Graph Theory and Optimization
Introduction on Graphs

Nicolas Nisse

Inria, France

Univ. Nice Sophia Antipolis, CNRS, I3S, UMR 7271, Sophia Antipolis, France

Master 1 international, Univ. Nice Sophia Antipolis, September, 2015

Lecture Notes:

http://www-sop.inria.fr/members/Frederic.Giroire/teaching/ubinet/

Outline

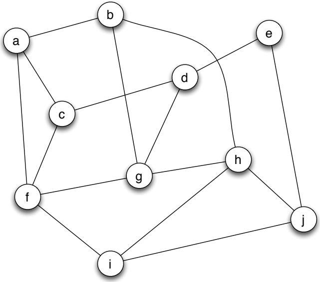
Graph: terminology and notations (Vertex/Edge)

A graph $G = (V,E)$

Vertices: $V = V(G)$ is a finite set *circles*

Edges: $E = V(E) \subseteq \{\{u,v\} \mid u,v \in V\}$ is a binary relation on V

lines between two circles



Example: $G = (V,E)$ with $V = \{a,b,c,d,e,f,g,h,i,j\}$ and
$E = \{\{a,b\},\{a,c\},\{a,f\},\{b,g\},\{b,h\},\{c,f\},\{c,d\},\{d,g\},\{d,e\},\{e,j\},\{f,g\},\{f,i\},\{g,h\},\{h,i\},\{h,j\},\{i,j\}\}$.

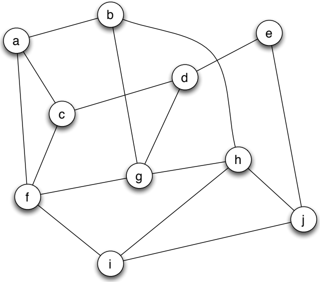
Graph: terminology and notations (Vertex/Edge)

A graph $G = (V, E)$

Vertices: $V = V(G)$ is a finite set *circles*

Edges: $E = V(E) \subseteq \{\{u, v\} \mid u, v \in V\}$ is a binary relation on V

lines between two circles



Example: $G = (V, E)$ with $V = \{a, b, c, d, e, f, g, h, i, j\}$ and

$E = \{\{a, b\}, \{a, c\}, \{a, f\}, \{b, g\}, \{b, h\}, \{c, f\}, \{c, d\}, \{d, g\}, \{d, e\}, \{e, j\}, \{f, g\}, \{f, i\}, \{g, h\}, \{h, i\}, \{h, j\}, \{i, j\}\}$.

Exercise: What is the maximum number of edges of a graph with n vertices?

Outline

1st Example: roads' network

What is the "best" road for reaching Oulu from Helsinki?

1st Example: roads' network

What is the “best" road for reaching Oulu from Helsinki?

Model geographical netowrk by a graph

1st Example: roads' network

What is the “best" road for reaching Oulu from Helsinki?

Model geographical netowrk by a graph

Use powerful tools that deal with graphs

1st Example: roads' network

More difficult setting

- traffic jam
- bus/subway schedule
- no-left, no-right and no U-turn signs at intersections.

Again, graph algorithm tools may help

That is how your GPS work !!

2nd Example: the Internet

Internet network (Autonomous Systems)

Optical networks (WDM)

- node= IP routers
- links= optical fiber
- capacity on links

- How to compute “best" routes?

- Where to put Amplificators?

- Which links to be turned off to limit energy consumption?

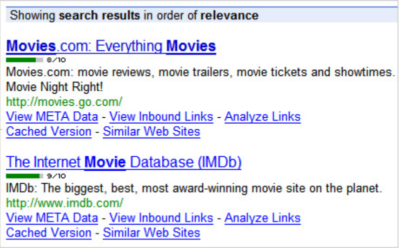
...

3rd Example: Social Network

Model of social interaction
a user = a node
two friends = an edge

- structure of social networks?
- communities?
- how to do advertisement?
- how to prevent advertisement?

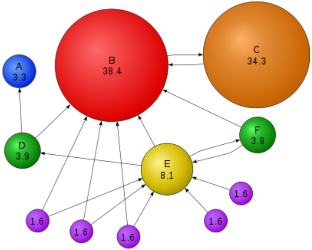
More Example: Web (google)



Google PageRank:
sort search results
node= web page
link = hyperlink

1. finding pages with the word movies in it
2. determining the importance of a page.

More Example: Web (google)



Google PageRank:

sort search results

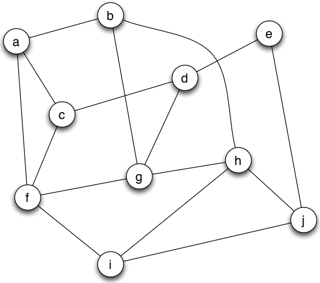
node= web page

link = hyperlink

1. finding pages with the word movies in it
2.
   - build the graph of the Web
   - do a random walk on a the graph or compute the eigenvector of a matrix

Outline

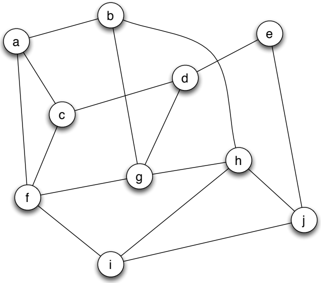
Graph: terminology and notations (Neighbor/Degree)



- two vertices $x \in V$ and $y \in V$ are adjacent or neighbors if $\{x,y\} \in E$
  i.e. there is an edge $\{x,y\}$
- $N(x)$: set of neighbors of $x \in V$ **ex:** $N(g) = \{b,d,f,h\} \subseteq V$
- degree of $x \in V$: number of neighbors of x i.e., $deg(x) = |N(x)|$

Exercise: Prove that, for any graph $G = (V,E)$, $\sum_{x \in V} deg(x) = 2|E|$

Graph: terminology and notations (Neighbor/Degree)

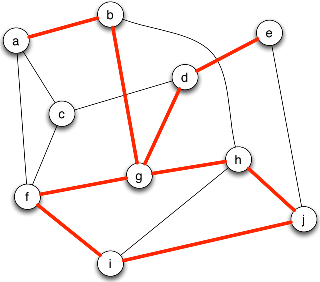


- two vertices $x \in V$ and $y \in V$ are adjacent or neighbors if $\{x, y\} \in E$
  i.e. there is an edge $\{x, y\}$
- $N(x)$: set of neighbors of $x \in V$ **ex:** $N(g) = \{b, d, f, h\} \subseteq V$
- degree of $x \in V$: number of neighbors of x i.e., $deg(x) = |N(x)|$

Exercise: Prove that, for any graph $G = (V, E)$, $\sum_{x \in V} deg(x) = 2|E|$

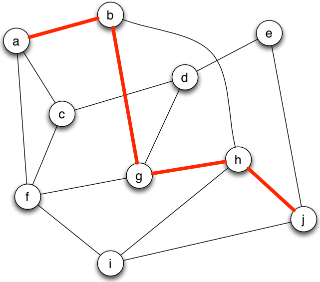
Outline

Terminology and notations (Path/Distance/Connectedness)



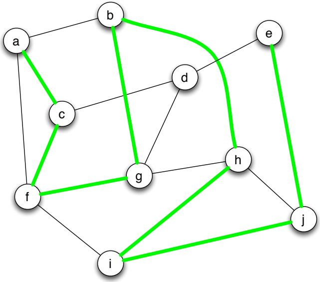
- **Walk**: sequence $(v_1, \cdots, v_\ell)$ of vertices such that consecutive vertices are adjacent, i.e., $\{v_i, v_{i+1}\} \in E$ for any $1 \leq i < \ell$
  ex: $W = (a, b, g, h, j, i, f, g, d, e)$

Terminology and notations (Path/Distance/Connectedness)



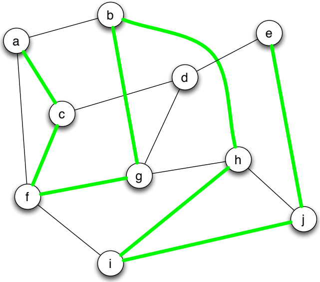
- Path: sequence $(v_1, \cdots, v_\ell)$ of distinct vertices such that consecutive vertices are adjacent, i.e., $\{v_i, v_{i+1}\} \in E$ for any $1 \leq i < \ell$
  ex: $P_1 = (a, b, g, h, j)$

Terminology and notations (Path/Distance/Connectedness)



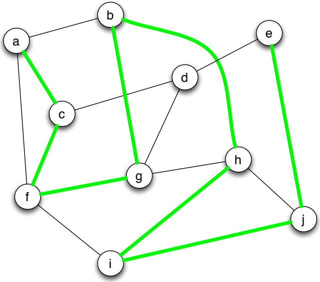
- Path: sequence $(v_1, \cdots, v_\ell)$ of distinct vertices such that consecutive vertices are adjacent, i.e., $\{v_i, v_{i+1}\} \in E$ for any $1 \leq i < \ell$
  ex: $P_1 = (a, b, g, h, j)$, $P_2 = (a, c, f, g, b, h, j, i, e)$

Terminology and notations (Path/Distance/Connectedness)



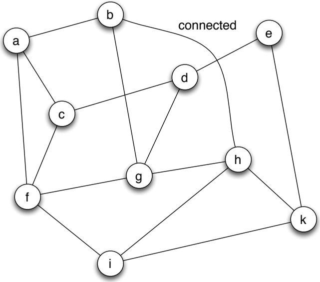
- Path: sequence $(v_1, \cdots, v_\ell)$ of distinct vertices such that consecutive vertices are adjacent, i.e., $\{v_i, v_{i+1}\} \in E$ for any $1 \leq i < \ell$
  ex: $P_1 = (a, b, g, h, j)$, $P_2 = (a, c, f, g, b, h, j, i, e)$
- Length of a path: number of its edges **ex:** $length(P_2) = 8$

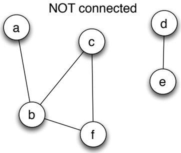
Terminology and notations (Path/Distance/Connectedness)



- Path: sequence $(v_1, \cdots, v_\ell)$ of distinct vertices such that consecutive vertices are adjacent, i.e., $\{v_i, v_{i+1}\} \in E$ for any $1 \leq i < \ell$
  ex: $P_1 = (a, b, g, h, j)$, $P_2 = (a, c, f, g, b, h, j, i, e)$
- Length of a path: number of its edges **ex:** $length(P_2) = 8$
- Distance between 2 vertices: length of a shortest path **ex:** $dist(a, j) = 3$

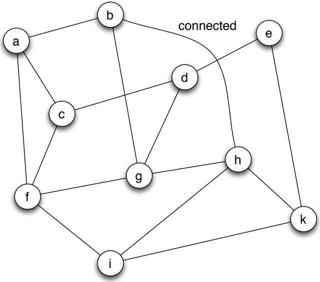
Terminology and notations (Path/Distance/Connectedness)

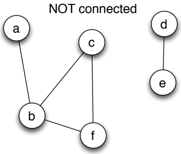




- Path: sequence $(v_1, \cdots, v_\ell)$ of distinct vertices such that consecutive vertices are adjacent, i.e., $\{v_i, v_{i+1}\} \in E$ for any $1 \leq i < \ell$
- Length of a path: number of its edges **ex:** $length(P_2) = 8$
- Distance between 2 vertices: length of a shortest path **ex:** $dist(a,j) = 3$
- $G = (V, E)$ is connected if, for any two vertices $x \in V$ and $y \in V$, there exists a path from x to y.

Terminology and notations (Path/Distance/Connectedness)



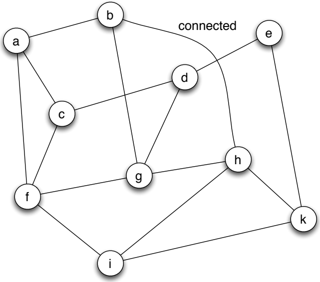


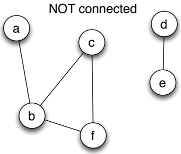
- Path: sequence $(v_1, \cdots, v_\ell)$ of distinct vertices such that consecutive vertices are adjacent, i.e., $\{v_i, v_{i+1}\} \in E$ for any $1 \leq i < \ell$
- Length of a path: number of its edges **ex:** $length(P_2) = 8$
- Distance between 2 vertices: length of a shortest path **ex:** $dist(a,j) = 3$
- $G = (V, E)$ is connected if, for any two vertices $x \in V$ and $y \in V$, there exists a path from x to y.

Exercise: Let G be a graph and $v \in V(G)$

Prove that G is connected iff, for any $y \in V$, there exists a path from x to y.

Terminology and notations (Path/Distance/Connectedness)



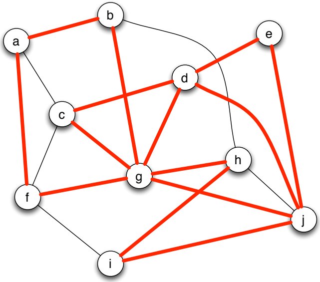


- Path: sequence $(v_1, \cdots, v_\ell)$ of distinct vertices such that consecutive vertices are adjacent, i.e., $\{v_i, v_{i+1}\} \in E$ for any $1 \leq i < \ell$
- Length of a path: number of its edges **ex:** $length(P_2) = 8$
- Distance between 2 vertices: length of a shortest path **ex:** $dist(a, j) = 3$
- $G = (V, E)$ is connected if, for any two vertices $x \in V$ and $y \in V$, there exists a path from x to y.

Exercise: Prove that if $|E| < |V| - 1$ then $G = (V, E)$ is NOT connected

Outline

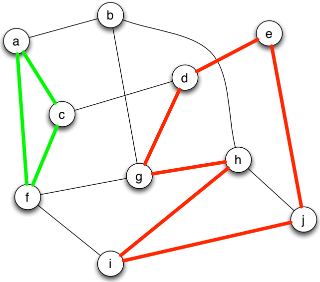
Graph: terminology and notations (Trail/Cycle)



- Trail: walk $(v_1, \cdots, v_\ell)$ such that $\ell \geq 3$ and $\{v_1, v_\ell\} \in E$

ex: $\mathit{Trail} = (a, b, g, d, c, g, h, i, j, d, e, j, g, f)$

Graph: terminology and notations (Trail/Cycle)



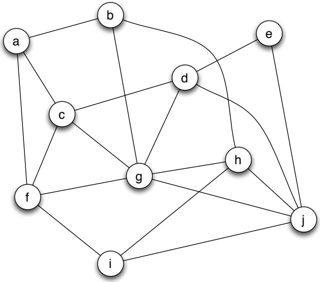
- Trail: walk $(v_1, \cdots, v_\ell)$ such that $\ell \geq 3$ and $\{v_1, v_\ell\} \in E$

  ex: *Trail* $= (a, b, g, d, c, g, h, i, j, d, e, j, g, f)$

- Cycle: path $(v_1, \cdots, v_\ell)$ such that $\ell \geq 3$ and $\{v_1, v_\ell\} \in E$

  ex: $C_1 = (d, e, j, i, h, g)$, $C_2 = (a, c, f)$

Graph: terminology and notations (Trail/Cycle)



- Trail: walk $(v_1, \cdots, v_\ell)$ such that $\ell \geq 3$ and $\{v_1, v_\ell\} \in E$

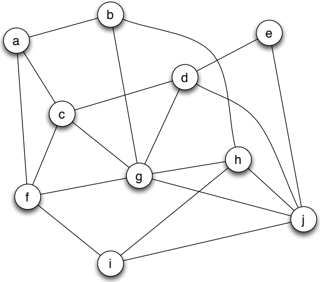
  ex: $Trail = (a,b,g,d,c,g,h,i,j,d,e,j,g,f)$

- Cycle: path $(v_1, \cdots, v_\ell)$ such that $\ell \geq 3$ and $\{v_1, v_\ell\} \in E$

  ex: $C_1 = (d,e,j,i,h,g)$, $C_2 = (a,c,f)$

- Eulerian trail: trail passing through all edges

Graph: terminology and notations (Trail/Cycle)



- Trail: walk $(v_1, \cdots, v_\ell)$ such that $\ell \geq 3$ and $\{v_1, v_\ell\} \in E$
  ex: $Trail = (a, b, g, d, c, g, h, i, j, d, e, j, g, f)$
- Cycle: path $(v_1, \cdots, v_\ell)$ such that $\ell \geq 3$ and $\{v_1, v_\ell\} \in E$
  ex: $C_1 = (d, e, j, i, h, g)$, $C_2 = (a, c, f)$
- Eulerian trail: trail passing through all edges

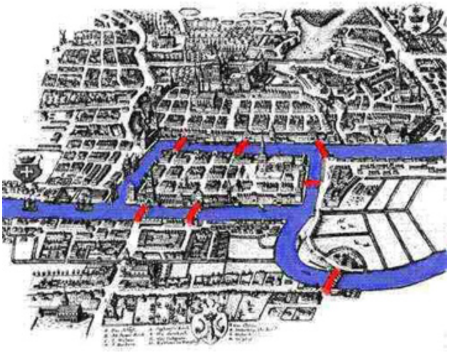
Exercise: Is this graph Eulerian, i.e., does it have an Eulerian trail?

Graph Theory an old story

Euler 1735: **Koenisberg bridges.**

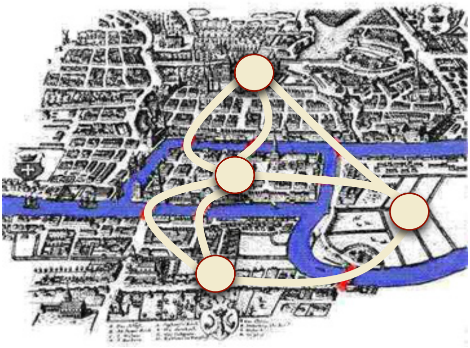
Existe-t-il un parcours empruntant tous les ponts une fois et une seule ?

Is there a trail going through each bridge exactly once?



Graph Theory an old story

Modeling: city = graph, island = vertex, bridge = edge

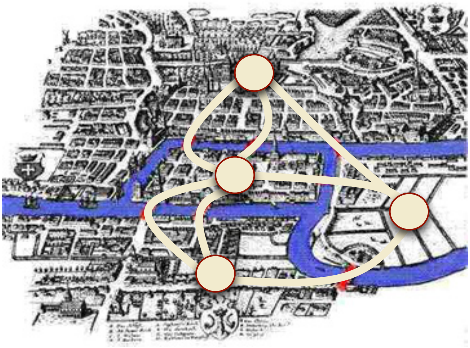


Graph Theory an old story

Modeling: city = graph, island = vertex, bridge = edge

Question: can we find an eulerian cycle in this graph?

Cycle going through all edges once and only once

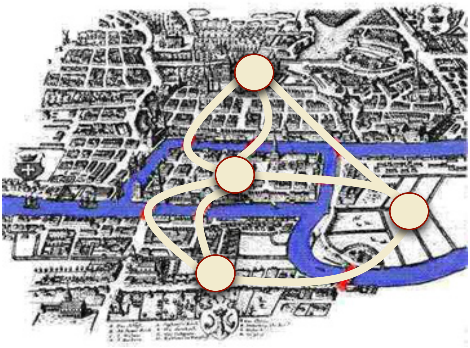


Graph Theory an old story

Modeling: city = graph, island = vertex, bridge = edge

Question: can we find an eulerian cycle in this graph?

Solution: Such cycle exists if and only if all nodes have even degree



Graph Theory an old story

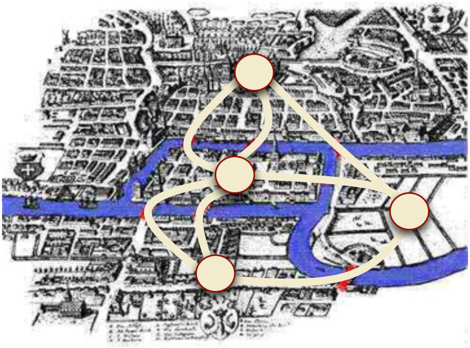
Modeling: city = graph, island = vertex, bridge = edge

Question: can we find an eulerian cycle in this graph?

Solution: Such cycle exists if and only if all nodes have even degree

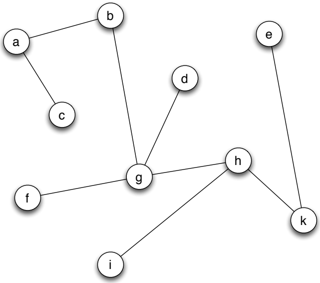
An intriguing variant: find a cycle going through all vertices once and only once (Hamiltonian cycle) is very difficult

One million dollar (Clay price) !



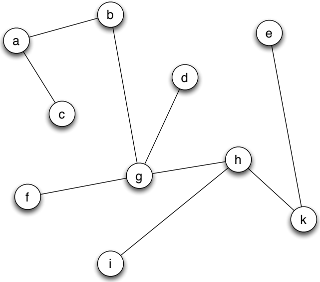
Outline

Graph: terminology and notations (Tree)



- Tree: connected graph without cycles
- Leaf: vertex of degree 1 in a tree

Graph: terminology and notations (Tree)



- Tree: connected graph without cycles
- Leaf: vertex of degree 1 in a tree

Trees are important because:

"simple" structure + "minimum" structure ensuring connectivity

Theorem:

$T = (V, E)$ is a tree $\Leftrightarrow$ T connected and $|V| = |E| + 1$

Graph: terminology and notations (Tree)

Theorem: $T = (V, E)$ is a tree $\Leftrightarrow$ T connected and $|V| = |E| + 1$

$\Leftarrow$ By contradiction:

- if T not a tree, then $\exists$ a cycle $(v_1, \cdots, v_\ell)$
- Let T' be obtained from T by removing edge $\{v_1, v_\ell\}$
- T' is connected *"technical" part, to be proved*
- $|E(T')| = |E| - 1 = |V| - 2 = |V(T')| - 2$
- so $|E'| < |V'| - 1$ and T' is not connected by previous Exercise

A contradiction

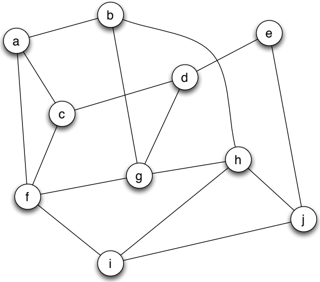
Graph: terminology and notations (Tree)

Theorem: $T=(V,E)$ is a tree $\Leftrightarrow$ T connected and $|V|=|E|+1$

$\Rightarrow$ Induction on $|V|$ OK if $|V|=1$

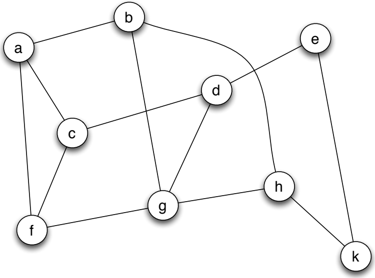
- Let $P=(v_1,\cdots,v_\ell)$ be a longest path in T (ℓ max., in particular $\ell \geq 2$)
- v_1 is a leaf. By contradiction:
  - assume $deg(v_1)>1$, and $x \in N(v_1) \setminus \{v_2\}$
  - $x \notin V(P)$ otherwise there is a cycle in T
  - then, $(x,v_1,\cdots,v_\ell)$ path longer than P, a contradiction
- then $S=T \setminus \{v_1\}$ is a tree *"technical" part, to be proved*
- $|V(S)|<|V|$ so, by induction $|V(S)|=|E(S)|+1$
- $|V|=|V(S)|+1$ and $|E|=|E(S)|+1$, so $|V|=|E|-1$

Graph: terminology and notations (subgraph)



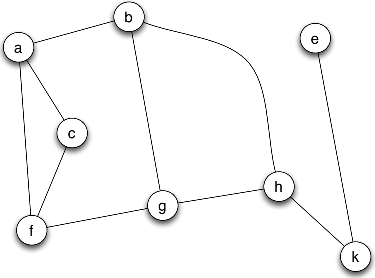
- Subgraph of $G = (V,E)$: any graph $H = (V',E')$ with
$$V' \subseteq V \text{ and } E' \subseteq \{\{x,y\} \in E \mid x,y \in V'\}$$
obtained from G by removing some vertices and edges

Graph: terminology and notations (subgraph)



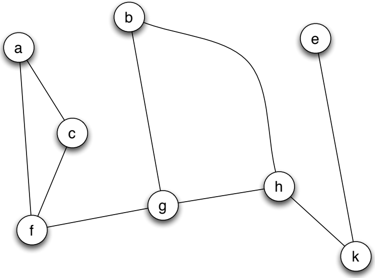
- Subgraph of $G=(V,E)$: any graph $H=(V',E')$ with
$$V' \subseteq V \text{ and } E' \subseteq \{\{x,y\} \in E \mid x,y \in V'\}$$
obtained from G by removing some vertices and edges

Graph: terminology and notations (subgraph)



- Subgraph of $G=(V,E)$: any graph $H=(V',E')$ with
  $V' \subseteq V$ and $E' \subseteq \{\{x,y\} \in E \mid x,y \in V'\}$
  obtained from G by removing some vertices and edges

Graph: terminology and notations (subgraph)

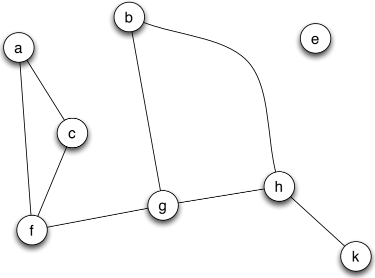


- Subgraph of $G = (V, E)$: any graph $H = (V', E')$ with

$$V' \subseteq V \text{ and } E' \subseteq \{\{x,y\} \in E \mid x,y \in V'\}$$

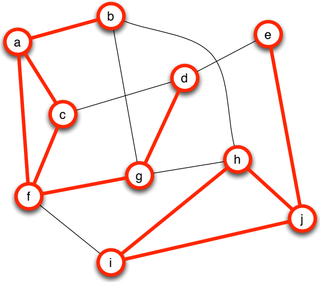
obtained from G by removing some vertices and edges

Graph: terminology and notations (subgraph)



- Subgraph of $G = (V, E)$: any graph $H = (V', E')$ with
  $$V' \subseteq V \text{ and } E' \subseteq \{\{x,y\} \in E \mid x,y \in V'\}$$
  obtained from G by removing some vertices and edges

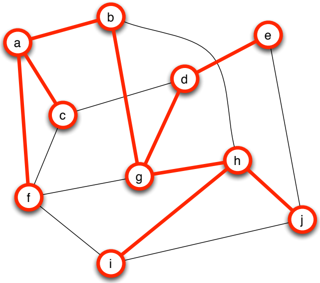
Graph: terminology and notations (subgraph)



- Subgraph: $H = (V', E')$ with $V' \subseteq V$ and $E' \subseteq \{\{x,y\} \in E \mid x,y \in V'\}$
- Spanning subgraph of G: subgraph $H = (V', E')$ where $V' = V$

obtained from G by removing only some edges

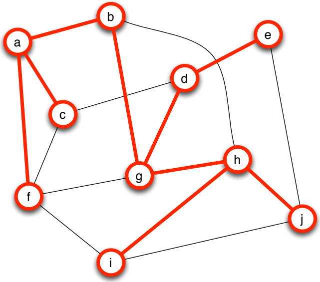
Graph: terminology and notations (subgraph)



- Subgraph: $H = (V', E')$ with $V' \subseteq V$ and $E' \subseteq \{\{x,y\} \in E \mid x, y \in V'\}$
- Spanning subgraph of G: subgraph $H = (V', E')$ where $V' = V$

  obtained from G by removing only some edges
- Spanning tree of G: spanning subgraph $H = (V, E')$ with H a tree

Graph: terminology and notations (subgraph)



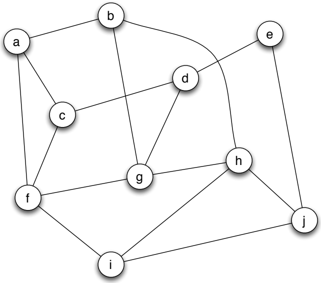
- Subgraph: $H = (V', E')$ with $V' \subseteq V$ and $E' \subseteq \{\{x,y\} \in E \mid x, y \in V'\}$
- Spanning subgraph of G: subgraph $H = (V', E')$ where $V' = V$

  obtained from G by removing only some edges
- Spanning tree of G: spanning subgraph $H = (V, E')$ with H a tree

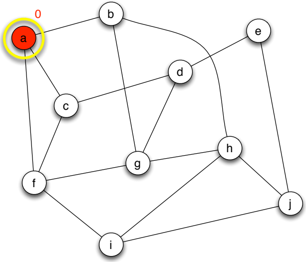
Exercise: A graph G is connected if and only if G has a spanning tree

Outline

Searching algorithms (BFS/DFS)

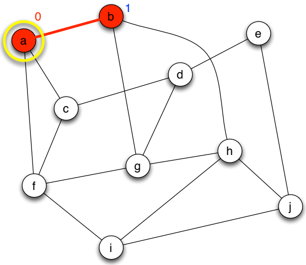


Searching algorithms (BFS/DFS)



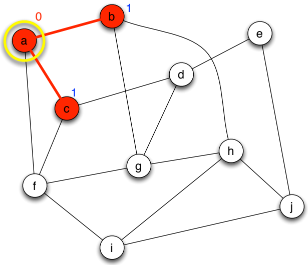
ToBeExplored=(a)

Searching algorithms (BFS/DFS)



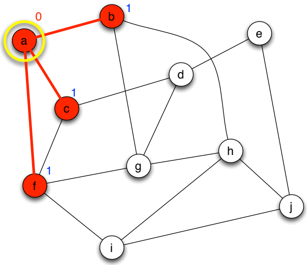
ToBeExplored=(a,b)

Searching algorithms (BFS/DFS)



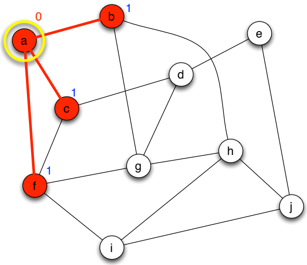
ToBeExplored=(a,b,c)

Searching algorithms (BFS/DFS)



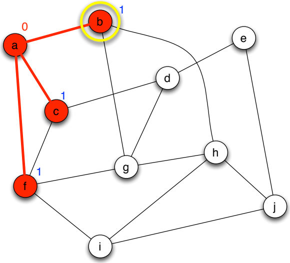
ToBeExplored=(a,b,c,f)

Searching algorithms (BFS/DFS)



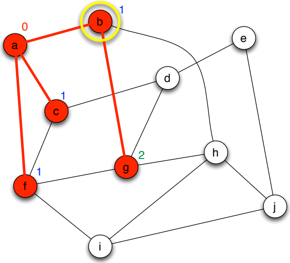
ToBeExplored=(b,c,f)

Searching algorithms (BFS/DFS)



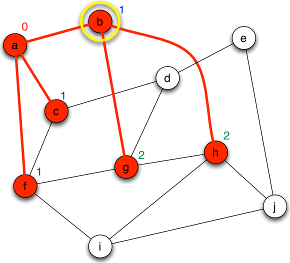
ToBeExplored=(b,c,f)

Searching algorithms (BFS/DFS)



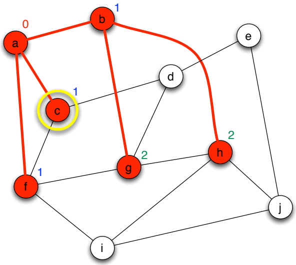
ToBeExplored=(b,c,f,g)

Searching algorithms (BFS/DFS)



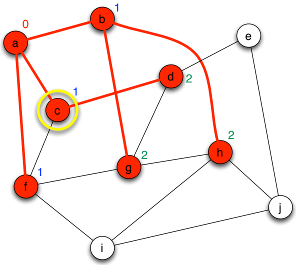
ToBeExplored=(b,c,f,g,h)

Searching algorithms (BFS/DFS)



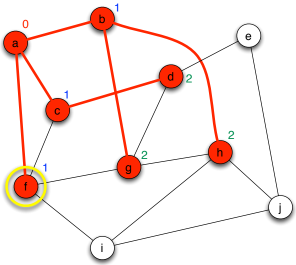
ToBeExplored=(c,f,g,h)

Searching algorithms (BFS/DFS)



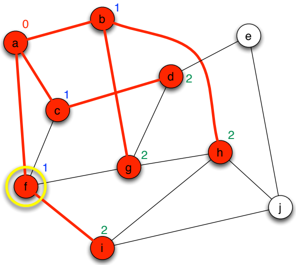
ToBeExplored=(c,f,g,h,d)

Searching algorithms (BFS/DFS)



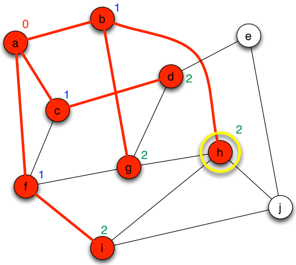
ToBeExplored=(f,g,h,d)

Searching algorithms (BFS/DFS)



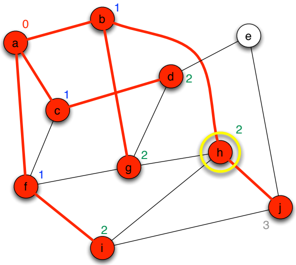
ToBeExplored=(f,g,h,d,i)

Searching algorithms (BFS/DFS)



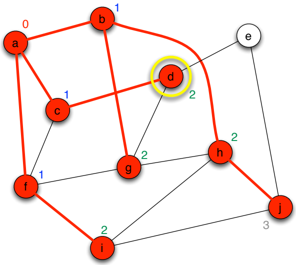
ToBeExplored=(h,d,i)

Searching algorithms (BFS/DFS)



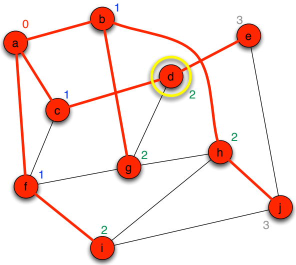
ToBeExplored=(h,d,i,j)

Searching algorithms (BFS/DFS)



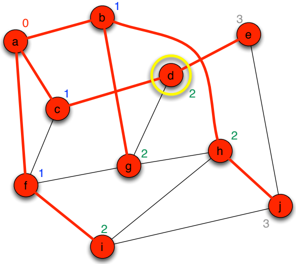
ToBeExplored=(d,i,j)

Searching algorithms (BFS/DFS)



ToBeExplored=(d,i,j,e)

Searching algorithms (BFS/DFS)



ToBeExplored=(d,i,j,e)

Breadth First Search

input: unweighted graph $G = (V, E)$ and $r \in V$

Initially: $d(r) = 0$, $ToBeExplored = (r)$
and $T = (V(T), E(T)) = (\{r\}, \emptyset)$

While $ToBeExplored \neq \emptyset$ **do**

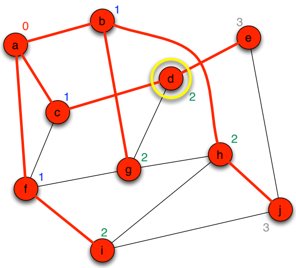
Let $v = head(ToBeExplored)$

for $u \in N(v) \setminus ToBeExplored$ **do**

$d(u) \leftarrow d(v) + 1$
add u in $V(T)$ and $\{v, u\}$ in $E(T)$
add u at the end of $ToBeExplored$

remove v from $ToBeExplored$

Searching algorithms (BFS/DFS)



ToBeExplored=(d,i,j,e)

Breadth First Search

input: unweighted graph $G = (V, E)$ and $r \in V$

Initially: $d(r) = 0$, $ToBeExplored = (r)$ and $T = (V(T), E(T)) = (\{r\}, \emptyset)$

While $ToBeExplored \neq \emptyset$ **do**

Let $v = head(ToBeExplored)$

for $u \in N(v) \setminus ToBeExplored$ **do**

$d(u) \leftarrow d(v) + 1$

add u in $V(T)$ and $\{v, u\}$ in $E(T)$

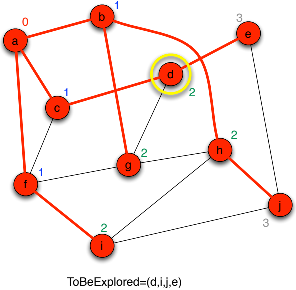
add u at the end of $ToBeExplored$

remove v from $ToBeExplored$

Output: for any $v \in V$, $d(v) = dist(r, v)$.

T is a shortest path tree of G rooted in r: i.e., T spanning subtree of G s.t. for any $v \in V$, the path from r to v in T is a shortest path from r to v in G.

Searching algorithms (BFS/DFS)



Breadth First Search

input: unweighted graph $G = (V, E)$ and $r \in V$

Initially: $d(r) = 0$, $ToBeExplored = (r)$
and $T = (V(T), E(T)) = (\{r\}, \emptyset)$

While $ToBeExplored \neq \emptyset$ **do**

Let $v = head(ToBeExplored)$

for $u \in N(v) \setminus ToBeExplored$ **do**

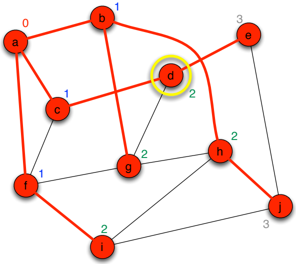
$d(u) \leftarrow d(v) + 1$
add u in $V(T)$ and $\{v, u\}$ in $E(T)$
add u at the end of $ToBeExplored$

remove v from $ToBeExplored$

Time-Complexity: $\#$ operations $= O(|E|)$ *each edge is considered*

Exercise: Give an algorithm that decides if a graph is connected

Searching algorithms (BFS/DFS)



ToBeExplored=(d,i,j,e)

Breadth First Search

input: unweighted graph $G = (V, E)$ and $r \in V$

Initially: $d(r) = 0$, $ToBeExplored = (r)$
and $T = (V(T), E(T)) = (\{r\}, \emptyset)$

While $ToBeExplored \neq \emptyset$ **do**

Let $v = head(ToBeExplored)$

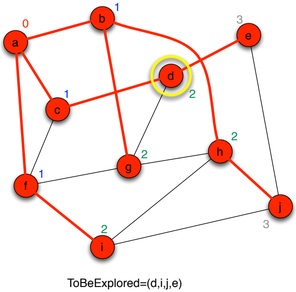
for $u \in N(v) \setminus ToBeExplored$ **do**

$d(u) \leftarrow d(v) + 1$
add u in $V(T)$ and $\{v, u\}$ in $E(T)$
add u at the end of *ToBeExplored*

remove v from *ToBeExplored*

Time-Complexity: $\#$ operations $= O(|E|)$ *each edge is considered*
Rmk1: allows to decide whether G is connected
G connected iff $dist(r, v) < \infty$ defined for all $v \in V$

Searching algorithms (BFS/DFS)



Breadth First Search

input: unweighted graph $G = (V, E)$ and $r \in V$

Initially: $d(r) = 0$, $ToBeExplored = (r)$
and $T = (V(T), E(T)) = (\{r\}, \emptyset)$

While $ToBeExplored \neq \emptyset$ **do**

Let $v = head(ToBeExplored)$

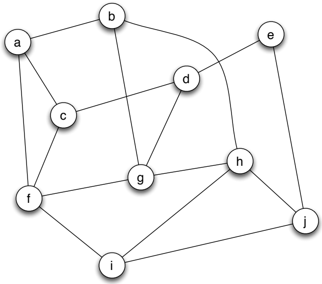
for $u \in N(v) \setminus ToBeExplored$ **do**

$d(u) \leftarrow d(v) + 1$
add u in $V(T)$ and $\{v, u\}$ in $E(T)$
add u at the end of $ToBeExplored$

remove v from $ToBeExplored$

Time-Complexity: $\#$ operations $= O(|E|)$ *each edge is considered*
Rmk2: gives only one shortest path tree, may be more...
depends on the ordering in which vertices are considered

Searching algorithms (BFS/DFS)



Depth First Search

input: unweighted graph $G = (V, E)$ and $r \in V$

Initially: $d(r) = 0$, $ToBeExplored = (r)$ and $T = (V(T), E(T)) = (\{r\}, \emptyset)$

While $ToBeExplored \neq \emptyset$ **do**

Let $v = head(ToBeExplored)$

for $u \in N(v) \setminus ToBeExplored$ **do**

$d(u) \leftarrow d(v) + 1$
add u in $V(T)$ and $\{v, u\}$ in $E(T)$
add u at the beginning of *ToBeExplored*

remove v from *ToBeExplored*

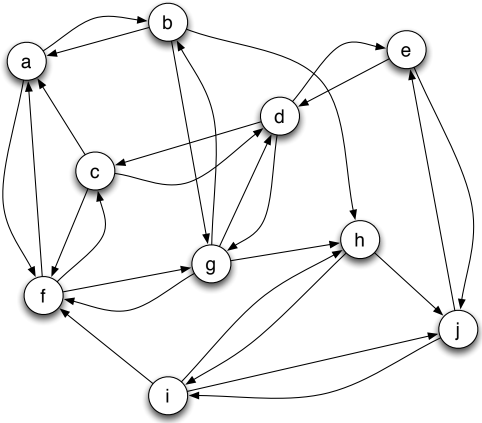
Exercise: Gives a DFS-tree rooted in a in this graph

Outline

Directed graphs

Directed graph: $D = (V, A)$, A : set of arcs, $(x, y) \in A$ ordered pair

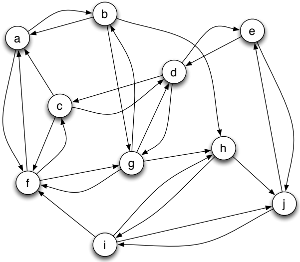
arrows



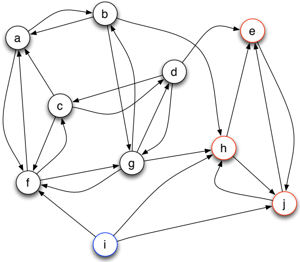
Directed graphs

Directed graph: $D = (V, A)$, A : set of arcs, $(x, y) \in A$ ordered pair

arrows



strongly connected



not strongly connected

$D = (V, A)$ is strongly connected if, for any two vertices $x \in V$ and $y \in V$, there exists a directed path from x to y AND a directed path from y to x.

Directed graphs strongly connected

Exercise 1: Let D be a digraph and $v \in V(D)$
Prove that D is strongly connected iff, for any $y \in V$, there exist a directed path from x to y AND a directed path from y to x.

Exercise 2:
Give an efficient algorithm that decides if a digraph is strongly connected

Summary: To be remembered

All definitions will be important in next lectures
Please remember:

- graph, vertex/vertices, edge
- neighbor, degree
- path, distance, cycle (Eulerian,Hamiltonian)
- connected graph
- tree
- subgraph, spanning subgraph
- BFS/DFS
- Directed graph