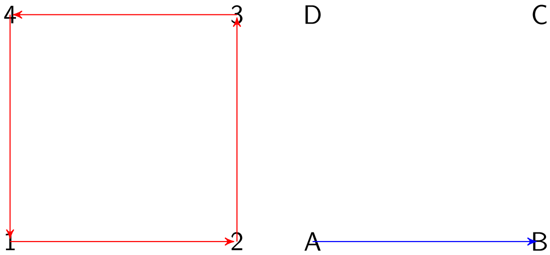
Euler Circuit

Examples of Euler Path and Circuit



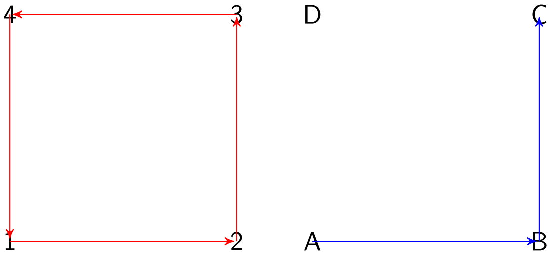
Euler Circuit

Examples of Euler Path and Circuit



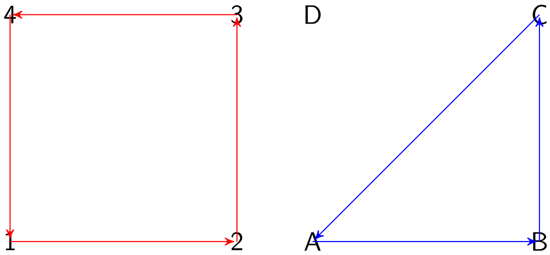
Euler Circuit

Examples of Euler Path and Circuit



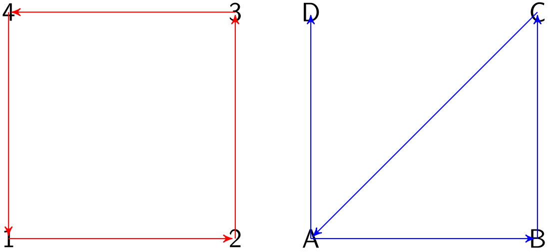
Euler Circuit

Examples of Euler Path and Circuit



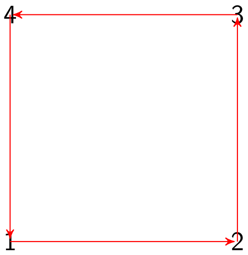
Euler Circuit

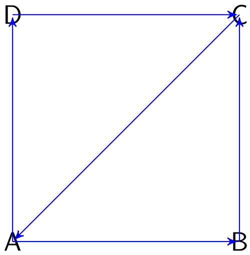
Examples of Euler Path and Circuit



Euler Circuit

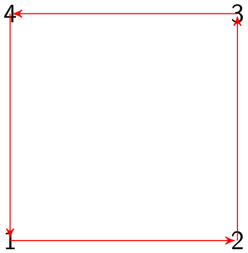
Examples of Euler Path and Circuit



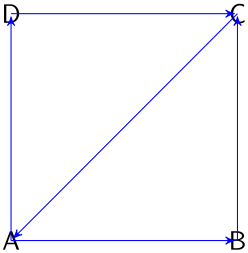


Euler Circuit

Examples of Euler Path and Circuit

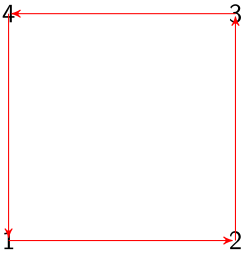


Euler Circuit

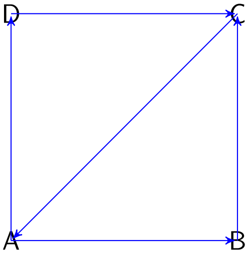


Euler Path

Examples of Euler Path and Circuit



Euler Circuit



Euler Path

Conditions for Existence

Conditions for Existence

In a connected multigraph,

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

Huynh Tuong Nguyen,
Tran Vinh Tan

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.

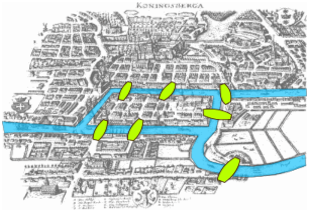
Conditions for Existence

In a connected multigraph,

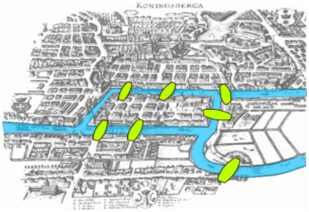
- Euler Circuit existence: **no odd-degree nodes exist** in the graph.
- Euler Path existence: **2 or no odd-degree nodes exist** in the graph.

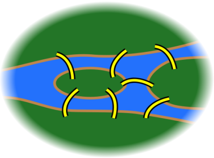
Back to the Seven Bridges Problem

Back to the Seven Bridges Problem

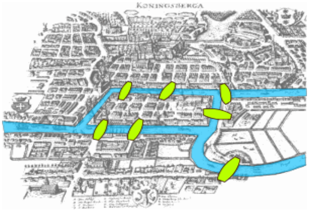


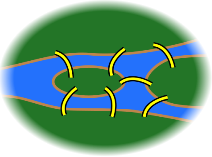
Back to the Seven Bridges Problem

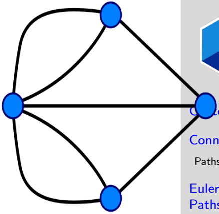




Back to the Seven Bridges Problem



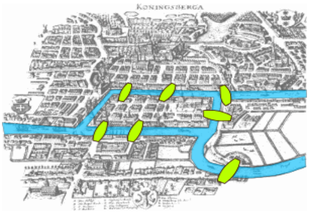


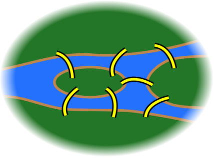


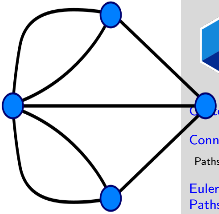
- Four vertices of odd degree

Contents

Back to the Seven Bridges Problem







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

Contents

Searching Euler Circuits and Paths – Fleury's Algorithm

Searching Euler Circuits and Paths – Fleury's Algorithm

- Choose a random vertex (if circuit) or an odd degree vertex (if path)

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

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.

Searching Euler Circuits and Paths – Hierholzer's Algorithm

Searching Euler Circuits and Paths – Hierholzer's Algorithm

- Choose a starting vertex and find a circuit

Searching Euler Circuits and Paths – Hierholzer's Algorithm

Huynh Tuong Nguyen,
Tran Vinh Tan

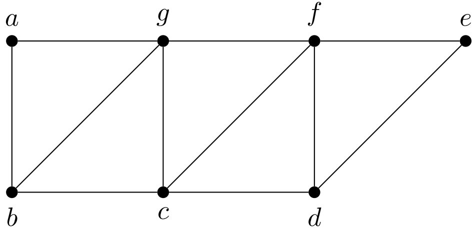
- Choose a starting vertex and find a circuit
- As long as there exists a vertex v that belongs to the current tour but that has adjacent edges not part of the tour, start another circuit from v

More efficient algorithm, $O(n)$.

Exercise

Example

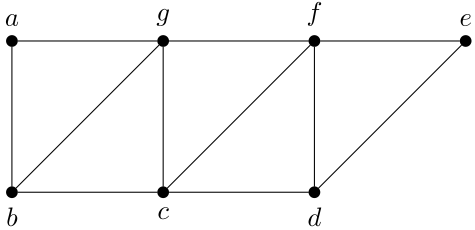
Are these following graph Euler path (circuit)? If yes, find one.

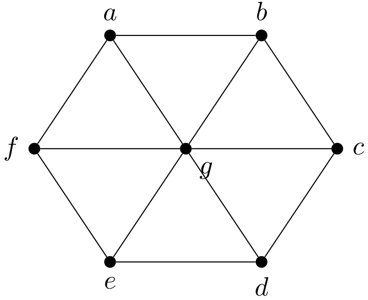


Exercise

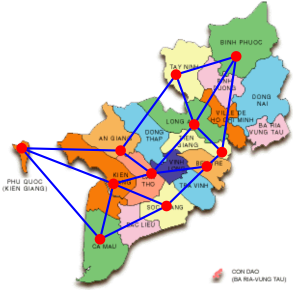
Example

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

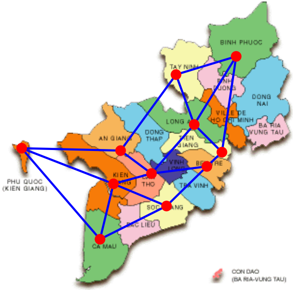




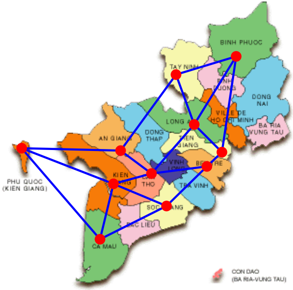
Traveling Salesman Problem



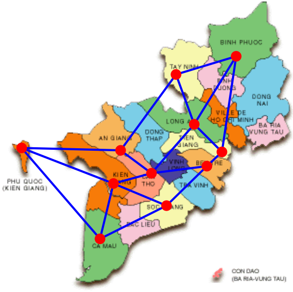
Traveling Salesman Problem



Traveling Salesman Problem



Traveling Salesman Problem



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

What Is A Hamilton Circuit?

Huynh Tuong Nguyen,
Tran Vinh Tan

What Is A Hamilton Circuit?

Definition

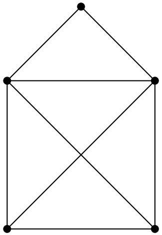
The circuit that visit each vertex in a graph once

Huynh Tuong Nguyen, Tran Vinh Tan

What Is A Hamilton Circuit?

Definition

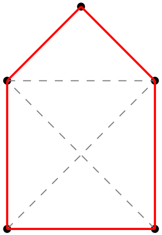
The circuit that visit each vertex in a graph once



What Is A Hamilton Circuit?

Definition

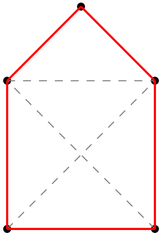
The circuit that visit each vertex in a graph once

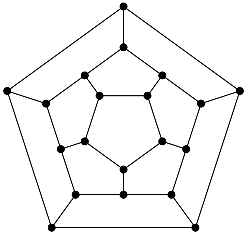


What Is A Hamilton Circuit?

Definition

The circuit that visit each vertex in a graph once

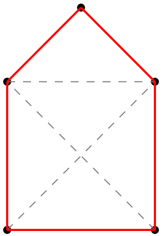


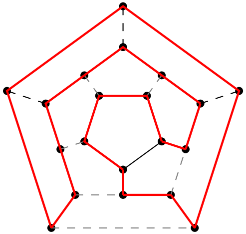


What Is A Hamilton Circuit?

Definition

The circuit that visit each vertex in a graph once

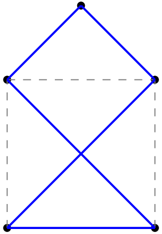


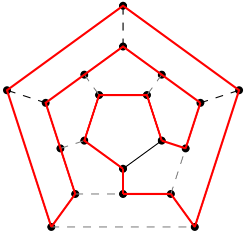


What Is A Hamilton Circuit?

Definition

The circuit that visit each vertex in a graph once





Rules of Hamilton Circuits

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

Rules of Hamilton Circuits

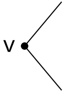
$$\deg(v) = 2 \text{ for } \forall v \text{ in Hamilton circuit!}$$

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

Rules of Hamilton Circuits

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

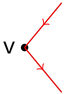
Rule 1 if $\deg(v) = 2$, both edge must be used.



Rules of Hamilton Circuits

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

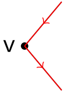
Rule 1 if $\deg(v) = 2$, both edge must be used.



Rules of Hamilton Circuits

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

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

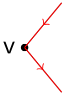


Rule 2 No subcircuit (*chu trình con*) can be formed.

Rules of Hamilton Circuits

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

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



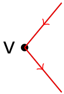
Rule 2 No subcircuit (*chu trình con*) can be formed.

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

Rules of Hamilton Circuits

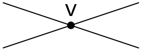
$\deg(v) = 2$ for $\forall v$ in Hamilton circuit!

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.

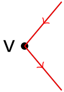


Huynh Tuong Nguyen, Tran Vinh Tan

Rules of Hamilton Circuits

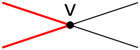
$\deg(v) = 2$ for $\forall v$ in Hamilton circuit!

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



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

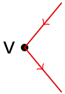
Rule 3 Once two edges at a vertex v is determined, all other edges incident at v must be removed.



Rules of Hamilton Circuits

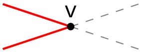
$\deg(v) = 2$ for $\forall v$ in Hamilton circuit!

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

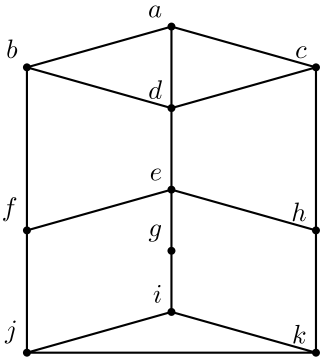


Rule 2 No subcircuit (*chu trình con*) can be formed.

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



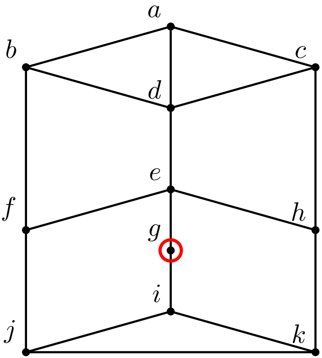
Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

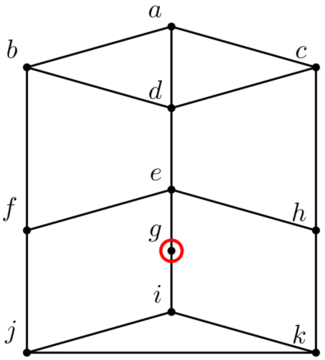
Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

Finding Hamilton Circuits



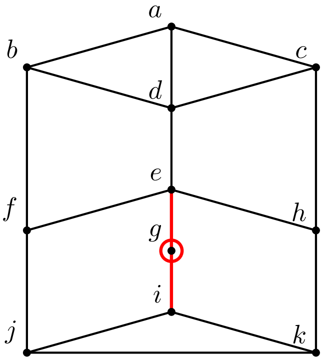
Vertices : cities

Edges : possible routes

Rule 1

$\deg(v) = 2$

Finding Hamilton Circuits



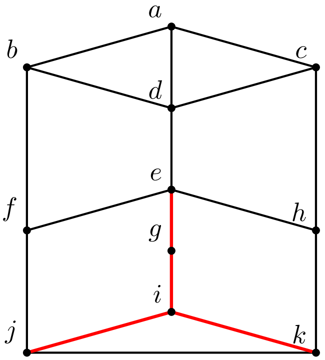
Vertices : cities

Edges : possible routes

Rule 1

$\deg(v) = 2$

Finding Hamilton Circuits



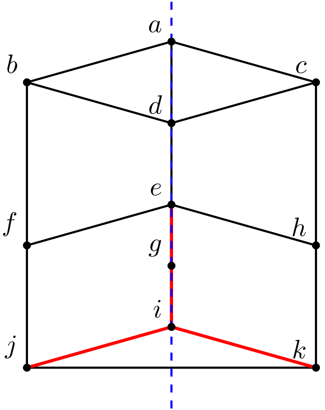
Vertices : cities

Edges : possible routes

Rule 1

$\deg(v) = 2$

Finding Hamilton Circuits



Vertices : cities

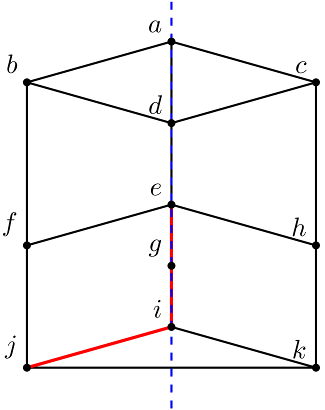
Edges : possible routes

Rule 1

$\deg(v) = 2$

Huynh Tuong Nguyen,
Tran Vinh Tan

Finding Hamilton Circuits



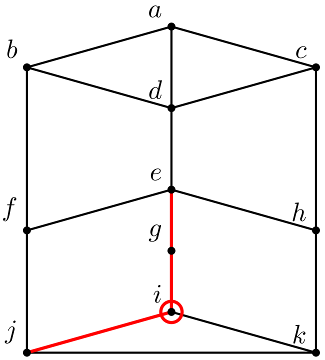
Vertices : cities

Edges : possible routes

Rule 1

$\deg(v) = 2$

Finding Hamilton Circuits



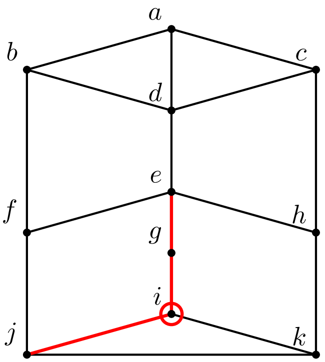
Vertices : cities

Edges : possible routes

Rule 1

$\deg(v) = 2$

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

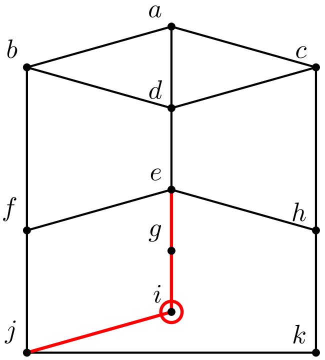
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

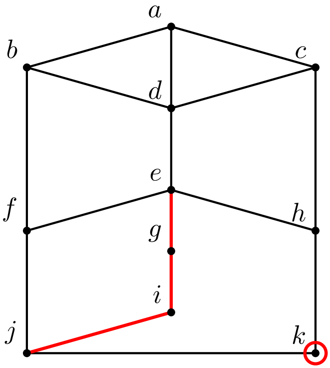
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

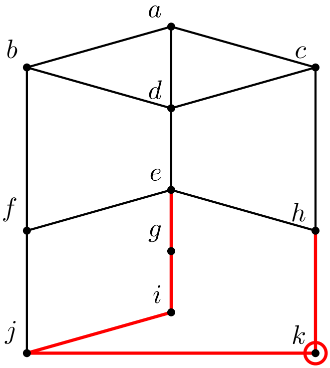
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



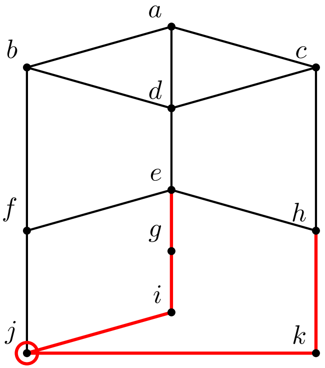
Vertices : cities

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

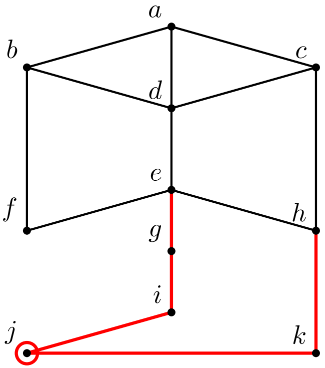
$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Huynh Tuong Nguyen, Tran Vinh Tan

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

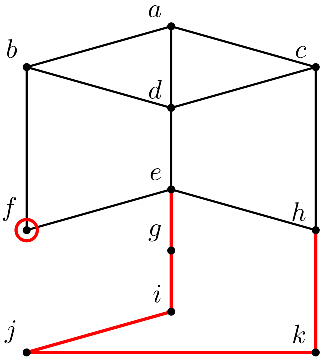
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

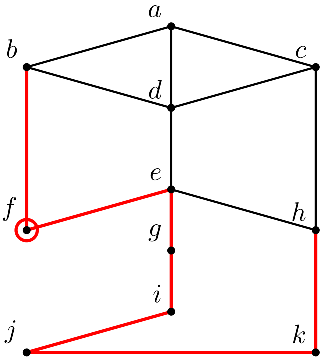
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

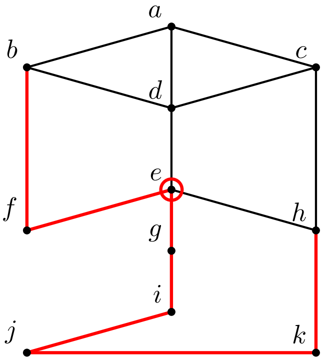
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



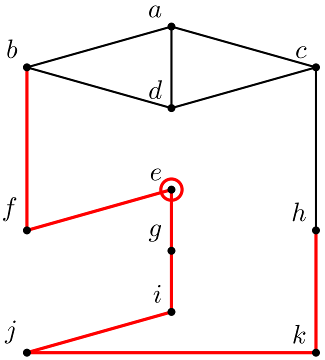
Vertices : cities

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

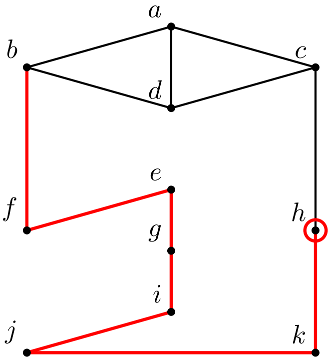
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

Rule 1

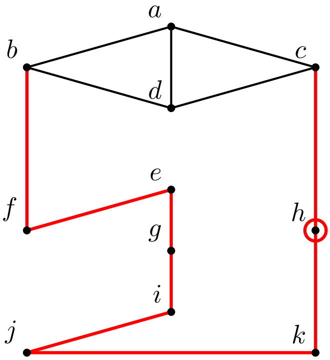
$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Huynh Tuong Nguyen, Tran Vinh Tan

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

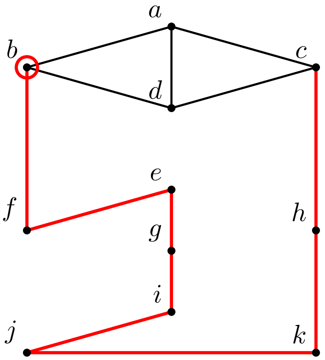
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



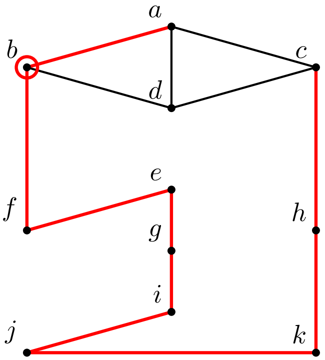
Vertices : cities

Edges : possible routes

Rule 1
$\deg(v) = 2$

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

Finding Hamilton Circuits



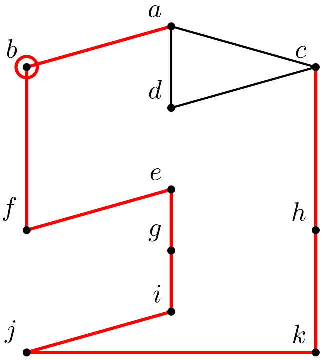
Vertices : cities

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

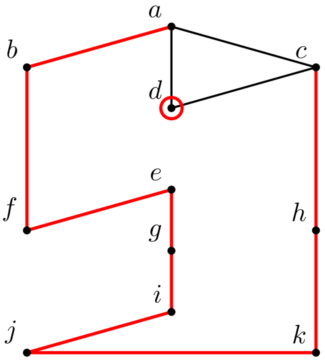
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

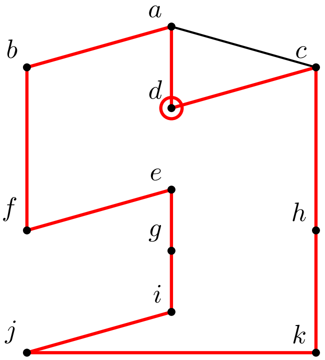
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

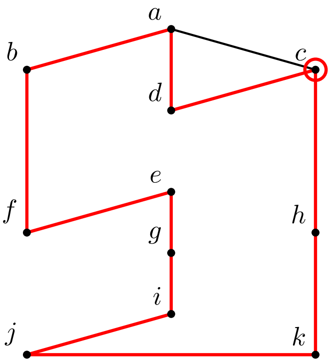
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

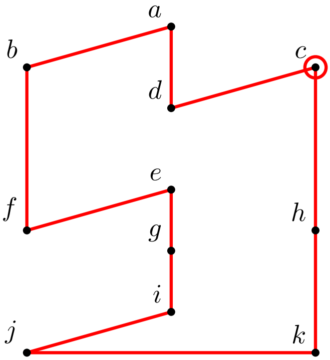
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



Vertices : cities

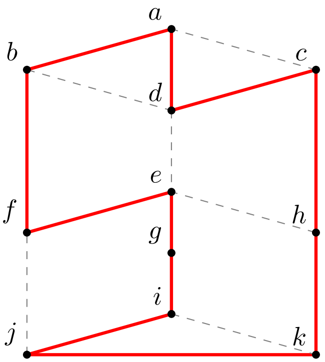
Edges : possible routes

Rule 1
$\deg(v) = 2$

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

Huynh Tuong Nguyen, Tran Vinh Tan

Finding Hamilton Circuits



We get Hamilton circuit!

Vertices : cities

Edges : possible routes

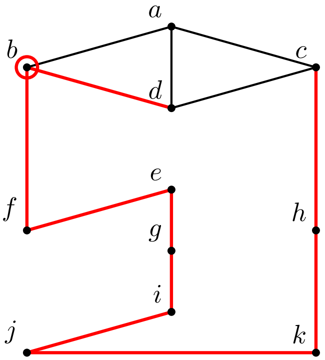
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

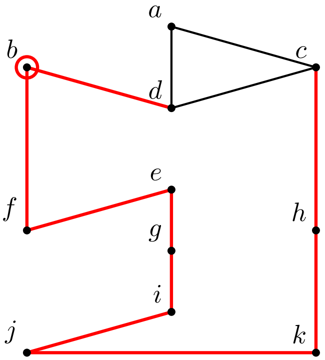
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

Rule 1

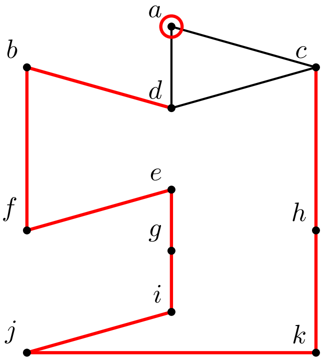
$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Huynh Tuong Nguyen, Tran Vinh Tan

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

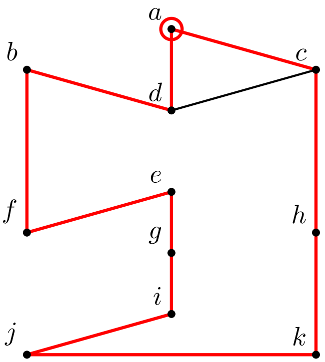
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

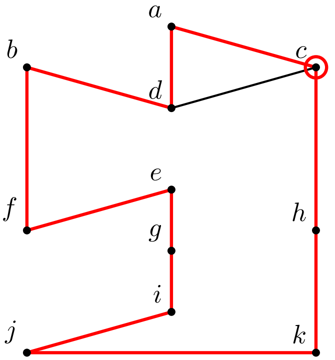
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

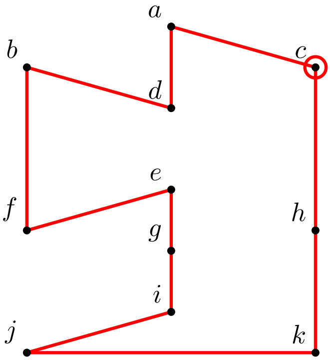
Rule 1

$\deg(v) = 2$

Rule 3

Once two edges are determined, other edges must be removed

Finding Hamilton Circuits



Vertices : cities

Edges : possible routes

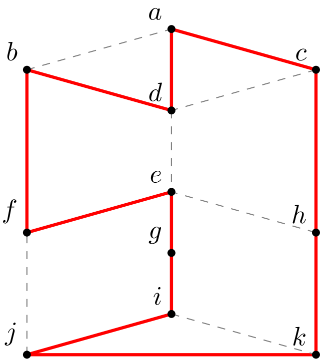
Rule 1

$\deg(v) = 2$

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

Existence of Hamilton Circuit

Hamilton circuit does not exist for all graph.

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.

Existence of Hamilton Circuit

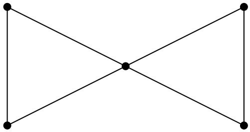
Hamilton circuit does not exist for all graph. But, there is no specific way to find whether Hamilton circuit exists or not.

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.

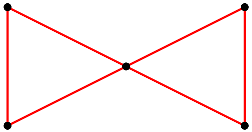
Simple check by rules of Hamilton circuit



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.

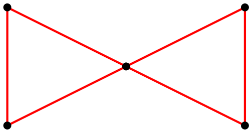
Simple check by rules of Hamilton circuit



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.

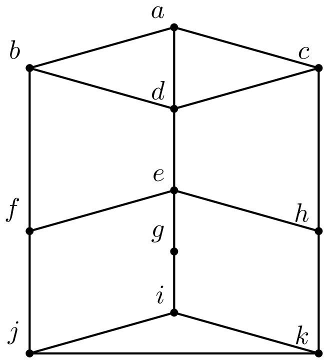
Simple check by rules of Hamilton circuit



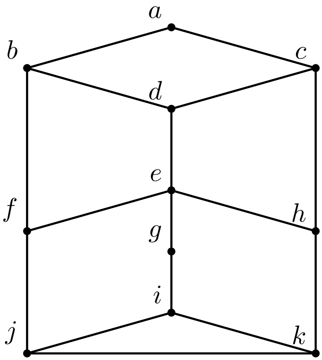
Violates Rule 2! (No subcircuit)

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

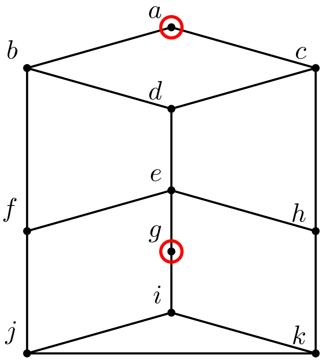
We can verify nonexistence of the graph during find Hamilton circuit.



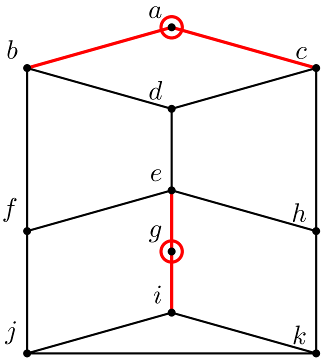
We can verify nonexistence of the graph during find Hamilton circuit.



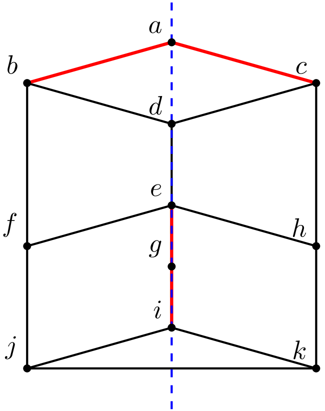
We can verify nonexistence of the graph during find Hamilton circuit.



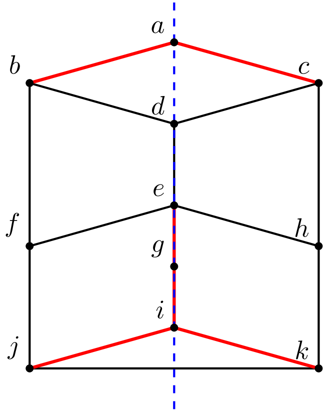
We can verify nonexistence of the graph during find Hamilton circuit.



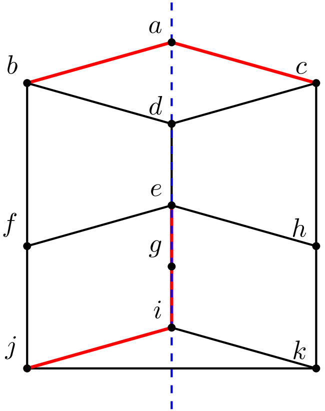
We can verify nonexistence of the graph during find Hamilton circuit.



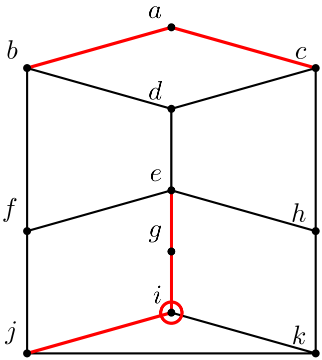
We can verify nonexistence of the graph during find Hamilton circuit.



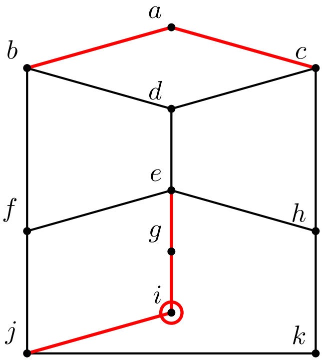
We can verify nonexistence of the graph during find Hamilton circuit.



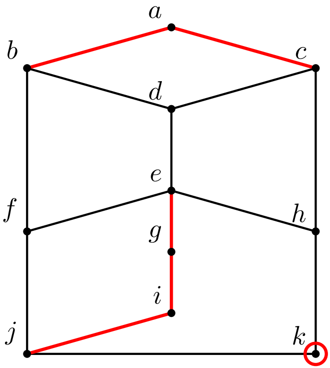
We can verify nonexistence of the graph during find Hamilton circuit.



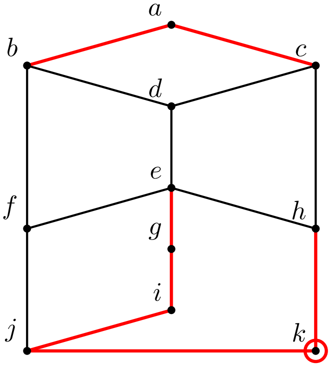
We can verify nonexistence of the graph during find Hamilton circuit.



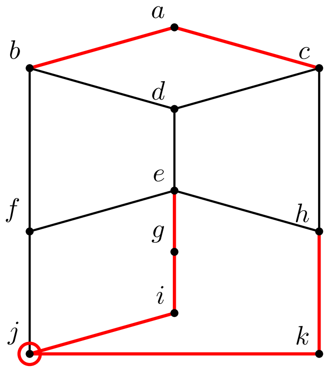
We can verify nonexistence of the graph during find Hamilton circuit.



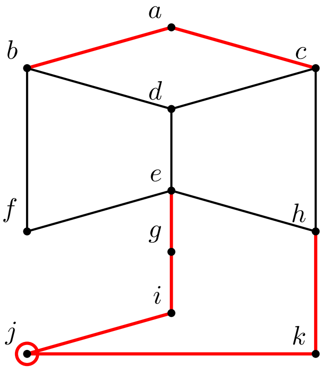
We can verify nonexistence of the graph during find Hamilton circuit.



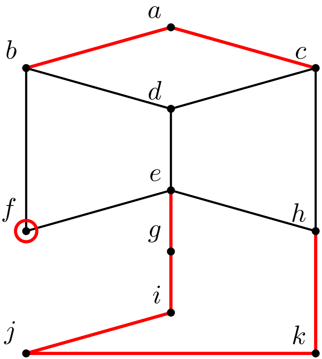
We can verify nonexistence of the graph during find Hamilton circuit.



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

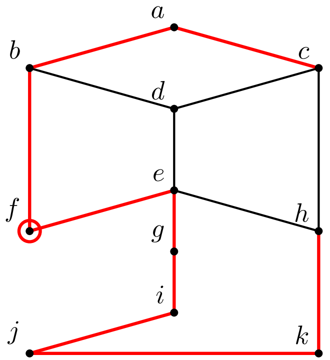


We can verify nonexistence of the graph during find Hamilton circuit.

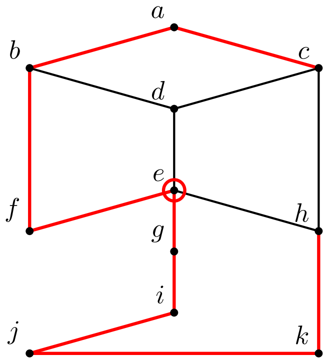


Huynh Tuong Nguyen, Tran Vinh Tan

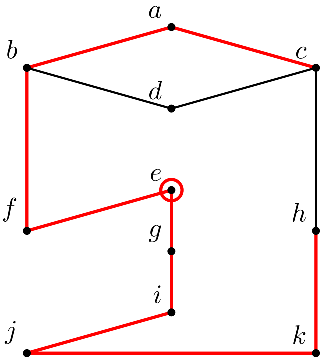
We can verify nonexistence of the graph during find Hamilton circuit.



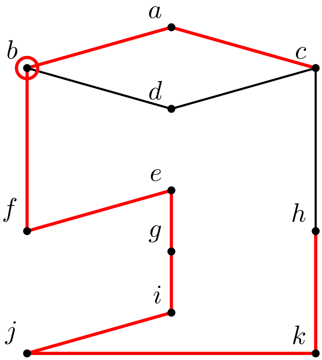
We can verify nonexistence of the graph during find Hamilton circuit.



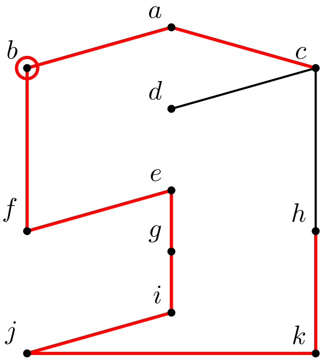
We can verify nonexistence of the graph during find Hamilton circuit.



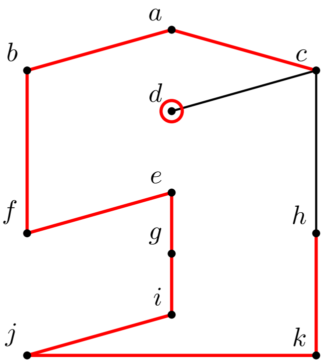
We can verify nonexistence of the graph during find Hamilton circuit.



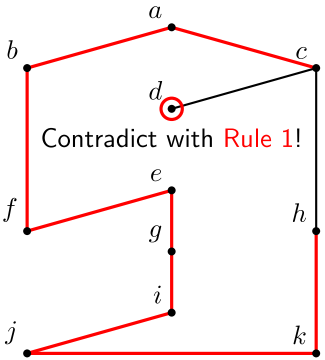
We can verify nonexistence of the graph during find Hamilton circuit.



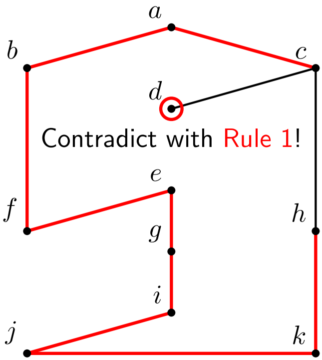
We can verify nonexistence of the graph during find Hamilton circuit.



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



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



Hamilton circuit doesn't exist!

Application – Gray Code

Definition

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

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
$\vdots$		$\vdots$	$\vdots$

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
⋮		⋮	⋮

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

Gray Code

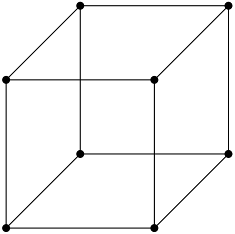
n-digit gray code can be generated by finding Hamilton circuits of n-dimensional hypercube!

Gray Code

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

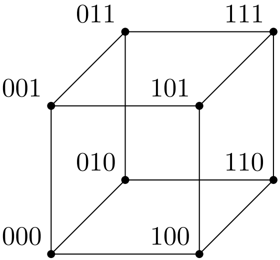
Gray Code

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



Gray Code

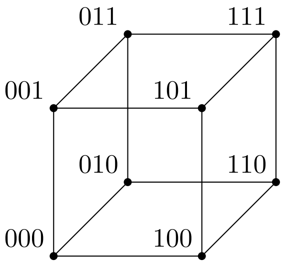
n-digit gray code can be generated by finding Hamilton circuits of n-dimensional hypercube! Consider the case $n = 3$.



Coordinate of each vertex is 3-digit binary sequences.

Gray Code

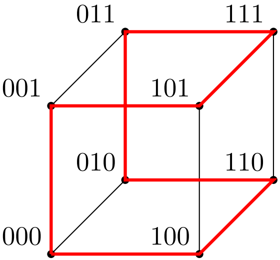
n-digit gray code can be generated by finding Hamilton circuits of n-dimensional hypercube! Consider the case $n = 3$.



Coordinate of each vertex is 3-digit binary sequences. Coordinates of adjacent vertices differ in just on place.

Gray Code

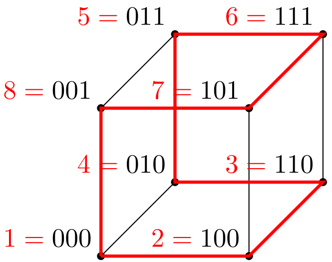
n-digit gray code can be generated by finding Hamilton circuits of n-dimensional hypercube! Consider the case $n = 3$.



Coordinate of each vertex is 3-digit binary sequences. Coordinates of adjacent vertices differ in just on place.

Gray Code

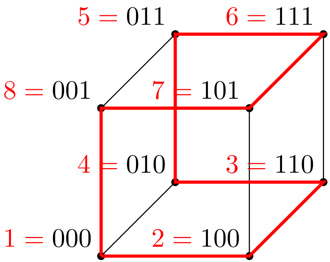
n-digit gray code can be generated by finding Hamilton circuits of n-dimensional hypercube! Consider the case $n = 3$.



Coordinate of each vertex is 3-digit binary sequences. Coordinates of adjacent vertices differ in just on place.

Gray Code

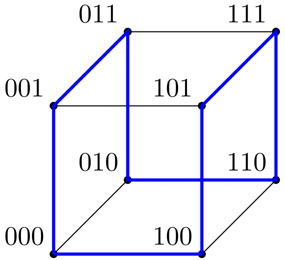
n-digit gray code can be generated by finding Hamilton circuits of n-dimensional hypercube! Consider the case $n = 3$.



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!

Gray Code

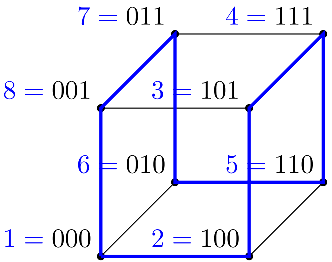
n-digit gray code can be generated by finding Hamilton circuits of n-dimensional hypercube! Consider the case $n = 3$.



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!

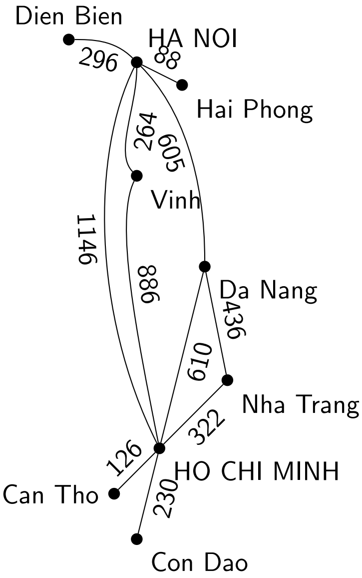
Gray Code

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!

Weighted Graphs



Problem

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

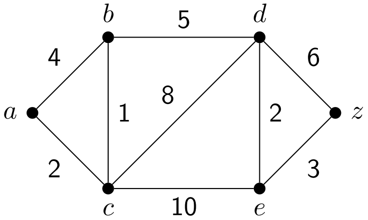
- The single-source shortest path problem, in which we have to find shortest paths from a source vertex v to all other vertices in the graph.
- The single-destination shortest path problem, in which we have to find shortest paths from all vertices in the graph to a single destination vertex v. This can 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

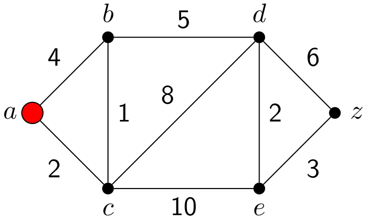
```
procedure Dijkstra(G,a)
// Initialization Step
  forall vertices v
     Label[v] := ∞
     Prev[v]  := -1
  endfor
  Label(a) := 0 // a is the source node
  S := ∅

// 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



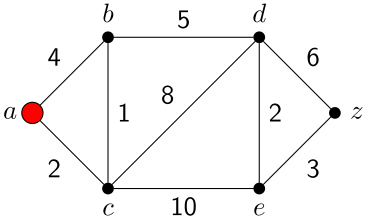
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞

Example



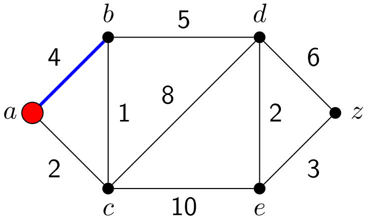
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞

Example



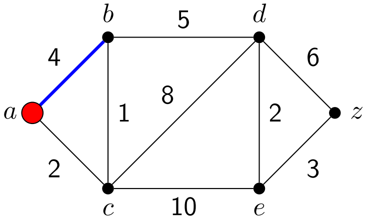
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0					

Example



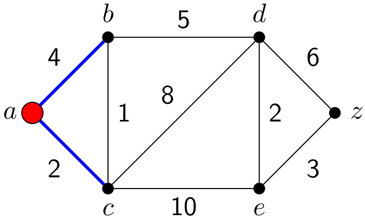
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0					

Example



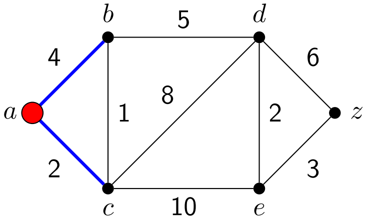
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4				

Example



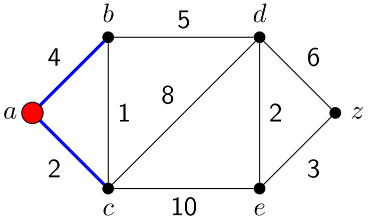
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4				

Example



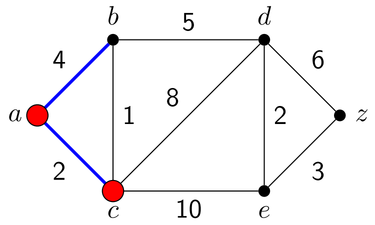
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	2			

Example



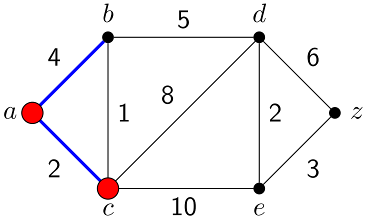
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	2	∞	∞	∞

Example



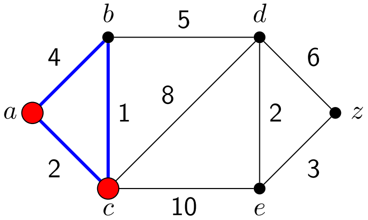
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞

Example



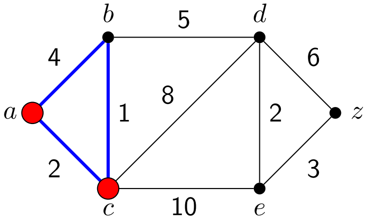
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0		2			

Example



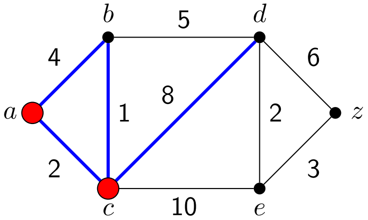
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0		2			

Example



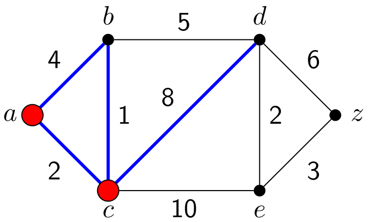
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	3	2			

Example



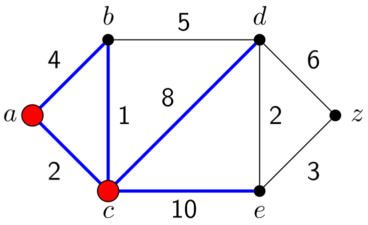
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	3	2			

Example



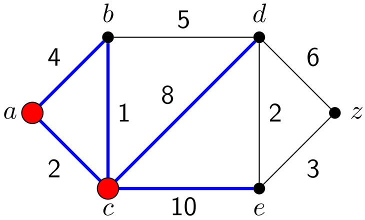
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	3	2	10		

Example



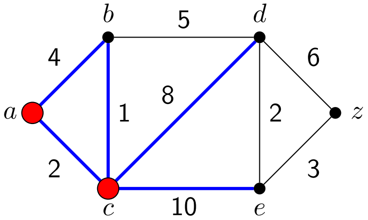
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	3	2	10		

Example



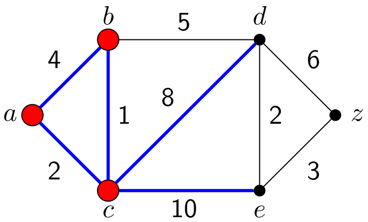
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	3	2	10	12	

Example



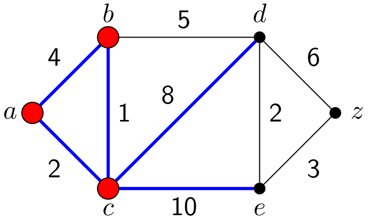
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	3	2	10	12	∞

Example



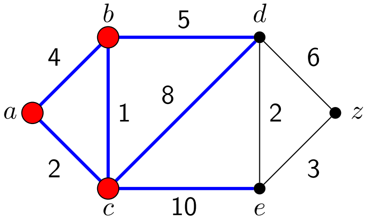
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞

Example



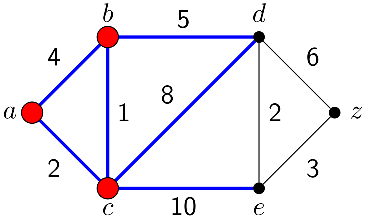
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2			

Example



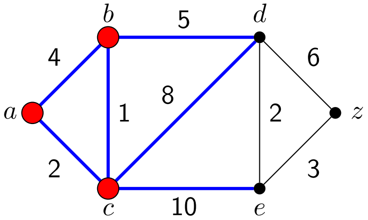
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2			

Example



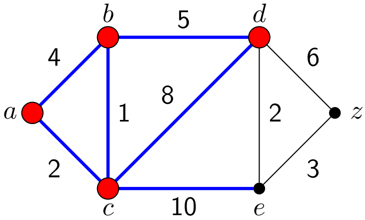
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2	8		

Example



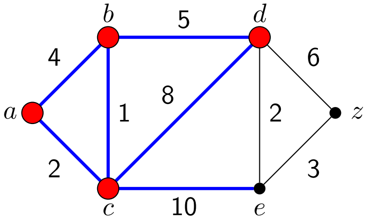
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2	8	12	∞

Example



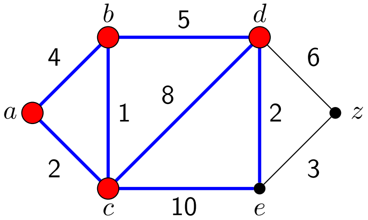
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2	**8**	12	∞

Example



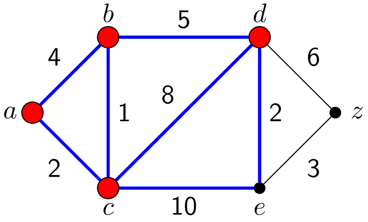
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2	**8**	12	∞
d	0	3	2	8		

Example



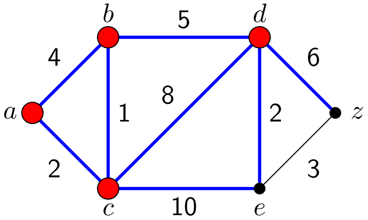
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2	**8**	12	∞
d	0	3	2	8		

Example



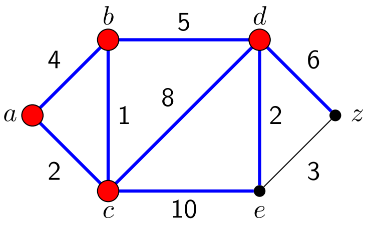
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2	**8**	12	∞
d	0	3	2	8	10	

Example



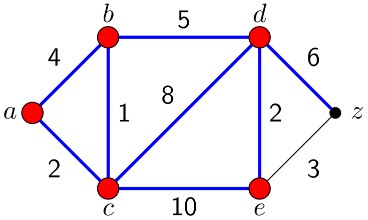
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2	**8**	12	∞
d	0	3	2	8	10	

Example



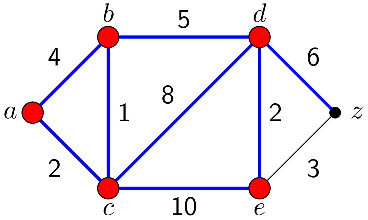
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2	**8**	12	∞
d	0	3	2	8	10	14

Example



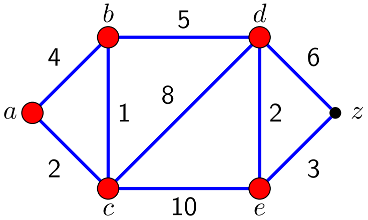
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2	**8**	12	∞
d	0	3	2	8	**10**	14

Example



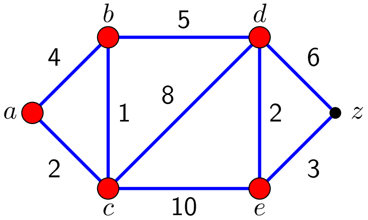
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2	**8**	12	∞
d	0	3	2	8	**10**	14
e	0	3	2	8	10	

Example



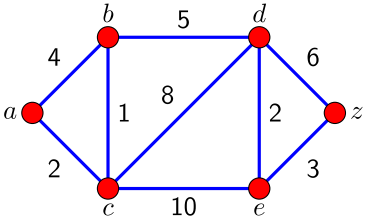
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2	**8**	12	∞
d	0	3	2	8	**10**	14
e	0	3	2	8	10	

Example



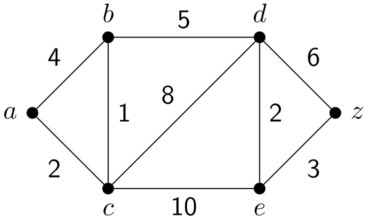
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2	**8**	12	∞
d	0	3	2	8	**10**	14
e	0	3	2	8	10	13

Example



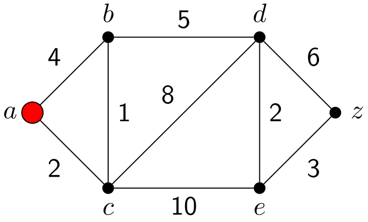
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a	0	4	**2**	∞	∞	∞
c	0	**3**	2	10	12	∞
b	0	3	2	**8**	12	∞
d	0	3	2	8	**10**	14
e	0	3	2	8	10	**13**

Example



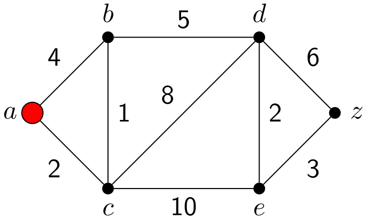
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞

Example



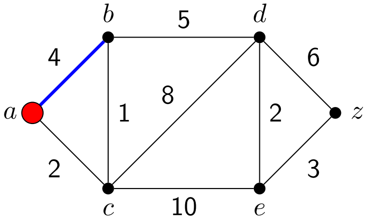
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞

Example



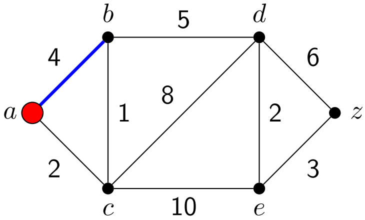
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a						

Example



S	a	b	c	d	e	z
$\emptyset$	$\underline{\mathbf{0}}$	∞	∞	∞	∞	∞
a						

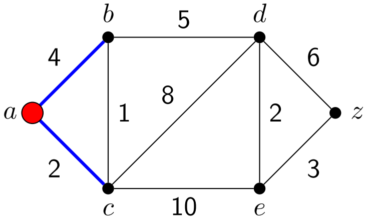
Example



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

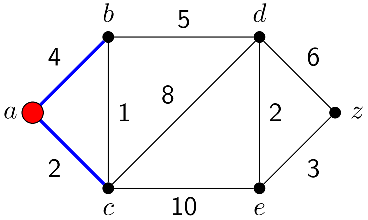
Huynh Tuong Nguyen, Tran Vinh Tan

Example



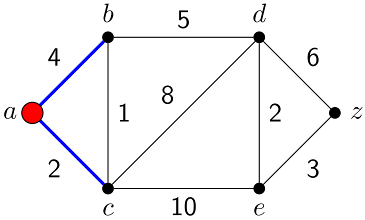
S	a	b	c	d	e	z
$\emptyset$	$\underline{\mathbf{0}}$	∞	∞	∞	∞	∞
a		4				

Example



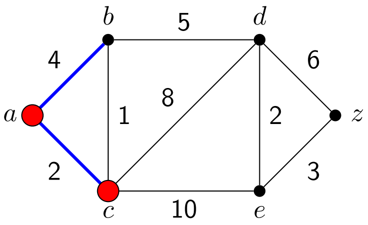
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a		4	2			

Example



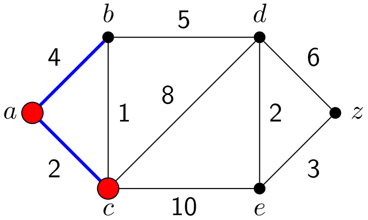
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a		4	2	∞	∞	∞

Example



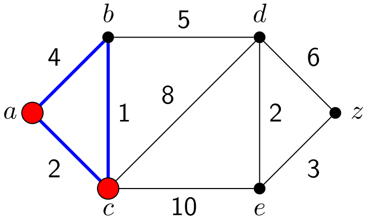
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a		4	**2**	∞	∞	∞

Example



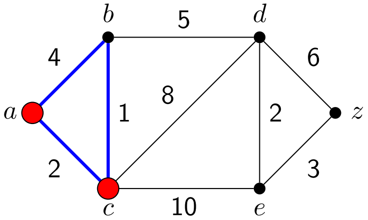
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a		4	**2**	∞	∞	∞
c						

Example



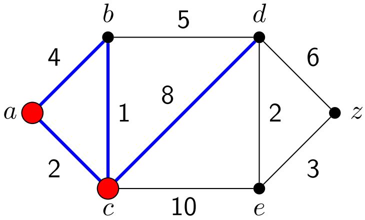
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a		4	**2**	∞	∞	∞
c						

Example



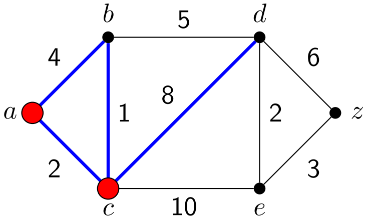
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a		4	**2**	∞	∞	∞
c		3				

Example



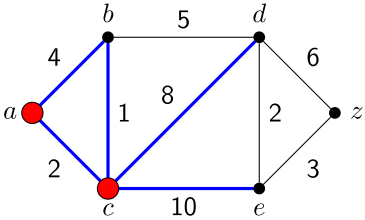
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a		4	**2**	∞	∞	∞
c		3				

Example



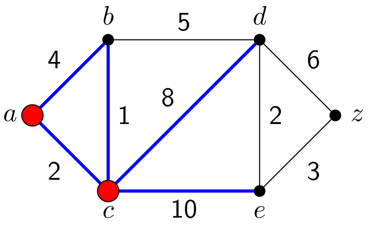
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a		4	**2**	∞	∞	∞
c		3		10		

Example



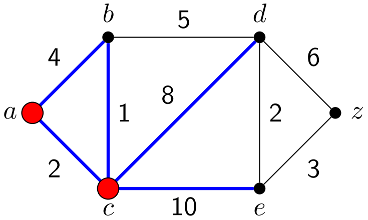
S	a	b	c	d	e	z
$\emptyset$	$\underline{\mathbf{0}}$	∞	∞	∞	∞	∞
a		4	$\underline{\mathbf{2}}$	∞	∞	∞
c		3		10		

Example



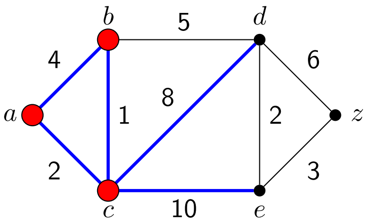
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a		4	**2**	∞	∞	∞
c		3		10	12	

Example



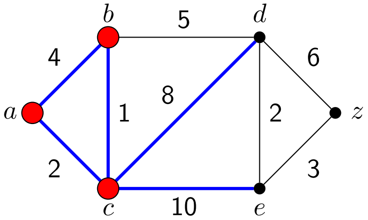
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a		4	**2**	∞	∞	∞
c		3		10	12	∞

Example



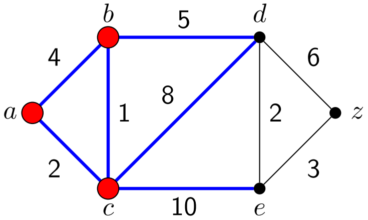
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a		4	**2**	∞	∞	∞
c		**3**		10	12	∞

Example



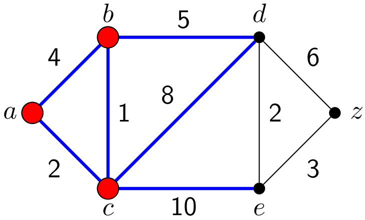
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a		4	**2**	∞	∞	∞
c		**3**		10	12	∞
b						

Example



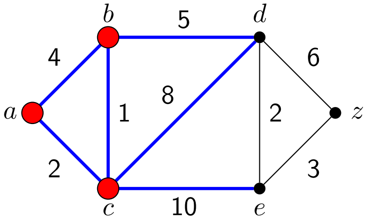
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a		4	**2**	∞	∞	∞
c		**3**		10	12	∞
b						

Example



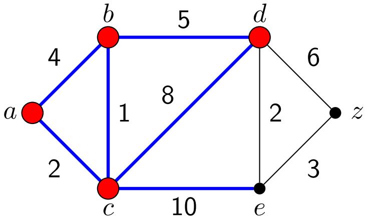
S	a	b	c	d	e	z
$\emptyset$	**$\underline{0}$**	∞	∞	∞	∞	∞
a		4	**$\underline{2}$**	∞	∞	∞
c		**$\underline{3}$**		10	12	∞
b				8		

Example



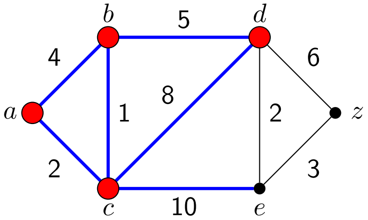
S	a	b	c	d	e	z
$\emptyset$	$\underline{\mathbf{0}}$	∞	∞	∞	∞	∞
a		4	$\underline{\mathbf{2}}$	∞	∞	∞
c		$\underline{\mathbf{3}}$		10	12	∞
b				8	12	∞

Example



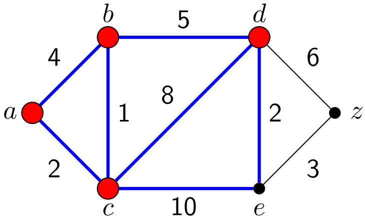
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a		4	**2**	∞	∞	∞
c		**3**		10	12	∞
b				**8**	12	∞

Example



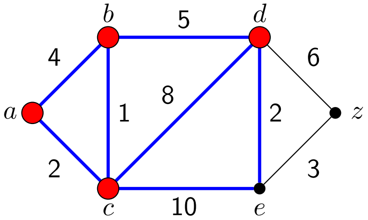
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a		4	**2**	∞	∞	∞
c		**3**		10	12	∞
b				**8**	12	∞
d						

Example



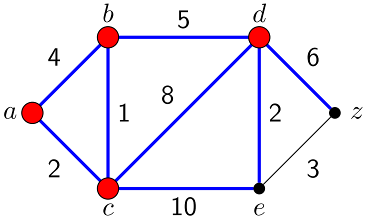
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a		4	**2**	∞	∞	∞
c		**3**		10	12	∞
b				**8**	12	∞
d						

Example



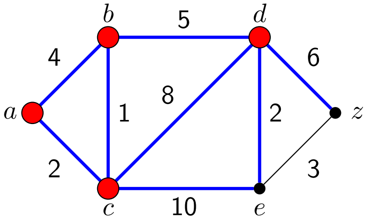
S	a	b	c	d	e	z
$\emptyset$	$\underline{\mathbf{0}}$	∞	∞	∞	∞	∞
a		4	$\underline{\mathbf{2}}$	∞	∞	∞
c		$\underline{\mathbf{3}}$		10	12	∞
b				$\underline{\mathbf{8}}$	12	∞
d					10	

Example



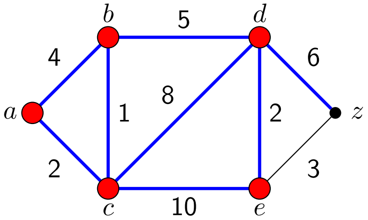
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a		4	**2**	∞	∞	∞
c		**3**		10	12	∞
b				**8**	12	∞
d					10	

Example



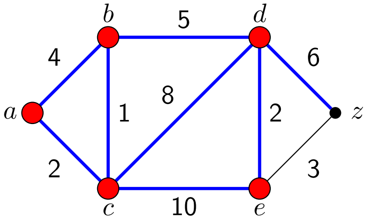
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a		4	**2**	∞	∞	∞
c		**3**		10	12	∞
b				**8**	12	∞
d					10	14

Example



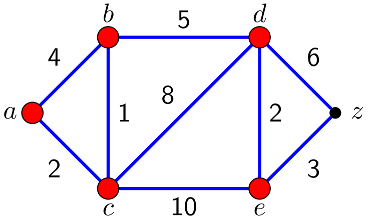
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a		4	**2**	∞	∞	∞
c		**3**		10	12	∞
b				**8**	12	∞
d					**10**	14

Example



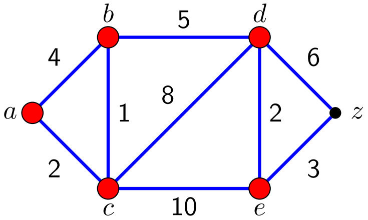
S	a	b	c	d	e	z
$\emptyset$	**<u>0</u>**	∞	∞	∞	∞	∞
a		4	**<u>2</u>**	∞	∞	∞
c		**<u>3</u>**		10	12	∞
b				**<u>8</u>**	12	∞
d					**<u>10</u>**	14
e						

Example



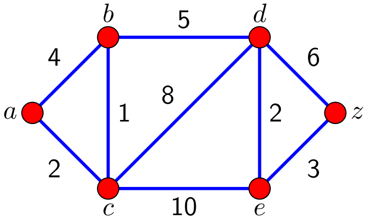
S	a	b	c	d	e	z
$\emptyset$	**<u>0</u>**	∞	∞	∞	∞	∞
a		4	**<u>2</u>**	∞	∞	∞
c		**<u>3</u>**		10	12	∞
b				**<u>8</u>**	12	∞
d					**<u>10</u>**	14
e						

Example



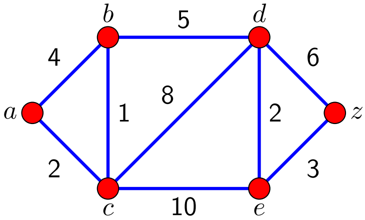
S	a	b	c	d	e	z
$\emptyset$	**0**	∞	∞	∞	∞	∞
a		4	**2**	∞	∞	∞
c		**3**		10	12	∞
b				**8**	12	∞
d					**10**	14
e						13

Example



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

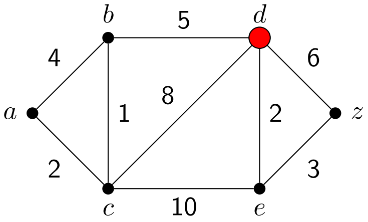
Example



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

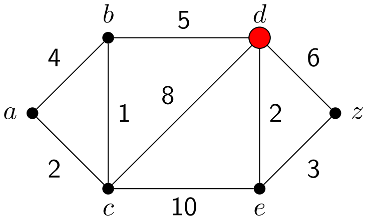
How to determine shortest path from a to d according to Dijkstra's alogrithm?



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

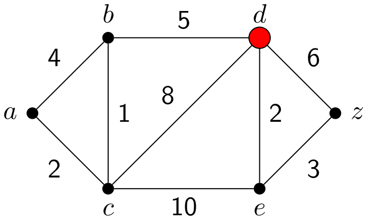
How to determine shortest path from a to d according to Dijkstra's alogrithm?



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

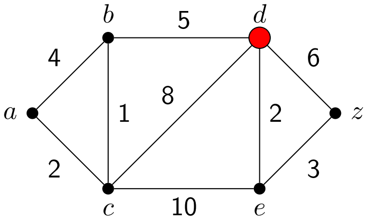
How to determine shortest path from a to d according to Dijkstra's alogrithm?



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

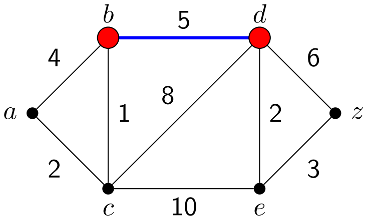
How to determine shortest path from a to d according to Dijkstra's alogrithm?



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

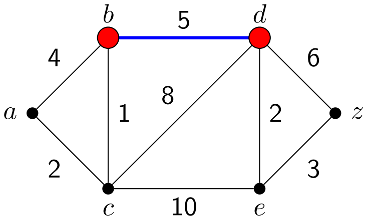
How to determine shortest path from a to d according to Dijkstra's alogrithm?



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

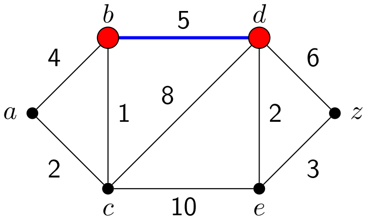
How to determine shortest path from a to d according to Dijkstra's alogrithm?



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

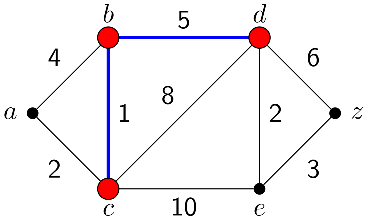
How to determine shortest path from a to d according to Dijkstra's alogrithm?



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

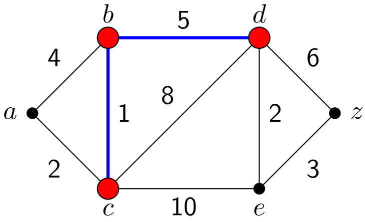
How to determine shortest path from a to d according to Dijkstra's alogrithm?



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	$\mathbf{3}$	2	$\underline{\mathbf{8}}$	12	∞
d	0	3	2	$\mathbf{8}$	$\underline{10}$	14
e	0	3	2	$\mathbf{8}$	10	$\underline{13}$

Back tracking procedure

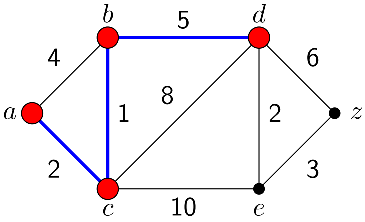
How to determine shortest path from a to d according to Dijkstra's alogrithm?



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 alogrithm?



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

Dijkstra's Algorithm

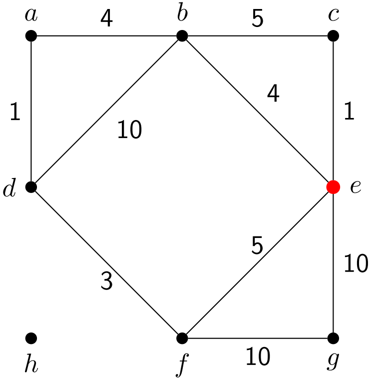
Property

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

Exercise

Example

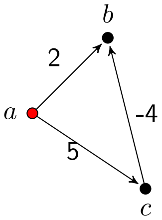
Find the shortest path from e to other vertices using Dijkstra's algorithm.



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"
```

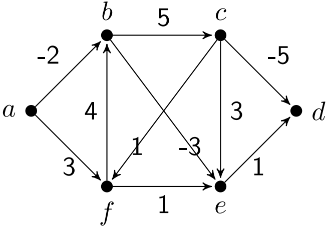
Property

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

Example

Example

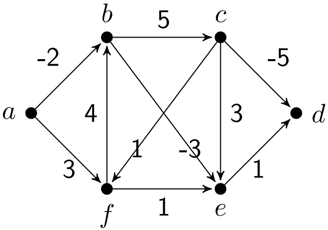
Step	a	b	c	d	e	f



Example

Example

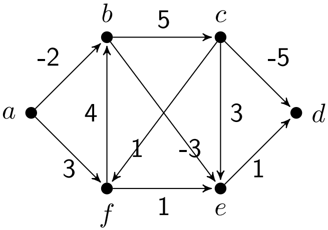
Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞



Example

Example

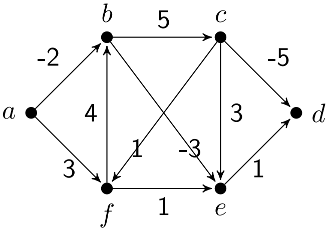
Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	-2a	∞	∞	∞	3a



Example

Example

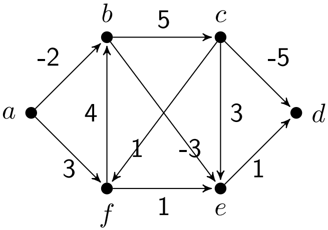
Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	-2a	∞	∞	∞	3a
2	0	-2	3b	∞	-5b	3



Example

Example

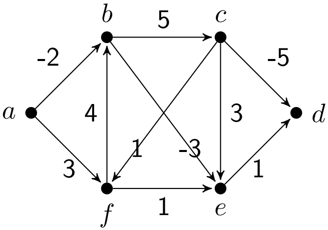
Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	-2a	∞	∞	∞	3a
2	0	-2	3b	∞	-5b	3
3	0	-2	3	-4e	-5	3



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

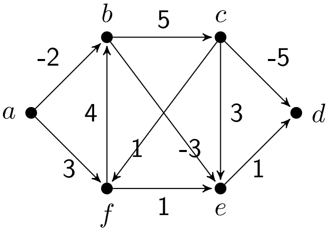


Example

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.

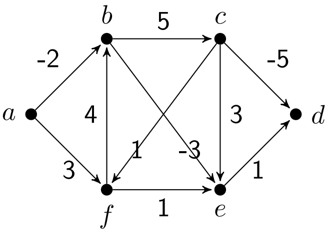


Backtracking procedure

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.



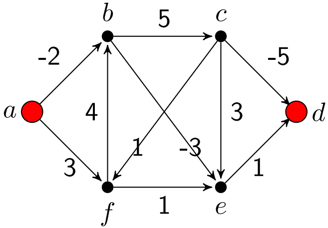
Backtracking procedure

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? $a \rightarrow \qquad d$



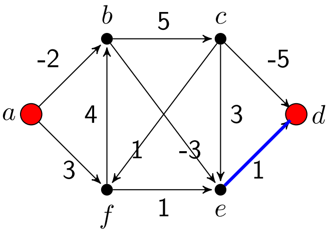
Backtracking procedure

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? $a \to \quad e \to d$



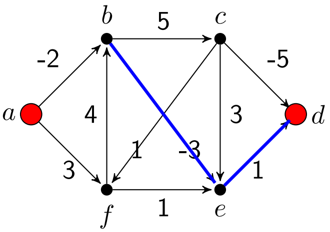
Backtracking procedure

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? $a \to b \to e \to d$



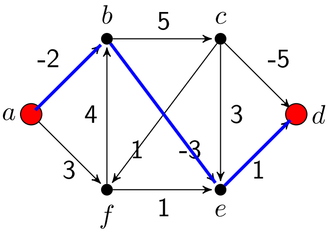
Backtracking procedure

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.

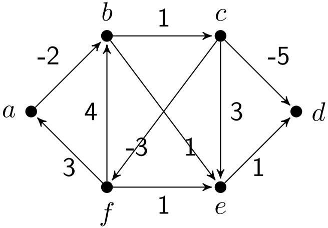
How to find shortest path from a to d? $a \to b \to e \to d$



Example

Example

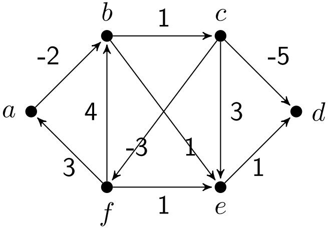
Step	a	b	c	d	e	f



Example

Example

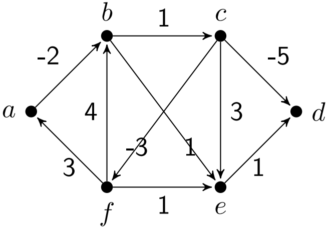
Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞



Example

Example

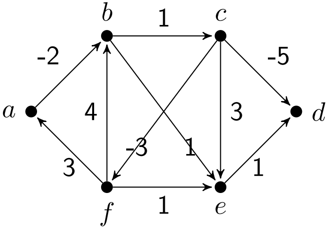
Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	-2a	∞	∞	∞	∞



Example

Example

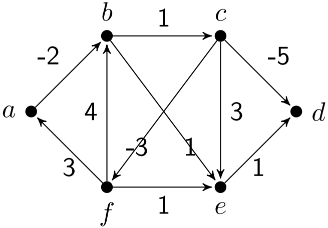
Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	-2a	∞	∞	∞	∞
2	0	-2	-1b	∞	-1b	∞



Example

Example

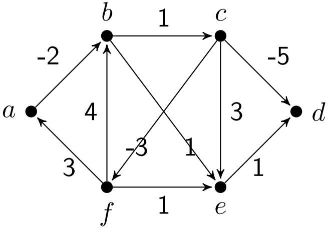
Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	-2a	∞	∞	∞	∞
2	0	-2	-1b	∞	-1b	∞
3	0	-2	-1	-6c	-1	-4c



Example

Example

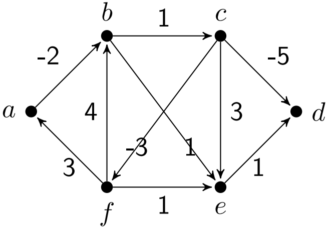
Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	-2a	∞	∞	∞	∞
2	0	-2	-1b	∞	-1b	∞
3	0	-2	-1	-6c	-1	-4c
4	-1f	-2	-1	-6	-3f	-4



Example

Example

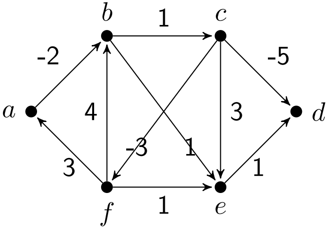
Step	a	b	c	d	e	f
0	0	∞	∞	∞	∞	∞
1	0	-2a	∞	∞	∞	∞
2	0	-2	-1b	∞	-1b	∞
3	0	-2	-1	-6c	-1	-4c
4	-1f	-2	-1	-6	-3f	-4
5	-1	-3a	-1	-6	-3	-4



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

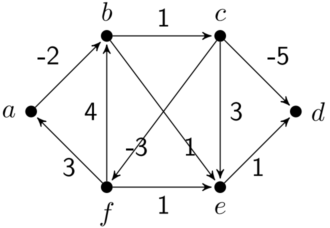


Example

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.

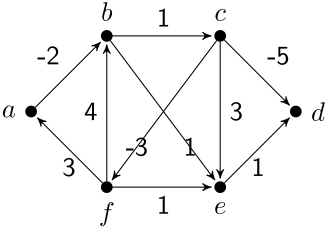


Example

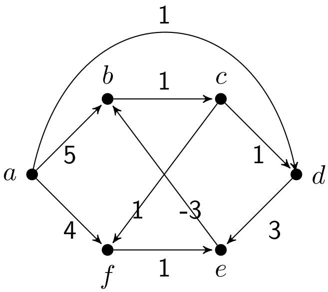
Example

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

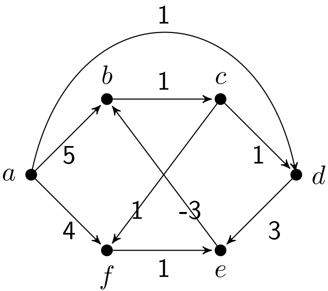
There exists a circle of negative length since Step 6 $\neq$ Step 5.



Exercise



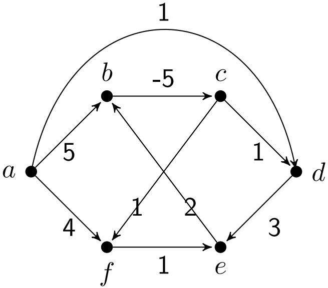
Exercise



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



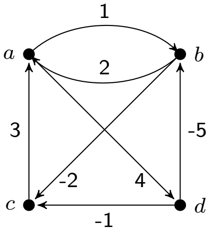
Floyd-Warshall Algorithm [1962]

```
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

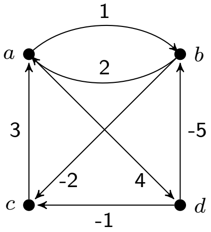
any G, any length; all-to-all; this is an software algorithm; complexity $O(|V|^3)$.

Example



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

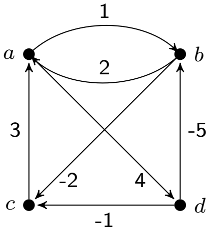
Example



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

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

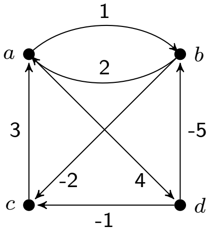
Example



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

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

Example

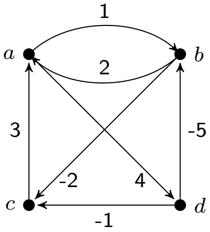


$$L^{(0)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & \infty_0 \\ 3_0 & \infty_0 & 0_0 & \infty_0 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

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

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

Example

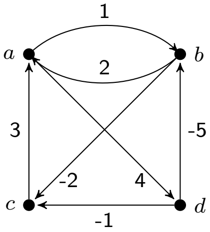


$$L^{(0)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & \infty_0 \\ 3_0 & \infty_0 & 0_0 & \infty_0 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

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

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

Example



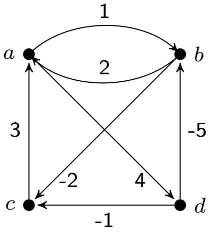
$$L^{(0)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & \infty_0 \\ 3_0 & \infty_0 & 0_0 & \infty_0 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

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

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

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

Example



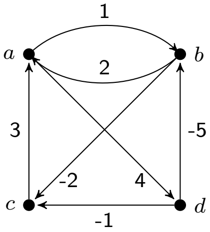
$$L^{(0)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & \infty_0 \\ 3_0 & \infty_0 & 0_0 & \infty_0 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

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

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

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

Example



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

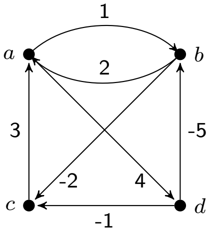
$$L^{(1)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & 6_1 \\ 3_0 & 4_1 & 0_0 & 7_1 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

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

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

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

Example



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

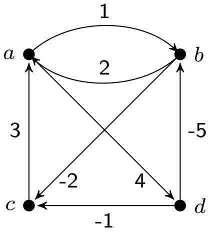
$$L^{(1)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -\ 2_0 & 6_1 \\ 3_0 & 4_1 & 0_0 & 7_1 \\ \infty_0 & -\ 5_0 & -\ 1_0 & 0_0 \end{pmatrix}$$

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

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

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

Example



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

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

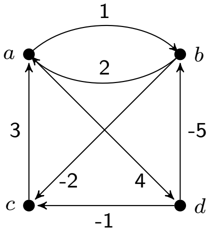
$$L^{(1)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & 6_1 \\ 3_0 & 4_1 & 0_0 & 7_1 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

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

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

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

Example



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

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

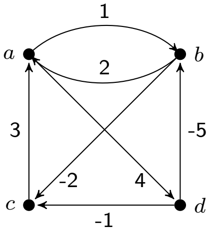
$$L^{(1)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & 6_1 \\ 3_0 & 4_1 & 0_0 & 7_1 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

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

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

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

Example



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 $L^{(1)}$)

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

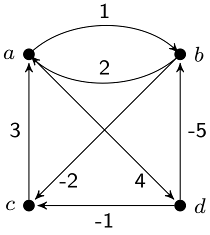
$$L^{(1)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & 6_1 \\ 3_0 & 4_1 & 0_0 & 7_1 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

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

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

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

Example



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 $L^{(1)}$)
$\Rightarrow bd = bc + ca + ad$

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

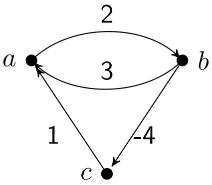
$$L^{(1)} = \begin{pmatrix} 0_0 & 1_0 & \infty_0 & 4_0 \\ 2_0 & 0_0 & -2_0 & 6_1 \\ 3_0 & 4_1 & 0_0 & 7_1 \\ \infty_0 & -5_0 & -1_0 & 0_0 \end{pmatrix}$$

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

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

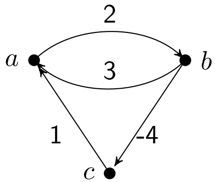
$$L^{(4)} = \begin{pmatrix} & & & 4_0 \\ & & & 5_3 \\ & & & 7_1 \\ -4_3 & -5_0 & -7_2 & 0_0 \end{pmatrix}$$

Example



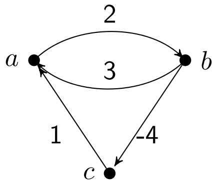
$$L^{(0)} = \begin{pmatrix} 0_0 & 2_0 & \infty_0 \\ 3_0 & 0_0 & -4_0 \\ 1_0 & \infty_0 & 0_0 \end{pmatrix}$$

Example



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

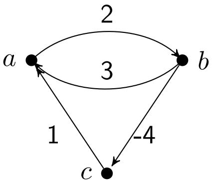
Example



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

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

Example

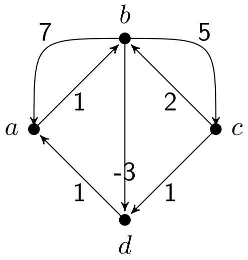


$$L^{(0)} = \begin{pmatrix} 0_0 & 2_0 & \infty_0 \\ 3_0 & 0_0 & -4_0 \\ 1_0 & \infty_0 & 0_0 \end{pmatrix} \quad L^{(1)} = \begin{pmatrix} 0_0 & 2_0 & \infty_0 \\ 3_0 & 0_0 & -4_0 \\ 1_0 & 3_1 & 0_0 \end{pmatrix}$$

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

STOP, there exists a circuit of negative length.

Exercise



Ford's algorithm

$\pi(1) = 0$

For each $j \in V$ do

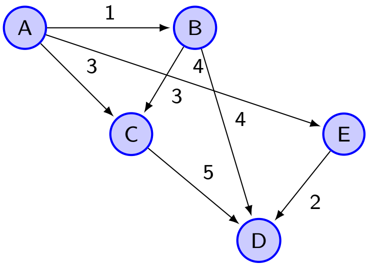
$\pi(j) = \min_{i \in \rho_j^{-1}(\pi(i) + \ell[i,j])}$

End

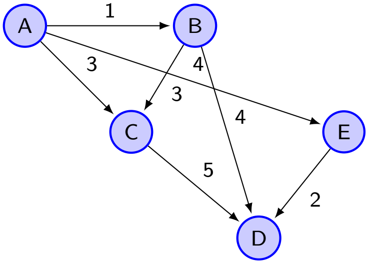
Property

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

Example

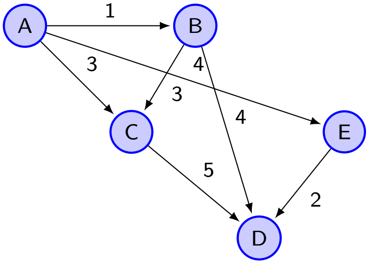


Example



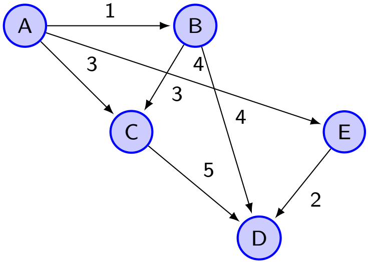
i	Γ_i^{-1}	rank(i)
A		
B		
C		
D		
E		

Example



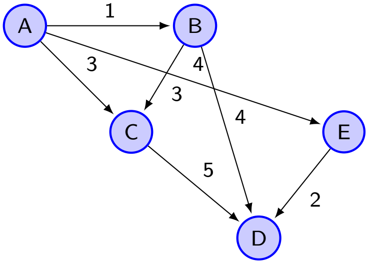
i	Γ_i^{-1}	rank(i)
A	-	
B	A	
C	A, B	
D	B, C , E	
E	A	

Example



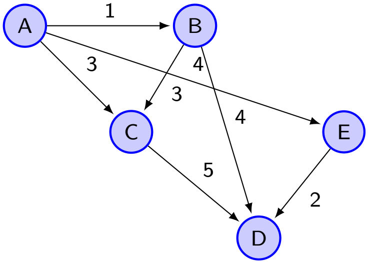
i	Γ_i^{-1}	rank(i)
A	-	0
B	A	
C	A, B	
D	B, C , E	
E	A	

Example



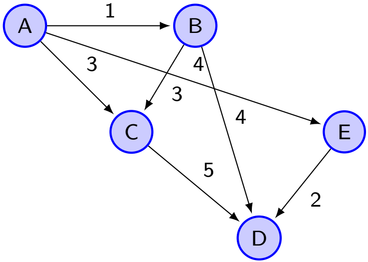
i	Γ_i^{-1}	rank(i)
A	-	0
B		1
C	B	
D	B, C , E	
E		1

Example



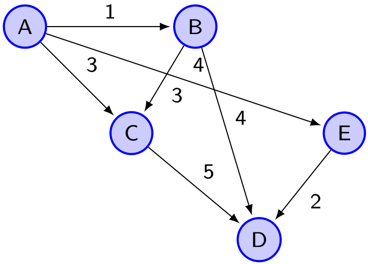
i	Γ_i^{-1}	rank(i)
A	-	0
B		1
C		2
D	C	
E		1

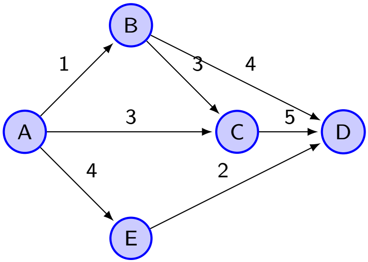
Example



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

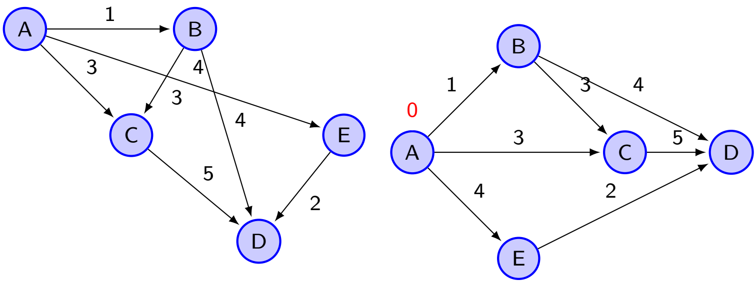
Example





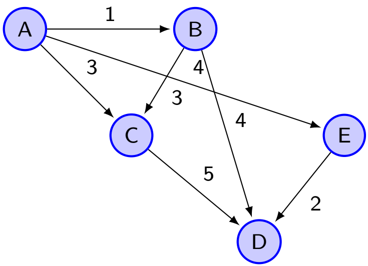
i	Γ_i^{-1}	rank(i)
A	-	0
B		1
C		2
D		3
E		1

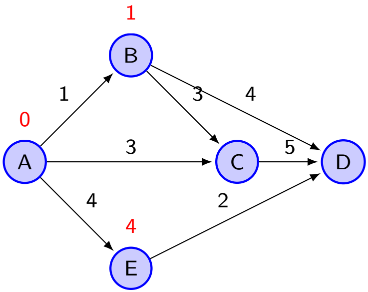
Example



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

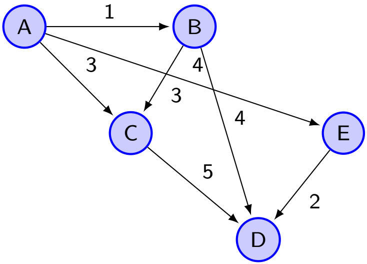
Example

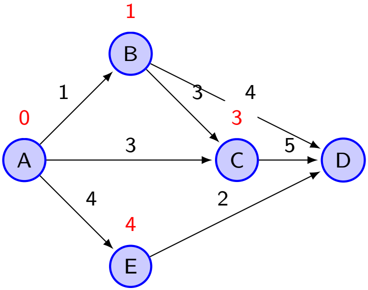




i	Γ_i^{-1}	rank(i)
A	-	0
B		1
C		2
D		3
E		1

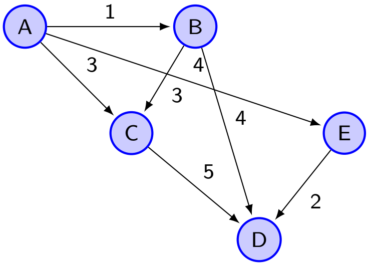
Example

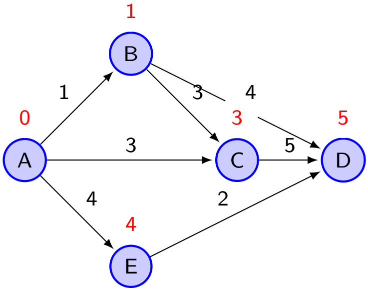




i	Γ_i^{-1}	rank(i)
A	-	0
B		1
C		2
D		3
E		1

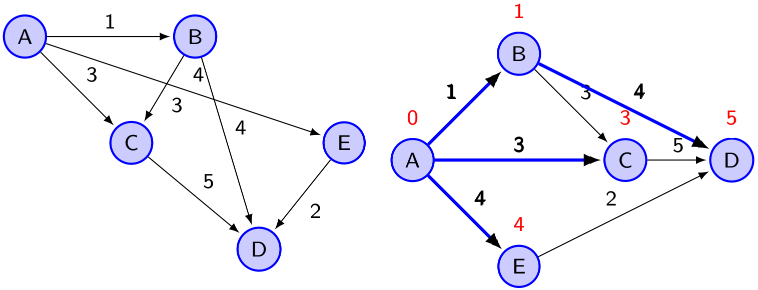
Example





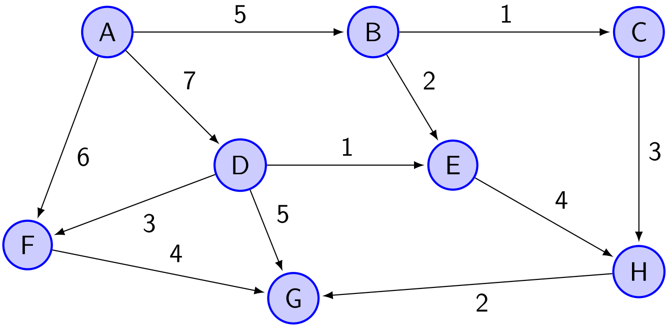
i	Γ_i^{-1}	rank(i)
A	-	0
B		1
C		2
D		3
E		1

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

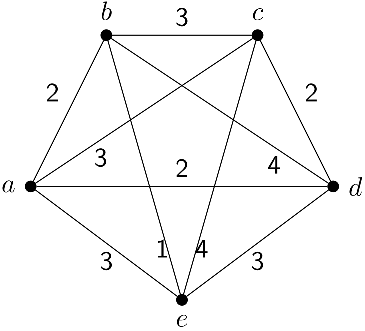
How to determine a shortest path from u to v in graph G which traverses at most $\leq$ a given constant number of intermediate vertices.

Traveling Salesman Problem

Problem

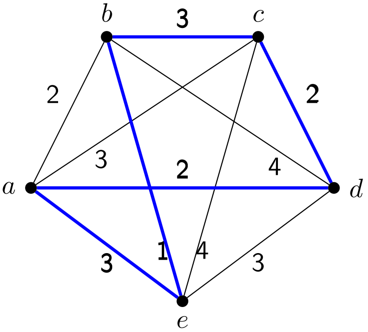
- Given a set of n customers located in n cities and distances for each pair of cities, the problem involves finding a round-trip with the minimum traveling cost.
- The vehicle must visit each customer exactly once and return to its point of origin also called depot.
- The objective function is the total 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



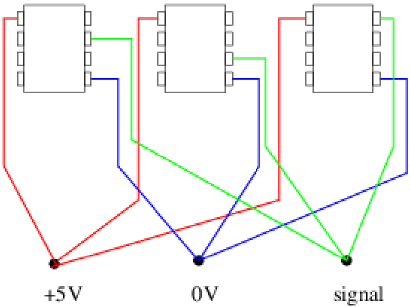
- The total number of possible Hamilton circuit is $(n-1)!/2$.
- For example, if there are 25 customers to visit, the total number of solutions is $24!/2 = 3.1 \times 10^{23}$.
- If the depot is located at node 1, then the optimal tour is $1-5-2-3-4-1$ with total cost equal to 11.

Vehicle Routing Problem

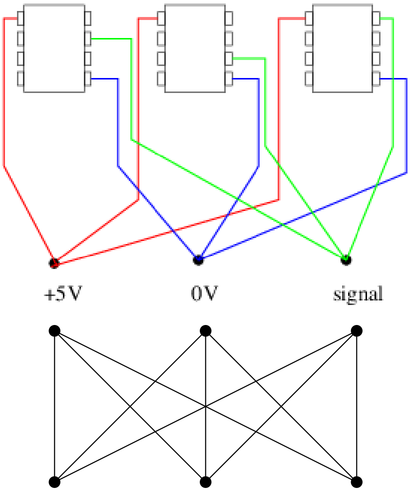
Problem

- 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.

Planar Graphs



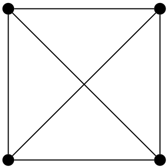
Planar Graphs



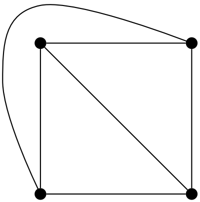
Planar Graphs

Definition

- A graph is called planar (*phẳng*) if it can be drawn in the plane without any edges crossing.
- Such a drawing is called planar representation (*biểu diễn phẳng*) of the graph.



K_4



K_4 with no crossing

Huynh Tuong Nguyen, Tran Vinh Tan

Important Corollaries

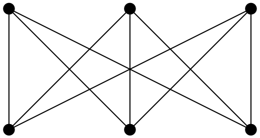
Corollary

- *If G is a* **connected planar simple graph** *with e edges and v vertices where $v \geq 3$, then $e \leq 3v - 6$.*

Important Corollaries

Corollary

- *If G is a* **connected planar simple graph** *with e edges and v vertices where $v \geq 3$, then $e \leq 3v - 6$.*
- *If G is a* **connected planar simple graph** *with e edges and v vertices where $v \geq 3$, and no circuits of length 3, then $e \leq 2v - 4$.*



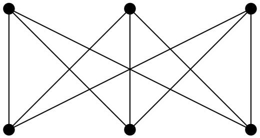
$K_{3,3}$

Huynh Tuong Nguyen,
Tran Vinh Tan

Important Corollaries

Corollary

- *If G is a* **connected planar simple graph** *with e edges and v vertices where $v \geq 3$, then $e \leq 3v - 6$.*
- *If G is a* **connected planar simple graph** *with e edges and v vertices where $v \geq 3$, and no circuits of length 3, then $e \leq 2v - 4$.*

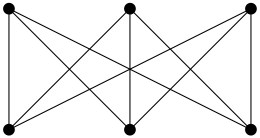


$K_{3,3}$
Non-planar

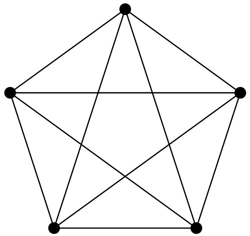
Important Corollaries

Corollary

- *If G is a* **connected planar simple graph** *with e edges and v vertices where $v \geq 3$, then $e \leq 3v - 6$.*
- *If G is a* **connected planar simple graph** *with e edges and v vertices where $v \geq 3$, and no circuits of length 3, then $e \leq 2v - 4$.*



$K_{3,3}$
Non-planar

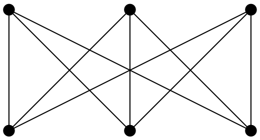


K_5

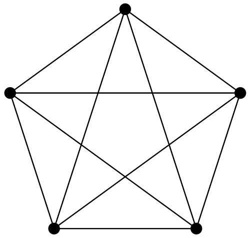
Important Corollaries

Corollary

- *If* G *is a* **connected planar simple graph** *with* e *edges and* v *vertices where* $v \geq 3$*, then* $e \leq 3v - 6$.
- *If* G *is a* **connected planar simple graph** *with* e *edges and* v *vertices where* $v \geq 3$*, and no circuits of length 3, then* $e \leq 2v - 4$.



$K_{3,3}$
Non-planar



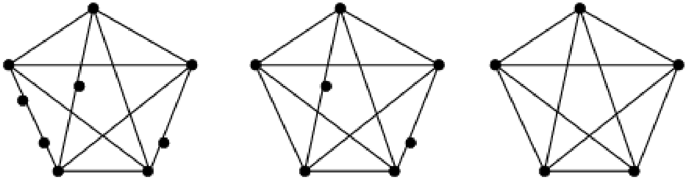
K_5
Non-planar

Contents

Elementary Subdivision

Definition

- Given a planar graph G, an elementary subdivision (*phân chia sơ cấp*) is removing an edge $\{u, v\}$ and adding a new vertex w together with edges $\{u, w\}$ and $\{w, v\}$.
- Graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are called homeomorphic (*đồng phôi*) if they can obtained from the same graph by a sequence of elementary subdivisions.



Kuratowski's Theorem

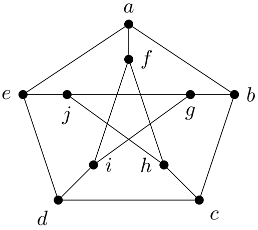
Theorem

A graph is nonplanar iff it contains a subgraph homeomorphic to $K_{3,3}$ or K_5.

Kuratowski's Theorem

Theorem

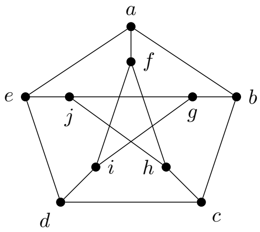
A graph is nonplanar iff it contains a subgraph homeomorphic to $K_{3,3}$ or K_5.

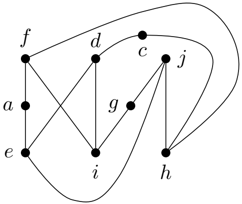


Kuratowski's Theorem

Theorem

A graph is nonplanar iff it contains a subgraph homeomorphic to $K_{3,3}$ *or* K_5.

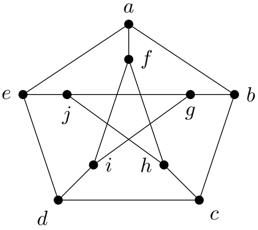


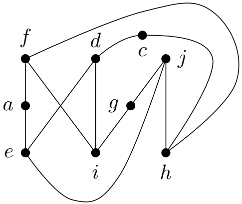


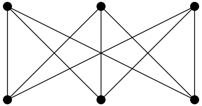
Kuratowski's Theorem

Theorem

A graph is nonplanar iff it contains a subgraph homeomorphic to $K_{3,3}$ or K_5.



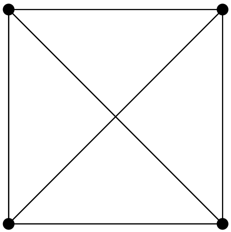




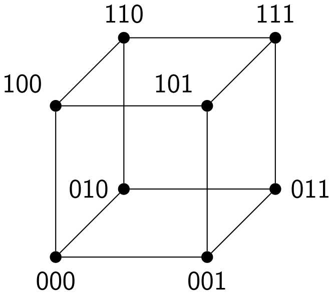
Exercise

Exercise

- Is K_4 planar?
- Is Q_3 planar?



K_4



Q_3

Maps and Graphs

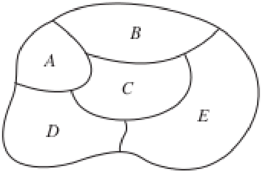
Definition

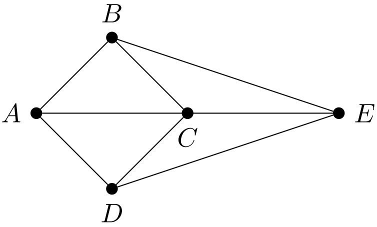
- Every map can be represented by a graph. We call it dual graph.
- Problem of coloring the regions of a map $\rightarrow$ coloring the vertices of the dual graph so that no two adjacent vertices have the same color.

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.

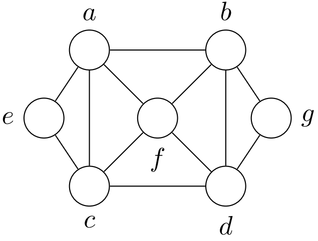




Graph coloring

Definition

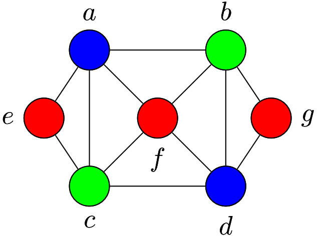
- A coloring (*tô màu*) of a simple graph is the assignment of a color to each vertex of the graph so that no two adjacent vertices are assigned the same color.



Graph coloring

Definition

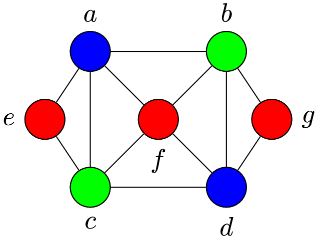
- A coloring (*tô màu*) of a simple graph is the assignment of a color to each vertex of the graph so that no two adjacent vertices are assigned the same color.
- The chromatic number (*số màu*) of a graph, denoted by $\chi(G)$, is the least number of colors needed for a coloring of this graph.

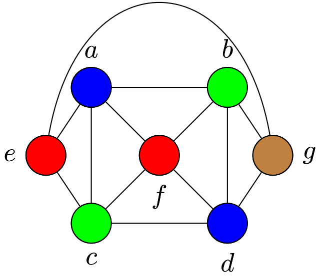


Graph coloring

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.





Four color theorem

Theorem (Four color theorem)

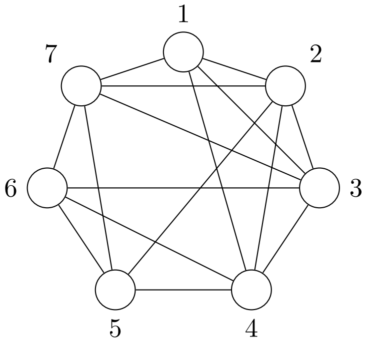
The chromatic number of a planar graph is no greater than four.

- 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

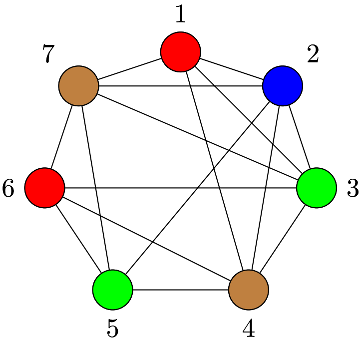
- How can the final exams at a university be scheduled so that no student has two exams at the same time?
- Suppose we have 7 finals, numbered 1 through 7.
- The pairs of courses have common students are depicted in the following graph



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?

Huynh Tuong Nguyen, Tran Vinh Tan

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?