Introduction to Algorithms

Class 3: Graph Algorithm 2

Jianxi Gao

Department of Computer Science
Rensselaer Polytechnic Institute

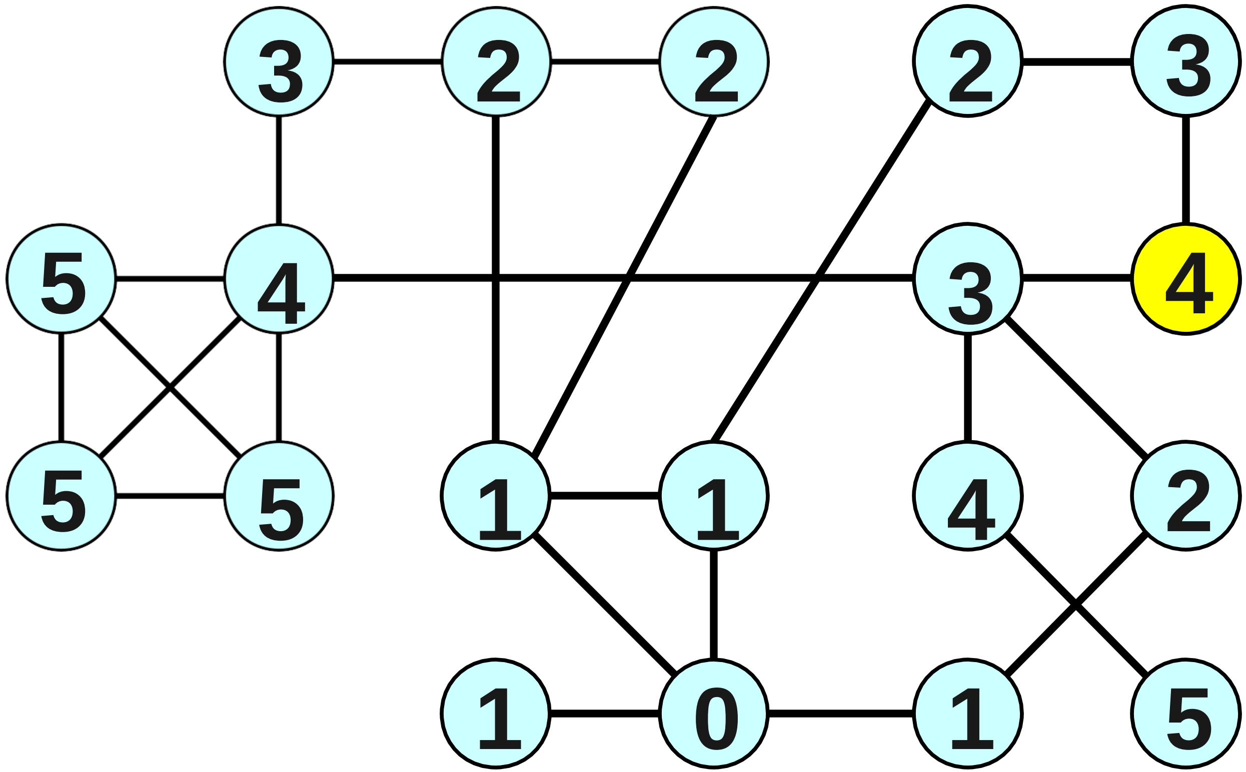
www.gaojianxi.com

Wednesday
Jan 8 Office Hours (Lab) *Must attend*
Jan 15 Office Hours (Lab) *Must attend*
Jan 22 Office Hours (Lab) *Must attend*
Jan 29 Office Hours (Lab) *Must attend*
Feb 5 Office Hours (Lab) *Must attend*
Feb 12 Attendance *optional* (Lab) ***Exam 1 on Wednesday,, 8am-9:50am***
Feb 19 Office Hours (Lab) *Must attend*
Feb 26 Office Hours (Lab) *Must attend*
March 5: Spring break
March 12 Office Hours (Lab) *Must attend*
March 19 Office Hours (Lab) *Must attend*
March 26 Office Hours (Lab) *Must attend*
April 2 Attendance *optional* (Lab) ***Exam 2 on Wednesday, 8am-9:50am***
April 9 Office Hours (Lab) *Must attend*
April 16 Office Hours (Lab) *Must attend*
April 23 Office Hours (Lab) *Attendance optional*

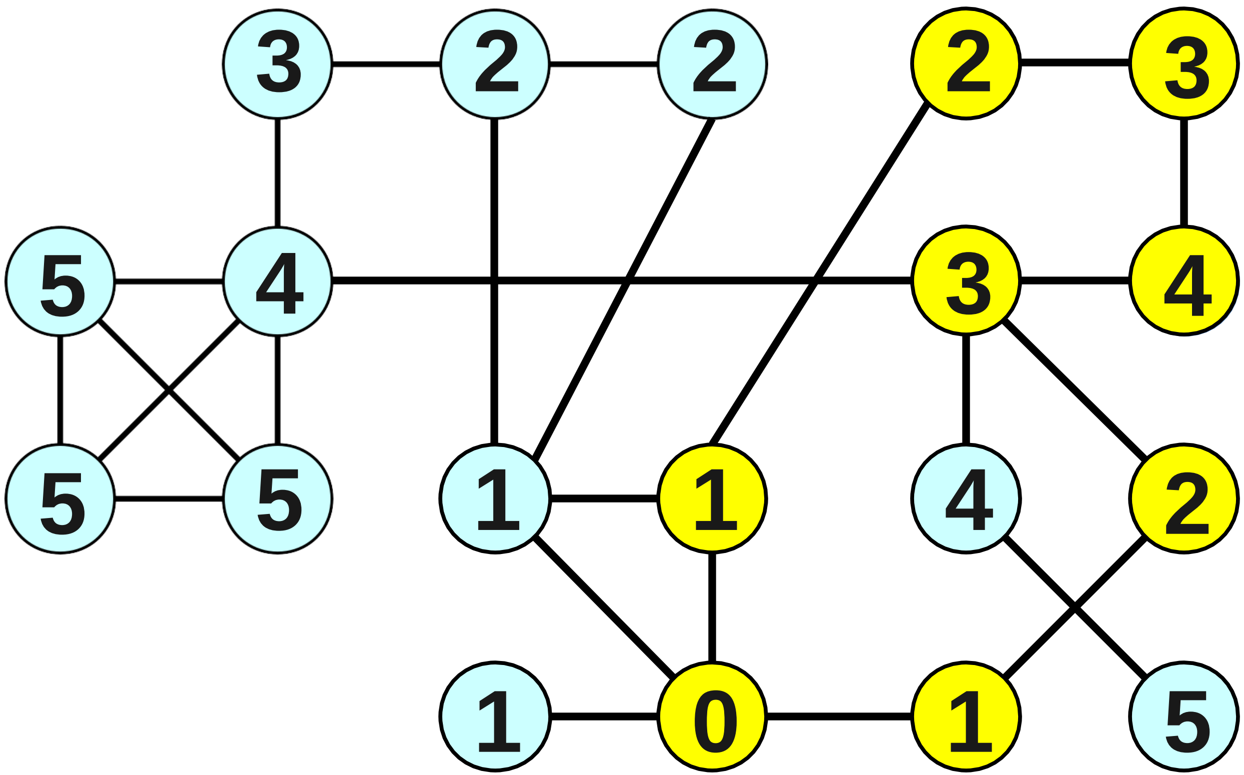
About the Lab

Lab Section 1: 10:00 - 12:00 PITTS 4206 Mentors: Steven, Joe, Yuqing	Lab Section 2: 10:00 - 12:00 PITTS 5114 Mentors: Ryan, Eleanor, Heman
Lab Section 3: 12:00 - 2:00 PITTS 5114 Mentors: Ryan, Eleanor, Heman	Lab Section 4: 12:00 - 2:00 PITTS 4206 Mentors: Steven, Joe, Yuqing
Lab Section 5: 2:00 - 4:00 PITTS 5114 Mentors: Dakota, Charlotte, Christian	Lab Section 6: 2:00 - 4:00 PITTS 4206 Mentors: Kerui, Mrunal, Max
Lab Section 7: 4:00 - 6:00 PITTS 5114 Mentors: Dakota, Charlotte, Christian	Lab Section 8: 4:00 - 6:00 PITTS 4206 Mentors: Kerui, Mrunal, Max

Number of shortest paths



Number of shortest paths

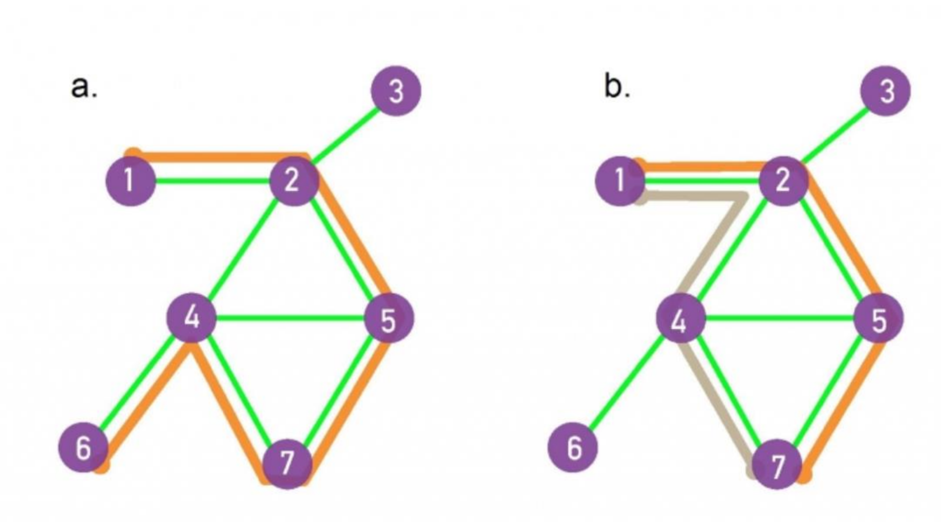


PATHS

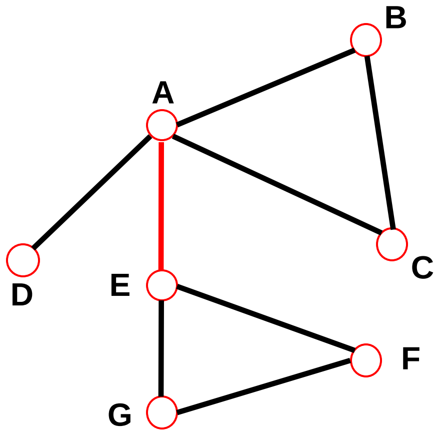
A *path is* a sequence of nodes in which each node is adjacent to the next one.

$P_{i0,in}$ of length *n* between nodes i_0 and i_n is an ordered collection of *n+1* nodes and *n* links

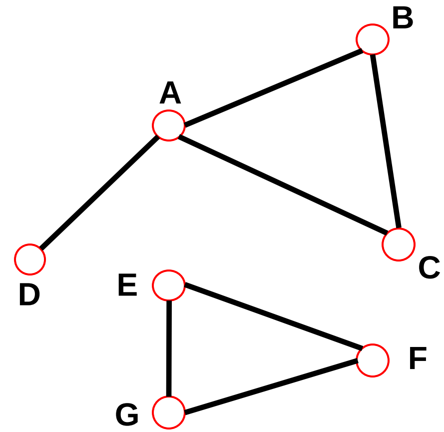
$$P_n = \{(i_0, i_1), (i_1, i_2), (i_2, i_3), \ldots, (i_{n-1}, i_n)\}$$



Connected graph



Connected



Not connected

Telephone
Internet
transportation

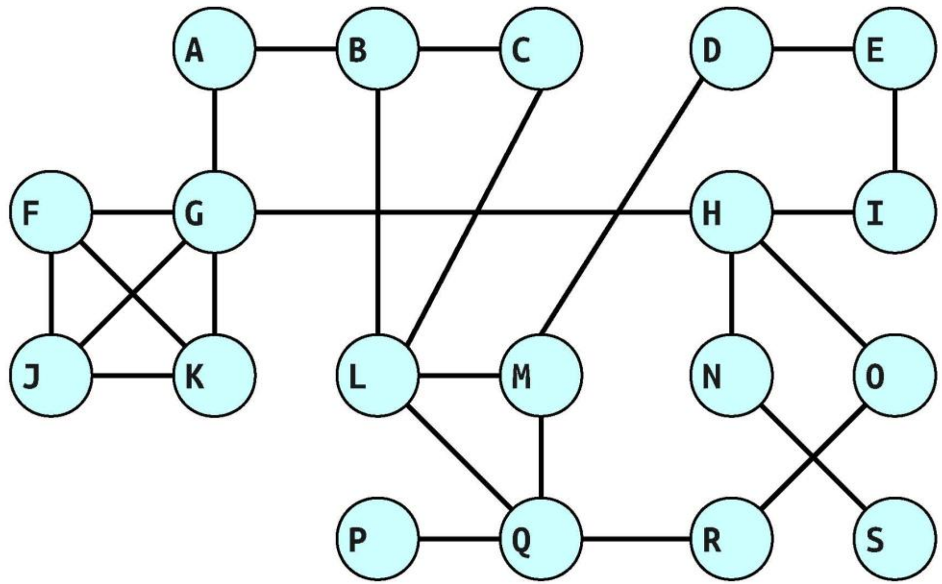
Bridge: A-F

Def 1: Nodes *u* and *v* are connected, if ∃ a path connection *u* and *v*.

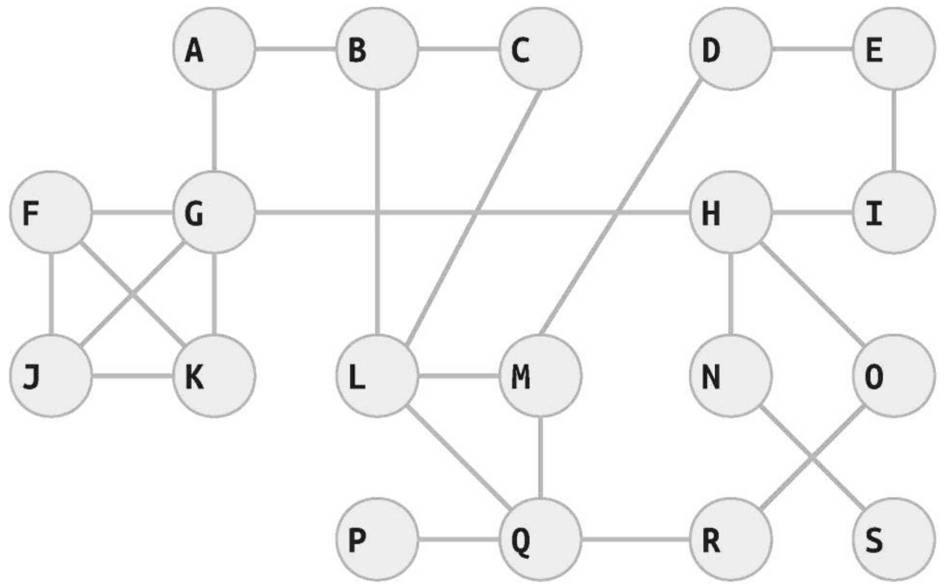
Def 2: A graph is connected if all nodes are connected to each other.

A disconnected graph is made up by two or more connected components.

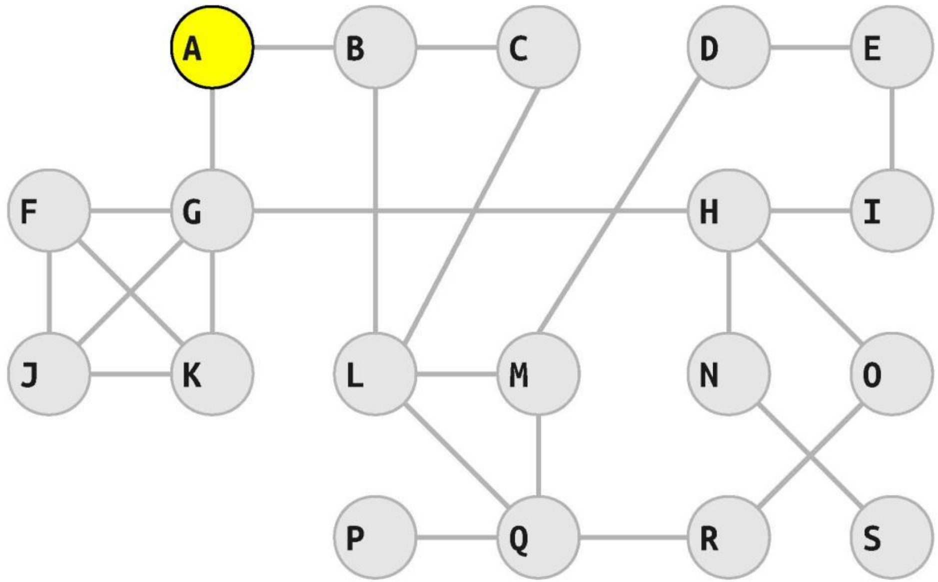
Breadth-First Search



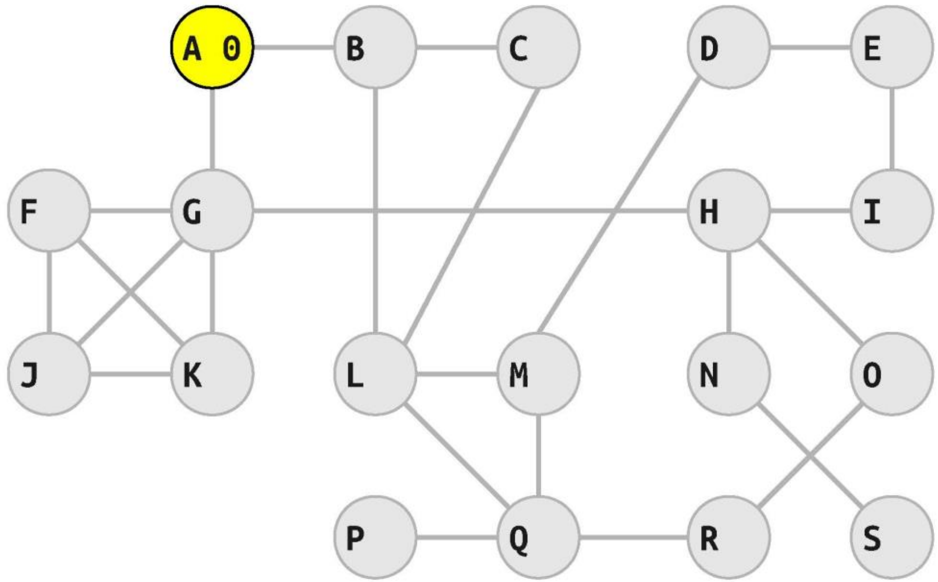
Breadth-First Search



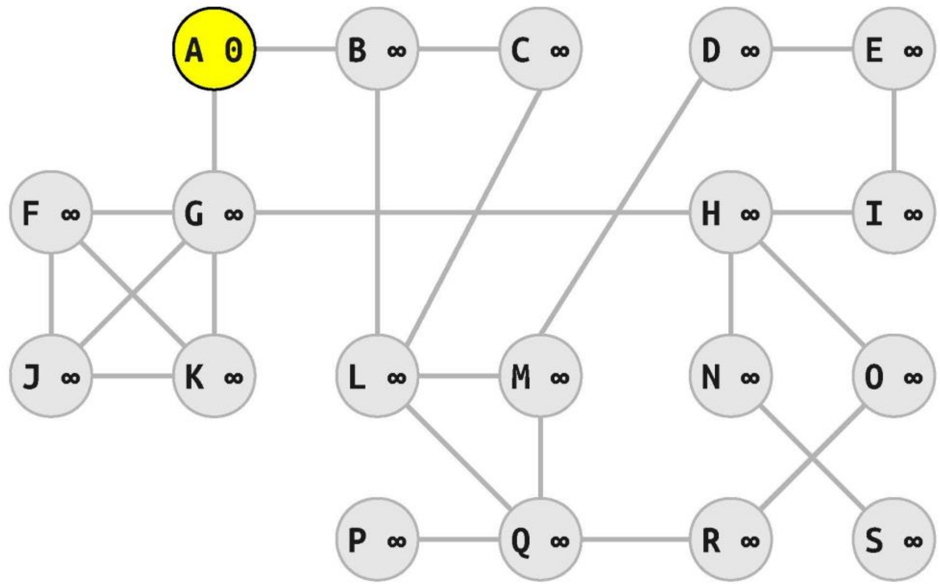
Breadth-First Search



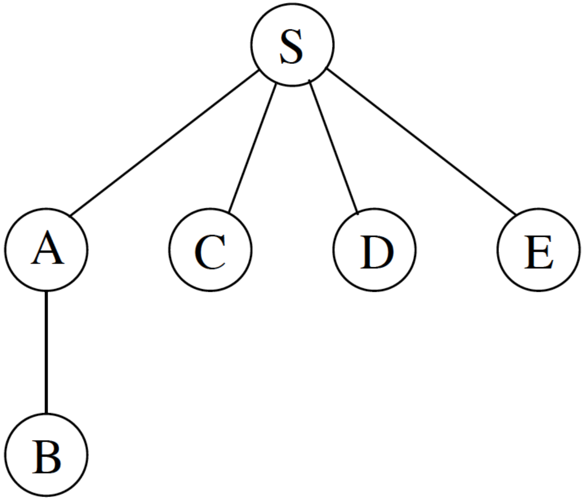
Breadth-First Search



Breadth-First Search



Example from the Book



Order of visitation	Queue contents after processing node
	$[S]$
S	$[A\ C\ D\ E]$
A	$[C\ D\ E\ B]$
C	$[D\ E\ B]$
D	$[E\ B]$
E	$[B]$
B	$[\ ]$

Breadth-first Search

Procedure BFS(G, s)

Input: Graph $G = (V, E)$, directed or undirected; vertex $s \in V$.

Output: For all vertices u reachable from s, $dist(t)$ is set to the distance from s to u.

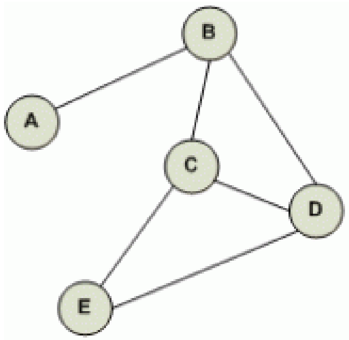


Fig 1. Undirected Graph

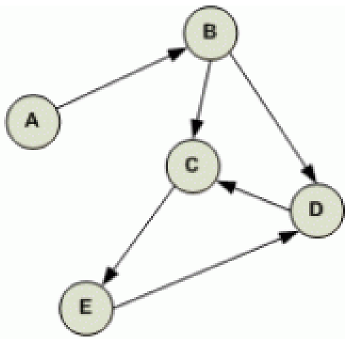


Fig 2. Directed Graph

Breadth-first Search

for all u $\in V$;

$dist(u) = \infty$

$dist(s) = 0$

$Q = [s]$ (queue only contains s)

while Q is not empty

$u = eject(Q)$ (Remove the first element u from Q)

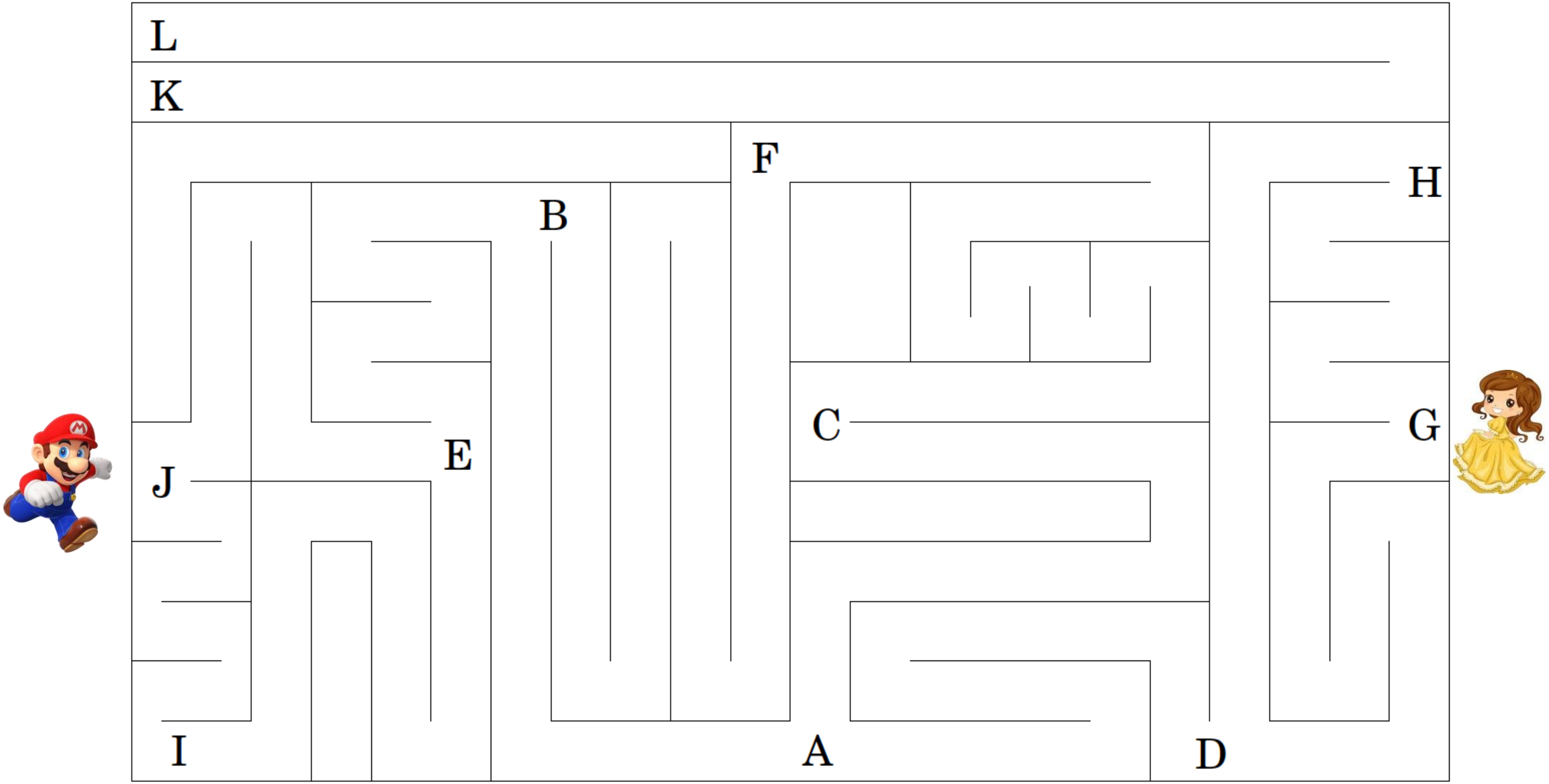
for all edges $(u, v) \in E$:

if $dist(v) = \infty$:

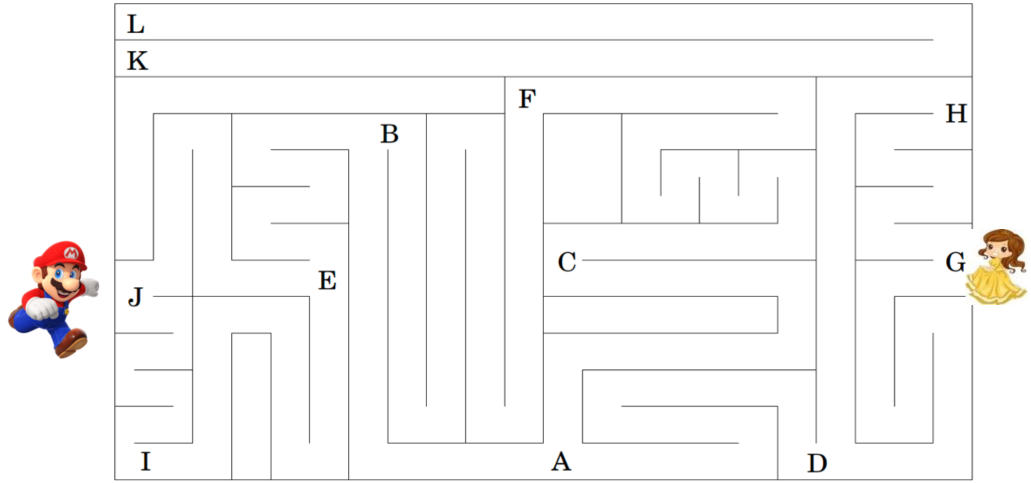
$inject(Q, v)$

$dist(v) = dist(u) + 1$

Meeting princess



Meeting princes



Here are the questions:

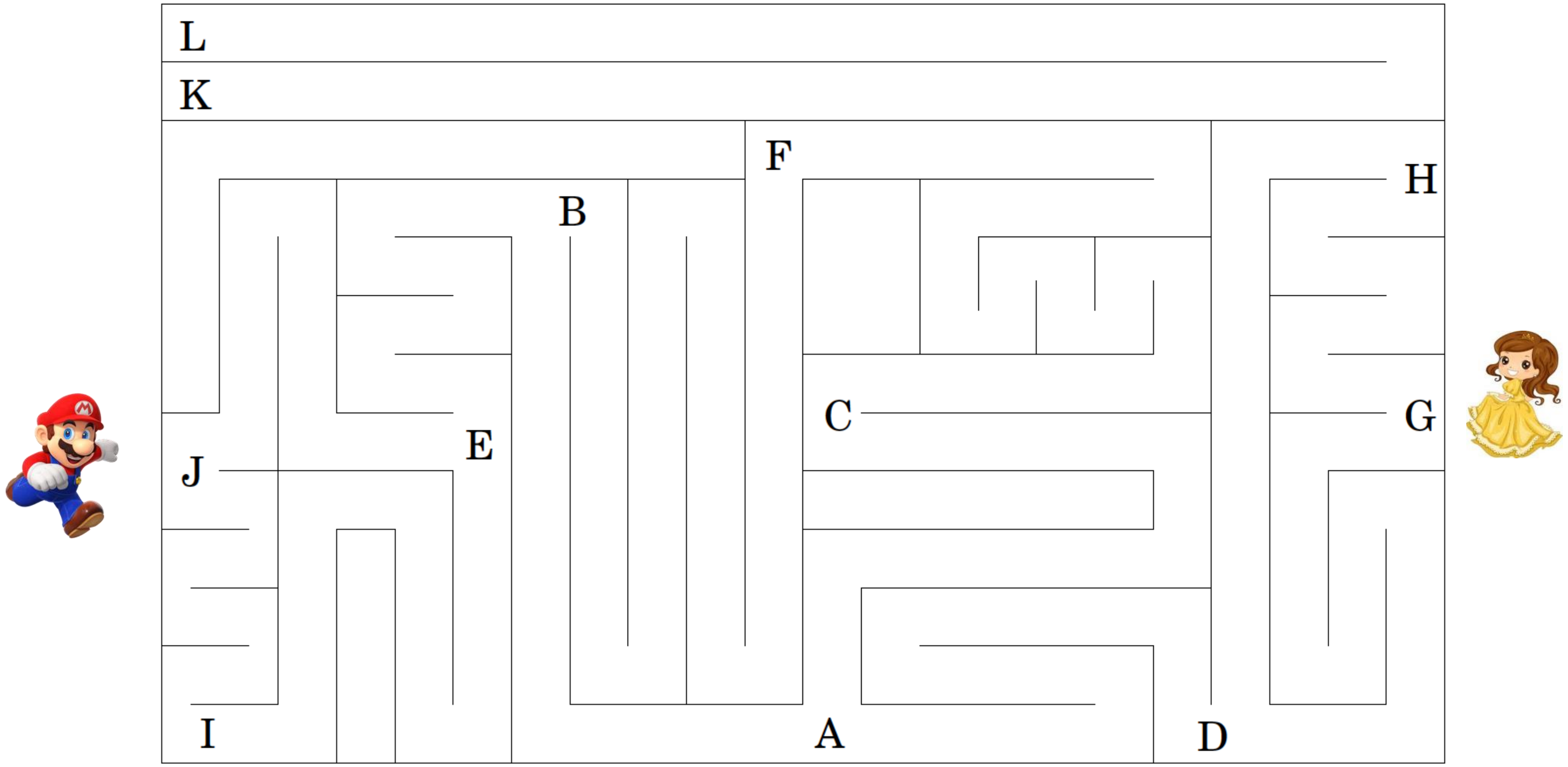
(1) Is there any path from Mario to Bell?

(2) Can Mario reach Bell in 4 steps (letters)?

(3) If not, the minimal steps?

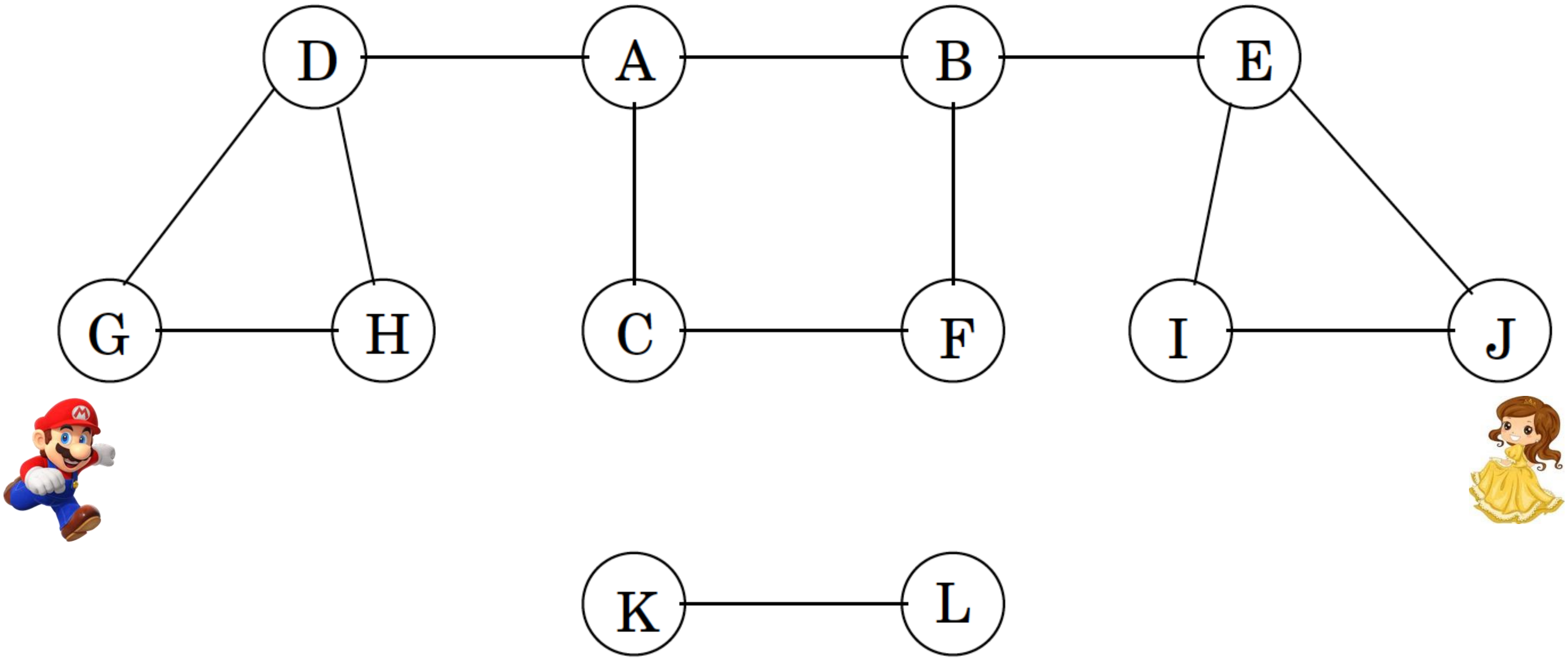
Let’s spend 10 mins to help Mario.

Meeting princess

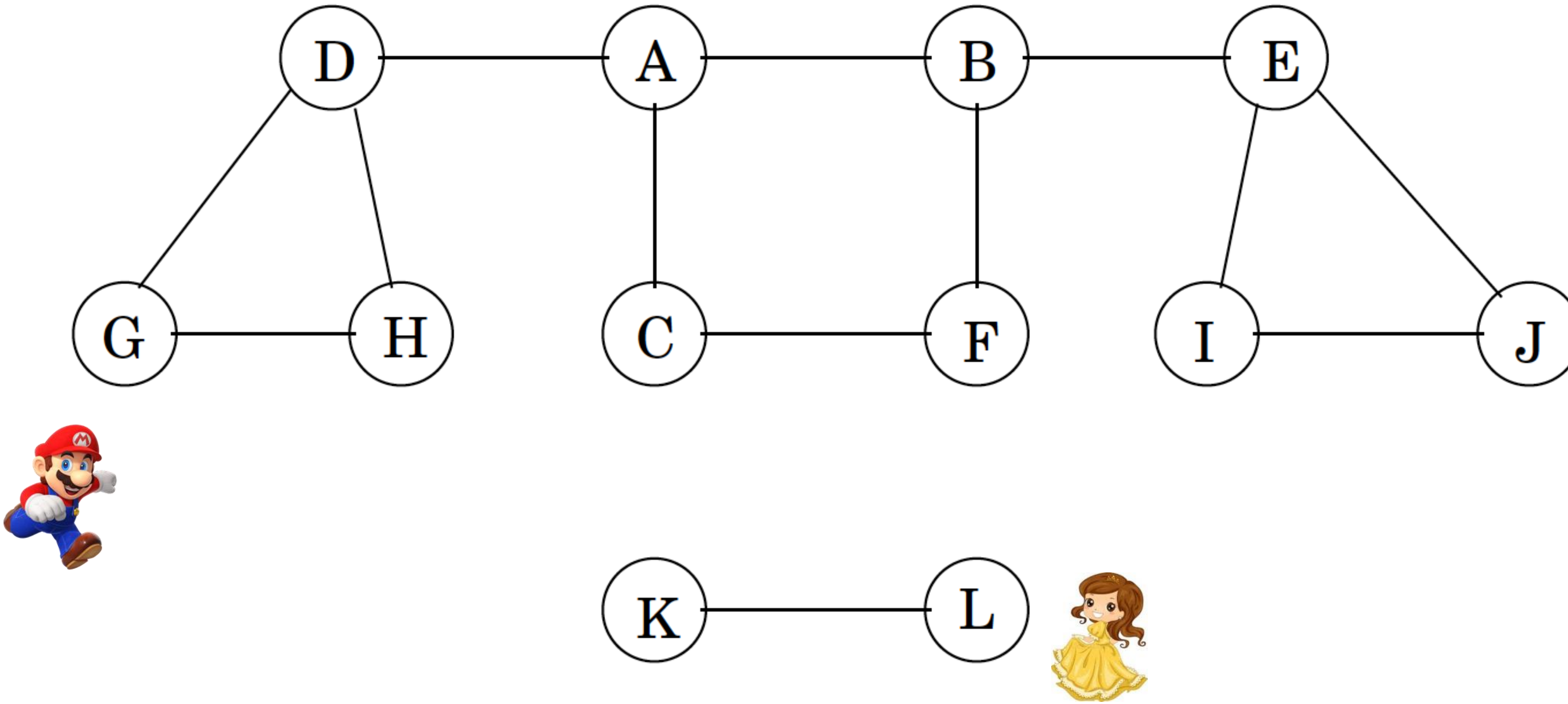


We need to build a graph.

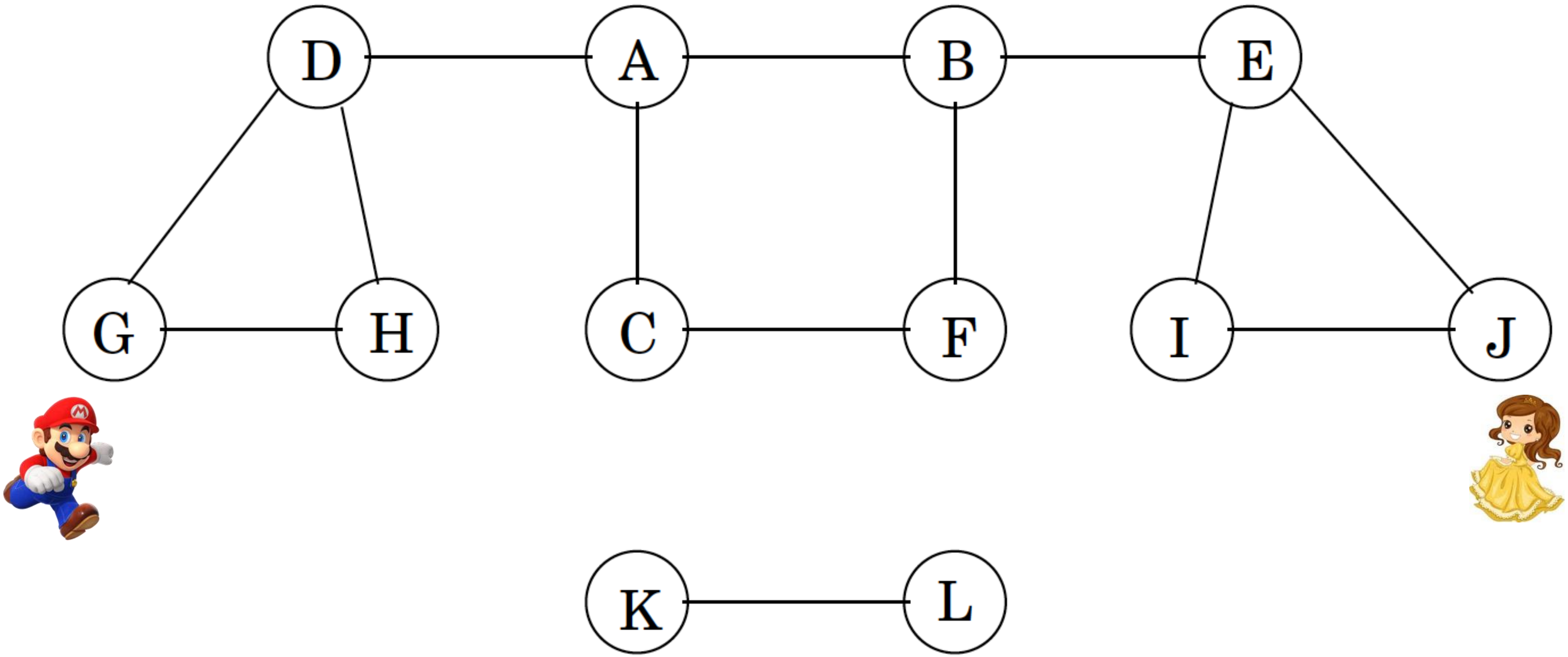
Meeting princess



Meeting princess



Meeting princess



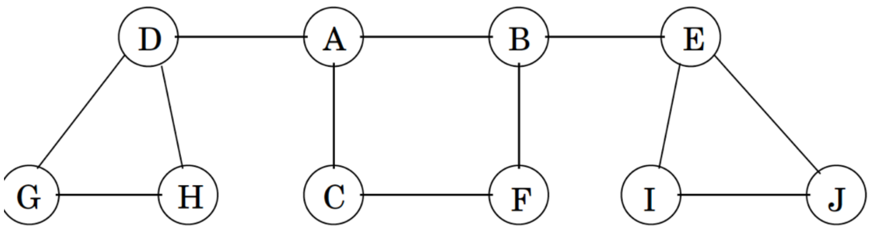
for all u $\in V = \{A, B, C, D, E, F, G, H, I, J\}$;

$$dist(u) = \infty$$

$$dist(G) = 0$$

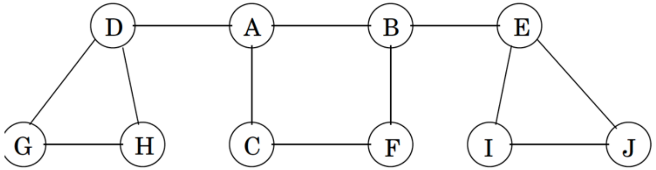
$Q = [G]$ (queue only contains G)

Initialization.



1. $u = G$,
2. $(G, D) \in \boldsymbol{E}$, **and** $dist(D) = \infty$
3. $Q = [D], dist(D) = dist(G) + 1 = 1$
4. $(G, H) \in \boldsymbol{E}$, **and** $dist(H) = \infty$
5. $Q = [D, H], dist(H) = dist(G) + 1 = 1$

Meeting princess BFS



5. $Q = [D, H], dist(H) = dist(G) + 1 = 1$

6. $u = D,$

7. $(D, A) \in \boldsymbol{E},$ **and** $dist(A) = \infty$

8. $Q = [H, A], dist(A) = dist(D) + 1 = 2$

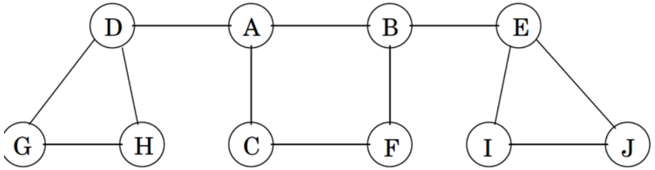
9. $(D, H) \in \boldsymbol{E},$ **and** $dist(H) = 1$ →**continue**

10. $u = H$

11. $(H, D) \in \boldsymbol{E},$ **and** $dist(D) = 1$ →**continue**

12. $Q = [A]$

Meeting princess BFS



12. $Q = [A]$

13. $u = A,$

14. $(A, B) \in \boldsymbol{E}$, **and** $dist(B) = \infty$

15. $Q = [B], dist(B) = dist(A) + 1 = 3$

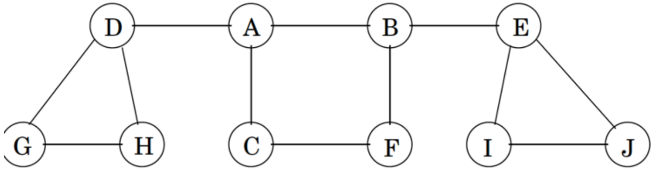
16. $(A, C) \in \boldsymbol{E}$, **and** $dist(C) = \infty$

17. $Q = [B, C], dist(C) = dist(A) + 1 = 3$

18. $(A, D) \in \boldsymbol{E}$, **and** $dist(D) = 1$ → **continue**

19. $Q = [B, C]$

Meeting princess BFS



19. $Q = [B, C]$ What is the next?

20. $u = B$,

21. $(B, E) \in \boldsymbol{E}$, **and** $dist(E) = \infty$

22. $Q = [C, E], dist(E) = dist(B) + 1 = 4$

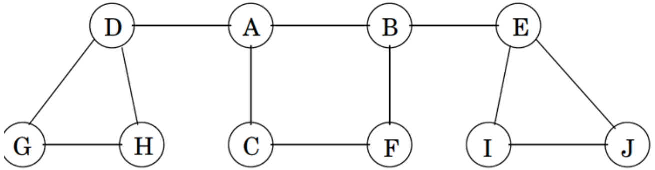
23. $(B, F) \in \boldsymbol{E}$, **and** $dist(F) = \infty$

24. $Q = [C, E, F], dist(F) = dist(B) + 1 = 4$

25. $(B, A) \in \boldsymbol{E}$, **and** $dist(A) = 3$ → **continue**

26. $Q = [C, E, F]$

Meeting princess BFS



26. $Q = [C, E, F]$

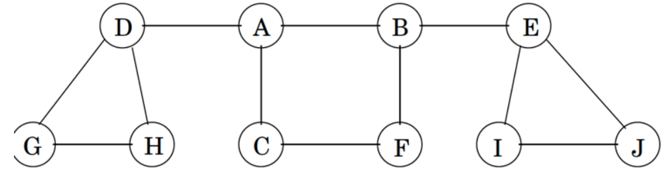
27. $u = C,$

28. $(C, A) \in \boldsymbol{E}, \textbf{and } dist(A) = 3$ → **continue**

29. $(C, F) \in \boldsymbol{E}, \textbf{and } dist(F) = 4$ → **continue**

30. $Q = [E, F]$

Meeting princess BFS



30. $Q = [E, F]$

31. $u = E,$

32. $(E, I) \in \boldsymbol{E}, \textbf{and } dist(I) = \infty$

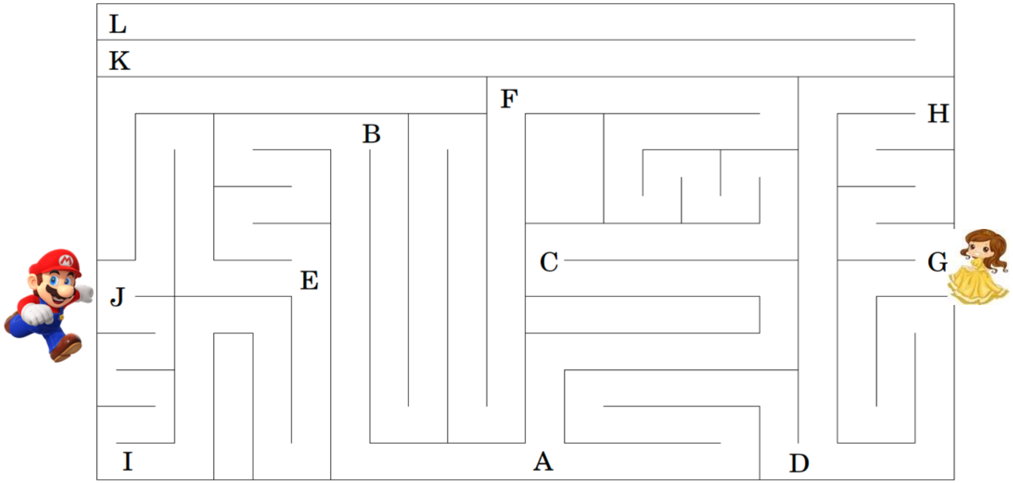
33. $Q = [F, I], dist(I) = dist(E) + 1 = 5$

34. $(E, J) \in \boldsymbol{E}, \textbf{and } dist(J) = \infty$

35. $Q = [F, I, J], dist(J) = dist(E) + 1 = 5$

For the rest of the elements in Q …

Meeting princes



Here are some questions:

(1) Is there any path from Mario to Bell? Yes

(2) Can Mario reach Bell in 4 steps? No

(3) If not, the minimal steps? 5

Main questions about BFS

(1) Is it correct?

Are you kidding me?

(2) Is it efficient?

Can we prove that this algorithm is correct?

Is it correct?

For each d=0,1,2,…, these is a moment at which

(1) all nodes at distance $\leq d$ from s have their distance correctly set;
(2) all other nodes have their distance set to ∞;
(3) The queue contains exactly the nodes at distance d.

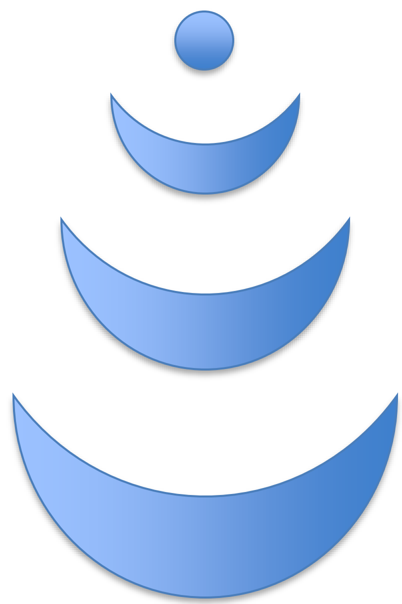
We will use an induction on distance to prove?

$Q = [s]$ d=0

$Q = [A_1, A_2, .. A_k]$ d=1

$Q = [B_1, B_2, \dots, B_r]$ d=2

$Q = [C_1, C_2, \dots, C_t]$ d=3



Is it efficient?

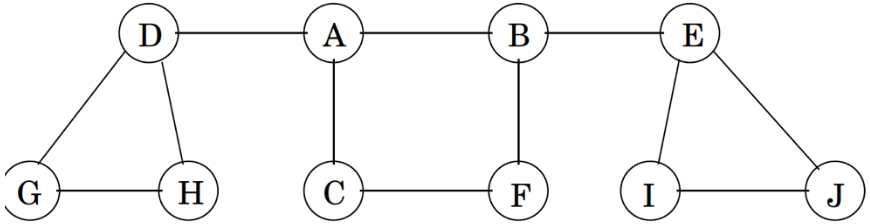
The overall running time of this algorithm is linear, $O(|V| + |E|)$.

The Meeting princess problem

1. $u = G,$
2. $(G, D) \in \boldsymbol{E}$, **and** $dist(D) = \infty$
3. $Q = [D], dist(D) = dist(G) + 1 = 1$
4. $(G, H) \in \boldsymbol{E}$, **and** $dist(H) = \infty$
5. $Q = [D, H], dist(H) = dist(G) + 1 = 1$

Why I only checked (G, D) and (G, H)

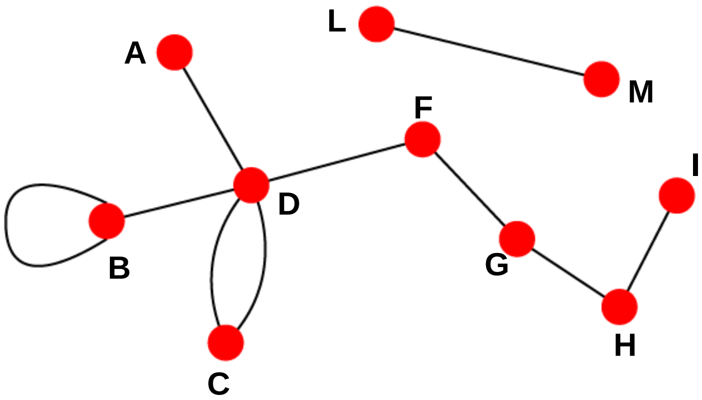
It depends on how you represent the graph?



ADJACENCY MATRIX (Undirected)

Links: undirected (*symmetrical*)

Graph:



Undirected links :

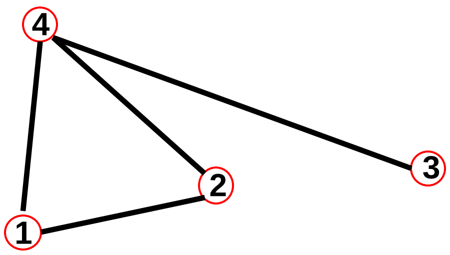
coauthorship links

Actor network

protein interactions

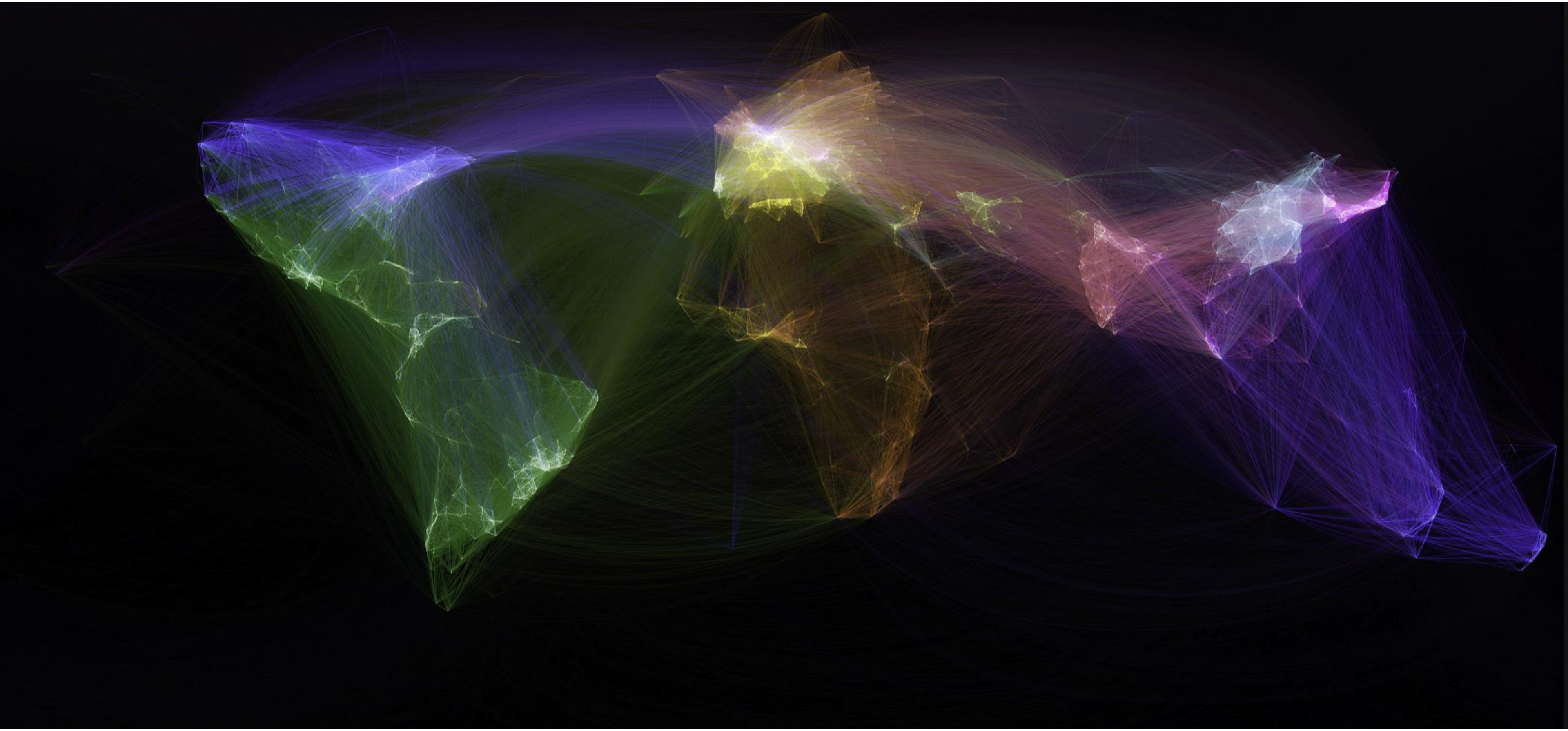
A_{ij}=1 if there is a link between node *i* and *j*

A_{ij}=0 if nodes *i* and *j* are not connected.



$$A_{ij} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

An Example of UNDIRECTED Graphs



Map of scientific collaboration between researchers

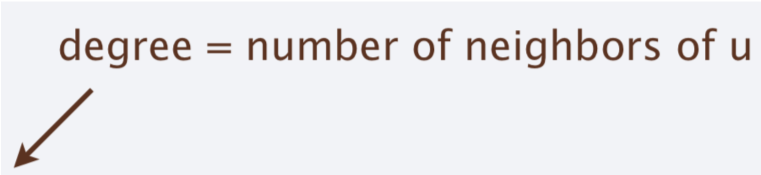
ADJACENCY MATRIX

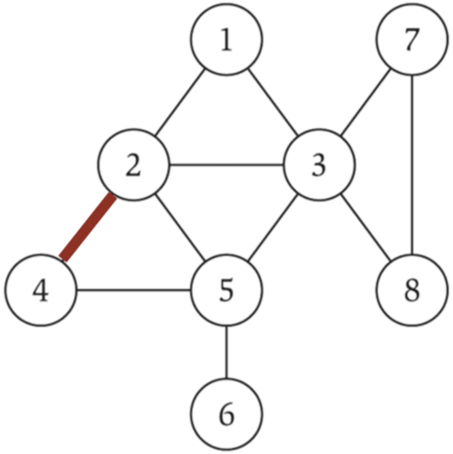
- Space proportional to $\Theta(n^2)$. $(n = |V|)$
- Checking if (u, v) is an edge takes $\Theta(1)$ time
- Identify all edges takes $\Theta(n^2)$ time

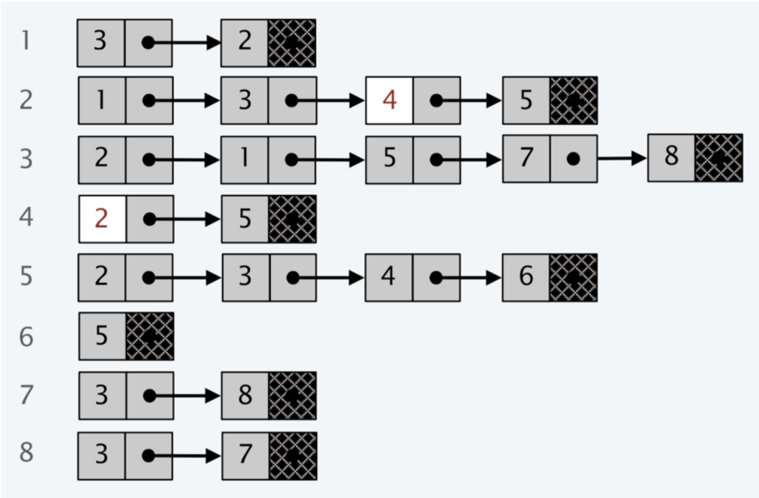
ADJACENCY LISTs

Node-indexed array of lists.

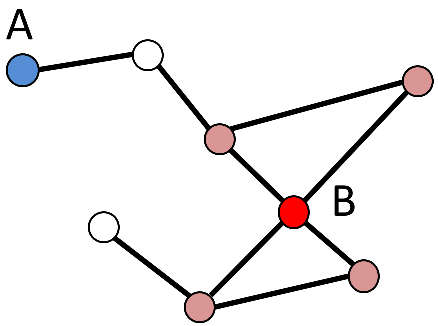
- Space proportional to Θ(|V|+|E|).
- Checking if (*u*, *v*) is an edge takes Θ(degree(u)) time
- Identify all edges takes Θ(|V|+|E|) time







ADJACENCY MATRIX AND NODE DEGREES



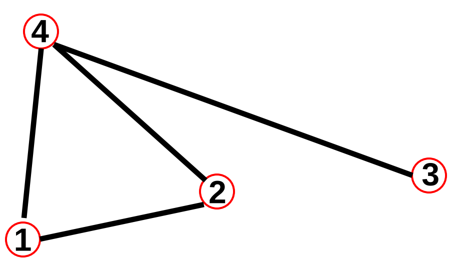
Node degree: the number of links connected to the node.

$$k_A = 1$$

$$k_B = 4$$

Question for a graph $G = (V, E)$:

What is the total degree of a graph?



$$A_{ij} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

$$A_{ij} = A_{ji}$$
$$A_{ii} = 0$$

$$k_i = \sum_{j=1}^{N} A_{ij}$$

$$k_j = \sum_{i=1}^{N} A_{ij}$$

$$L = \frac{1}{2} \sum_{i=1}^{N} k_i = \frac{1}{2} \sum_{ij}^{N} A_{ij}$$

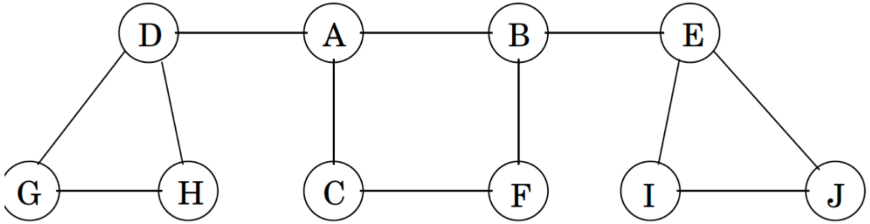
Is it efficient?

The overall running time of this algorithm is linear, $O(|V| + |E|)$.

The Meeting princess problem

1. $u = G,$
2. $(G, D) \in \boldsymbol{E}$, **and** $dist(D) = \infty$
3. $Q = [D], dist(D) = dist(G) + 1 = 1$
4. $(G, H) \in \boldsymbol{E}$, **and** $dist(H) = \infty$
5. $Q = [D, H], dist(H) = dist(G) + 1 = 1$

Should we use adjacency matrix or adjacency list?

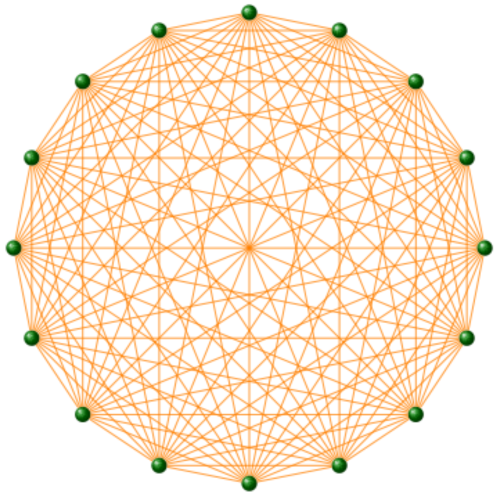


Average Degree

NETWORK	NODES	LINKS	DIRECTED UNDIRECTED	N	L	⟨k⟩
Internet	Routers	Internet connections	Undirected	192,244	609,066	6.33
WWW	Webpages	Links	Directed	325,729	1,497,134	4.60
Power Grid	Power plants, transformers	Cables	Undirected	4,941	6,594	2.67
Mobile Phone Calls	Subscribers	Calls	Directed	36,595	91,826	2.51
Email	Email addresses	Emails	Directed	57,194	103,731	1.81
Science Collaboration	Scientists	Co-authorship	Undirected	23,133	93,439	8.08
Actor Network	Actors	Co-acting	Undirected	702,388	29,397,908	83.71
Citation Network	Paper	Citations	Directed	449,673	4,689,479	10.43
E. Coli Metabolism	Metabolites	Chemical reactions	Directed	1,039	5,802	5.58
Protein Interactions	Proteins	Binding interactions	Undirected	2,018	2,930	2.90

COMPLETE GRAPH

The maximum number of links a network of N nodes can have is: $L_{\max} = \binom{N}{2} = \frac{N(N-1)}{2}$



A graph with degree $L=L_{max}$ is called a complete graph, and its average degree is **<k>=N-1**

REAL NETWORKS ARE SPARSE

Most networks observed in real systems are sparse:

$$L \ll L_{max}$$

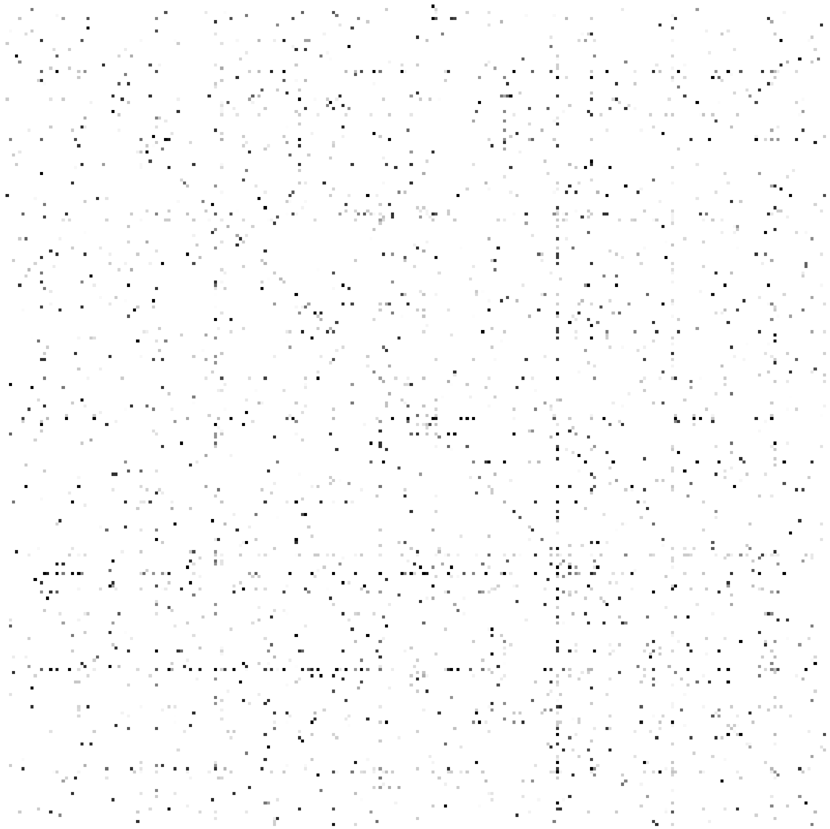
or

$$\langle k \rangle \ll N-1.$$

WWW (ND Sample):	N=325,729;	$L=1.4\ 10^6$	$L_{max}=10^{12}$	$\langle k \rangle=4.51$
Protein (*S. Cerevisiae*):	N= 1,870;	L=4,470	$L_{max}=10^7$	$\langle k \rangle=2.39$
Coauthorship (Math):	N= 70,975;	$L=2\ 10^5$	$L_{max}=3\ 10^{10}$	$\langle k \rangle=3.9$
Movie Actors:	N=212,250;	$L=6\ 10^6$	$L_{max}=1.8\ 10^{13}$	$\langle k \rangle=28.78$

(Source: Albert, Barabasi, RMP2002)

ADJACENCY MATRICES ARE SPARSE



Is BFS efficient?

for all u $\in V$; $dist(u) = \infty$	$O(n)$	
$dist(s) = 0$	$O(1)$	$O(n)$
$Q = [s]$ (queue only contains s)	$O(1)$	
while Q is not empty	$O(n)$	$O(n)$
$u = eject(Q)$	$O(1)$	
for all edges $(u, v) \in E$:	$O(n)$	$O(n)$
if $dist(v) = \infty$: $inject(Q, v)$ $dist(v) = dist(v)$ + 1	$O(1)$	

Looks like $O(n^2)$, not linear.

Trick!!!

Is BFS efficient?

for all u $\in V$; $\quad dist(u) = \infty$	$O(n)$	
$dist(s) = 0$	$O(1)$	$O(n)$
$Q = [s]$ (queue only contains s)	$O(1)$	
while Q is not empty	$O(n)$	$O(n)$
$\quad u = eject(Q)$	$O(1)$	
$\quad$ for all edges $(u, v) \in E$:	$O(degree(u))$	$O(m)$
$\quad\quad$ if $dist(v) = \infty$: $\quad\quad\quad inject(Q, v)$ $\quad\quad\quad dist(v) = dist(v) + 1$	$O(1)$	

The overall running time of this algorithm is linear, $O(|V| + |E|)$.

Graph Algorithms

Part II

UNDIRECTED NETWORKS

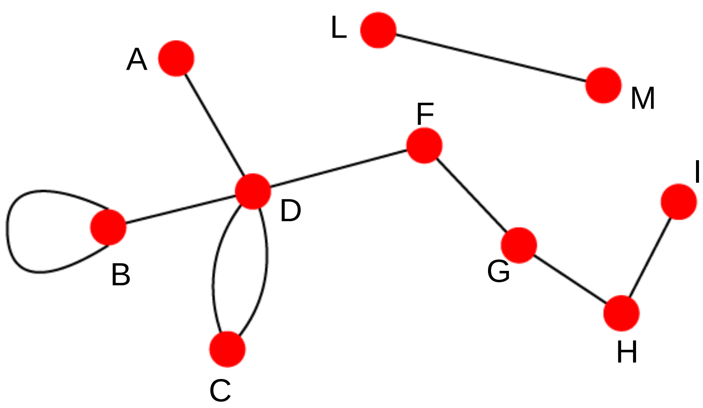
Notation. $G = (V,E)$

V = nodes (or vertices).

E = edges (or arcs) between pairs of nodes

Captures pairwise relationship between objects.

Network size parameters: $n = |V|$, $m = |E|$.



V = {A,B,C,D,E,F,G,H,I, L,M}

E {A-D,B-B,B-D,C-D,C-D,D-F,F-G,G-H,H-I,L-M}

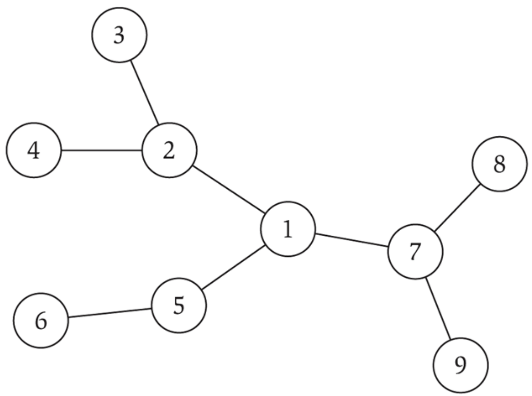
A-D is the same as D-A

n = 10, m = 10.

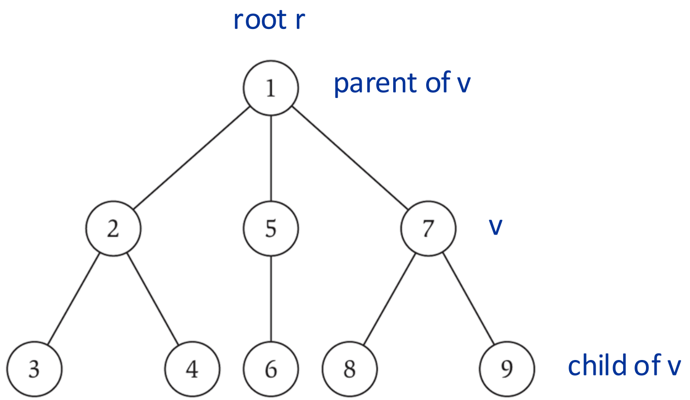
Some graph applications

Graph	*Nodes*	*Edges*
transportation	street intersections	highways
communication	computers	fiber optic cables
World Wide Web	web pages	hyperlinks
social	people	relationships
food web	species	predator-prey
software systems	functions	function calls
scheduling	tasks	precedence constraints
circuits	gates	wires

Tress



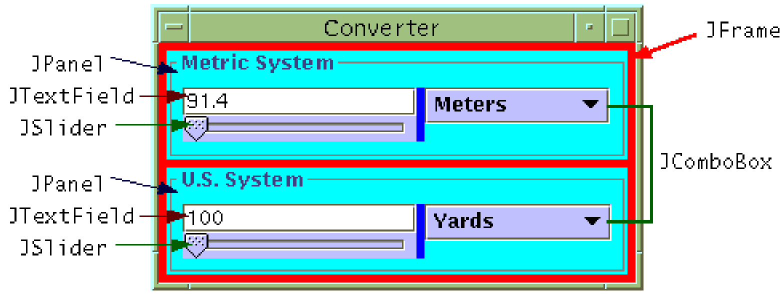
a tree

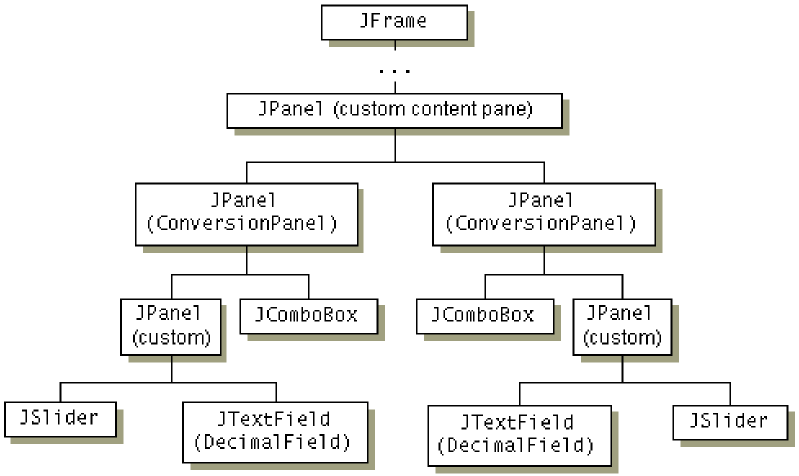


the same tree, rooted at 1

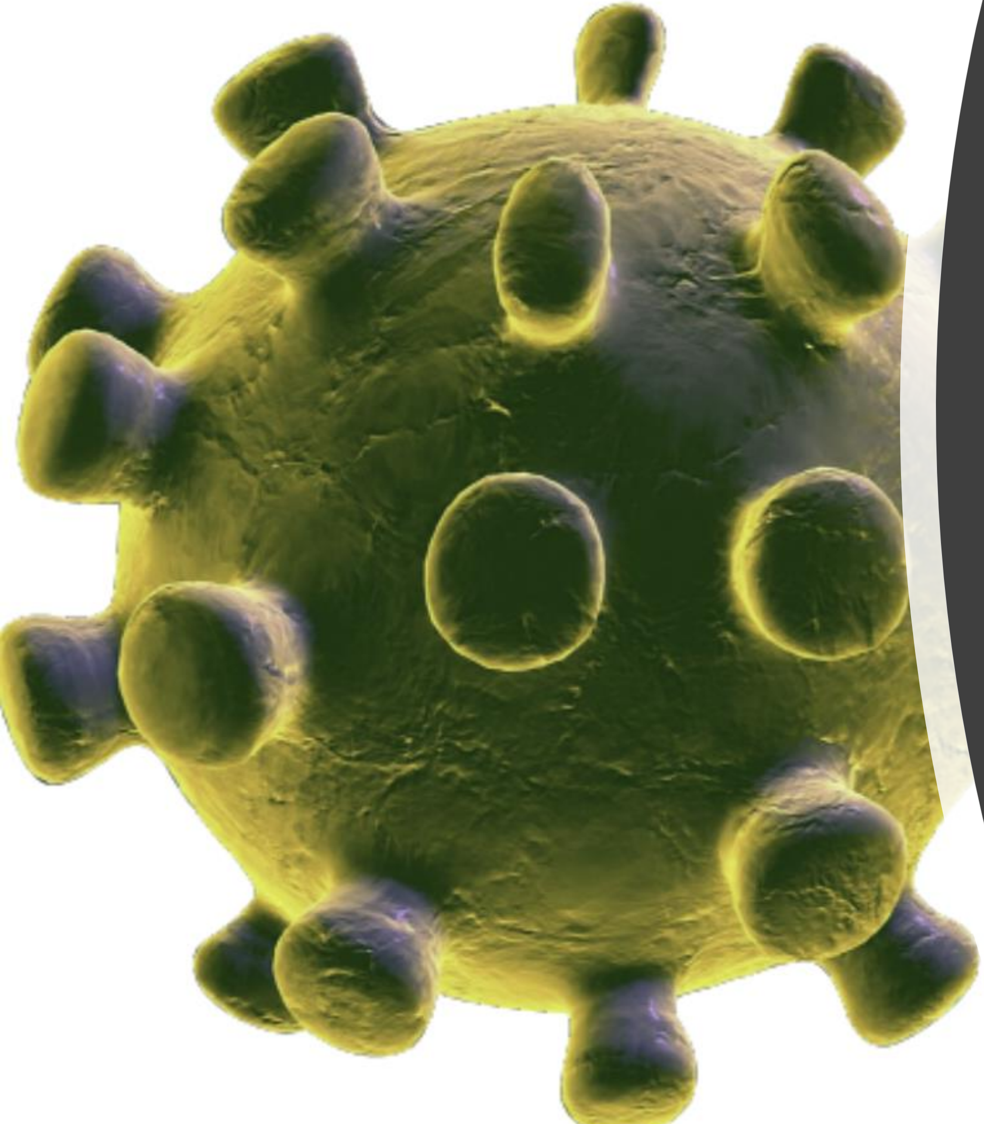
GUI tree

- GUI containment hierarchy. Describe organization of GUI widgets.





Reference: http://java.sun.com/docs/books/tutorial/uiswing/overview/anatomy.html

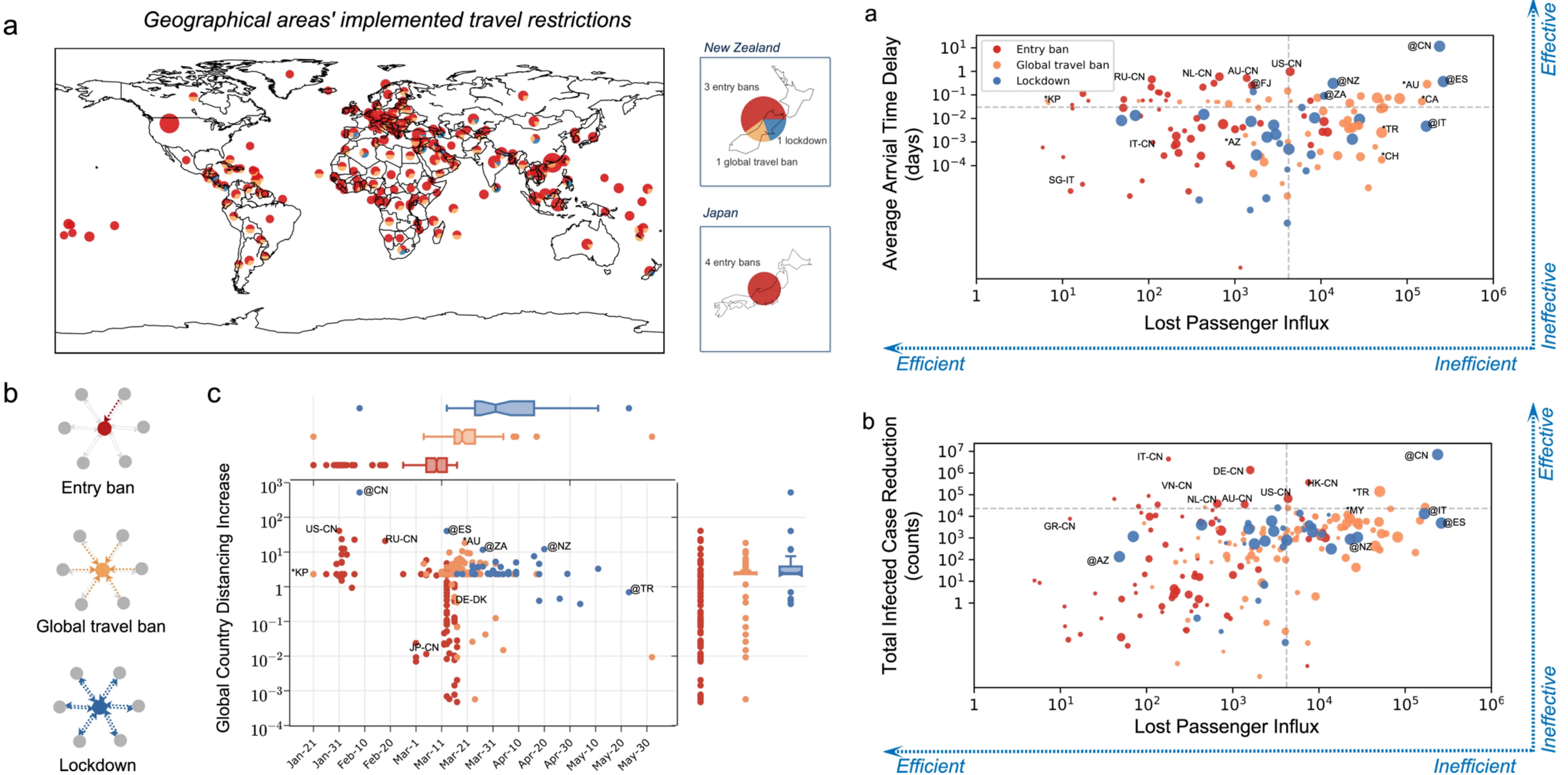


COVID-19

- It spreads so fast because of huge human mobility
- The spreading process seems unpredictable

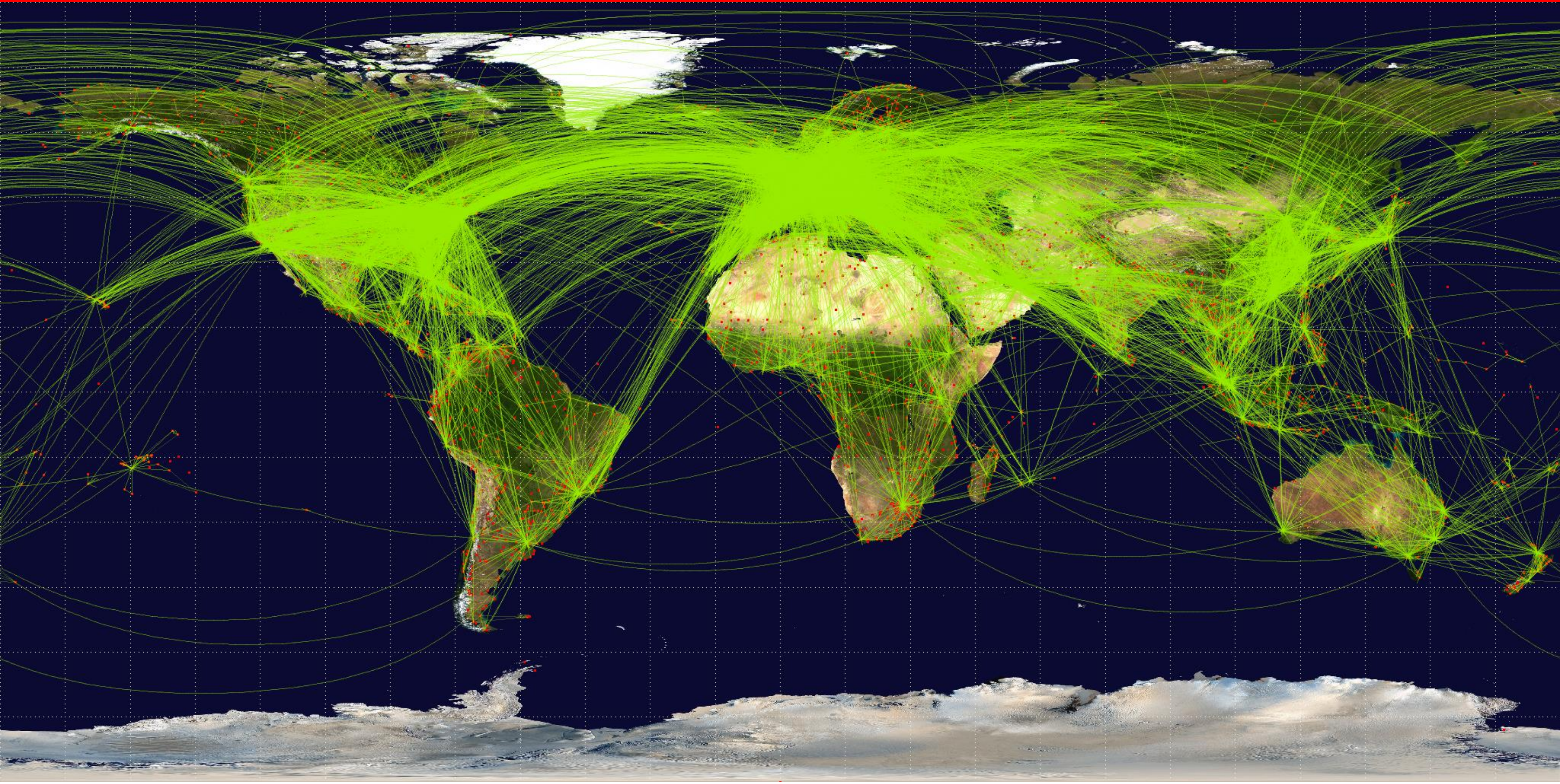
COVID-19

COVID-19: travel restrictions



https://science.rpi.edu/computer-science/news/new-covid-19-model-reveals-need-better-travel-restriction-implementation

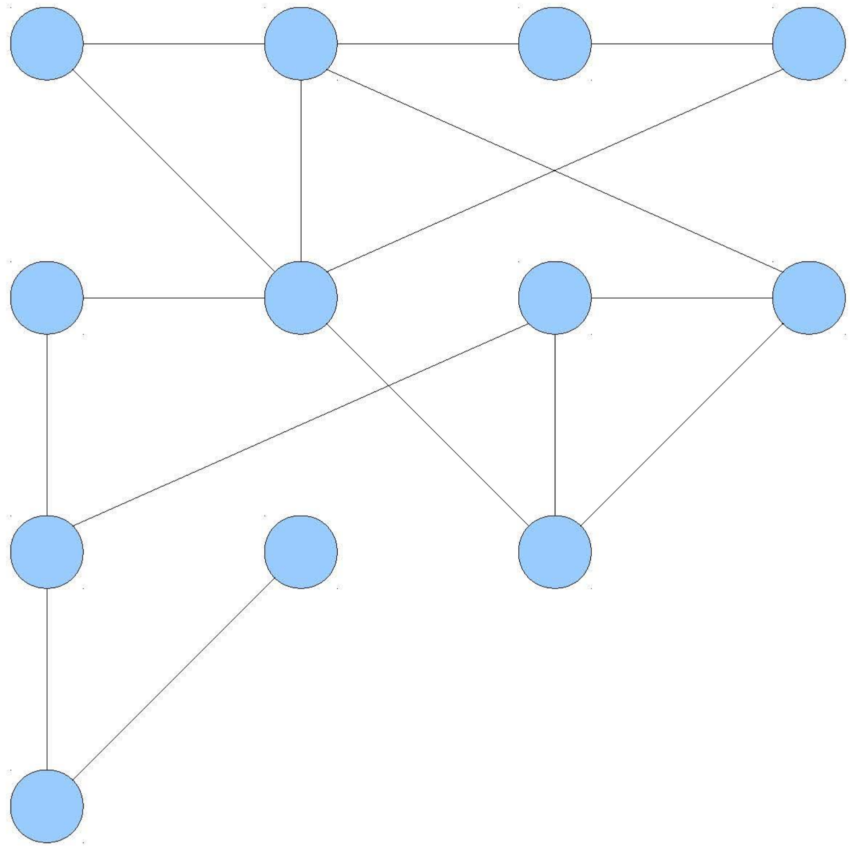
International airline network



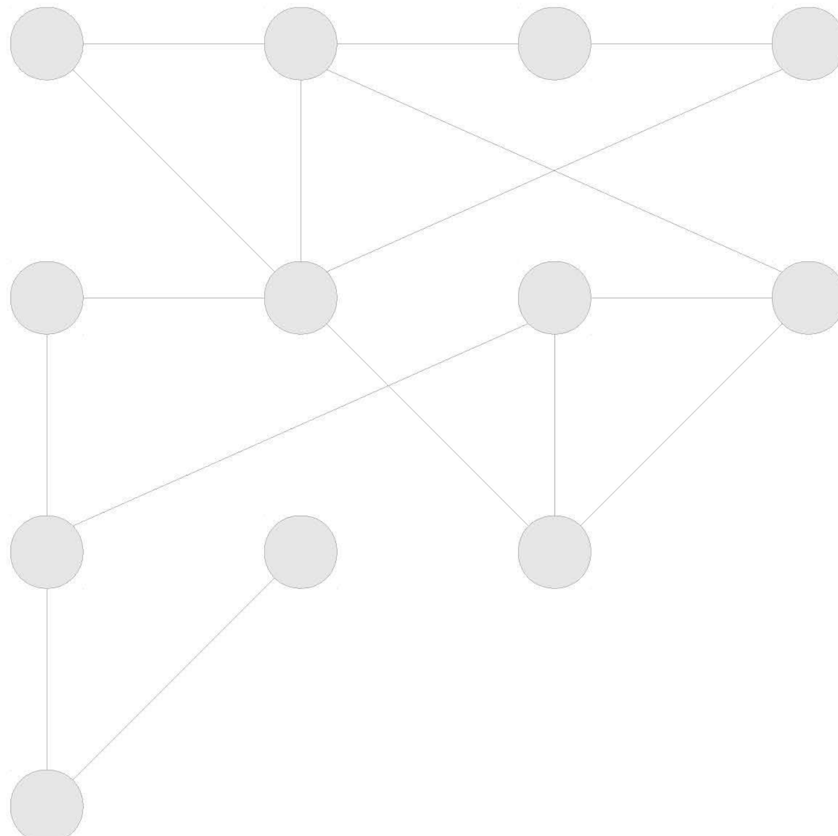
What parts of the graph are reachable
from a given vertex (for example Wuhan)?

Depth-First Search (DFS)

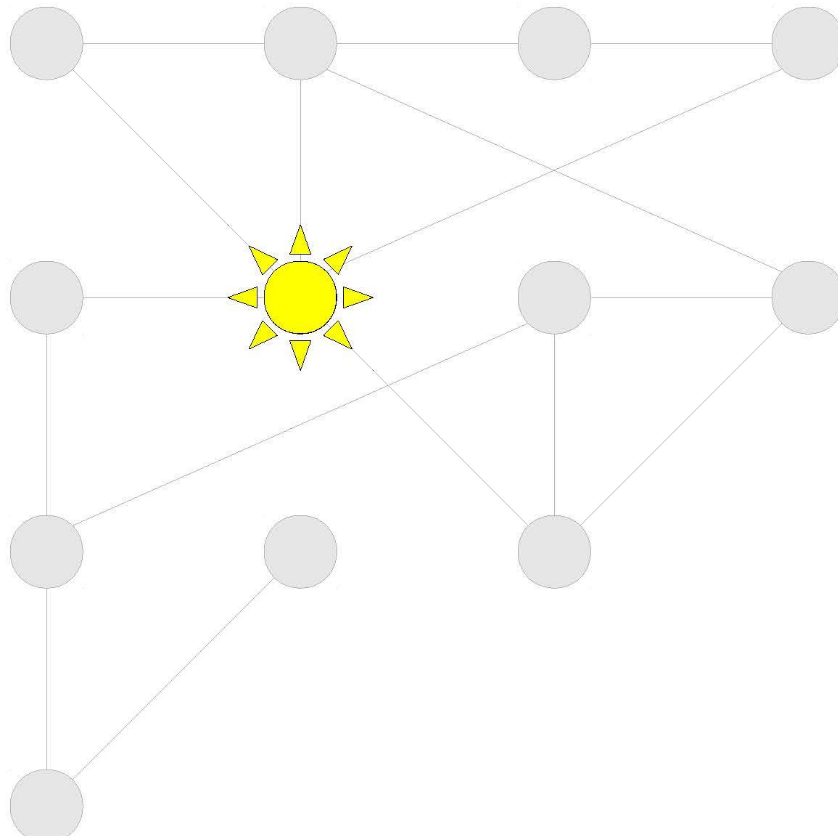
Depth-First Search



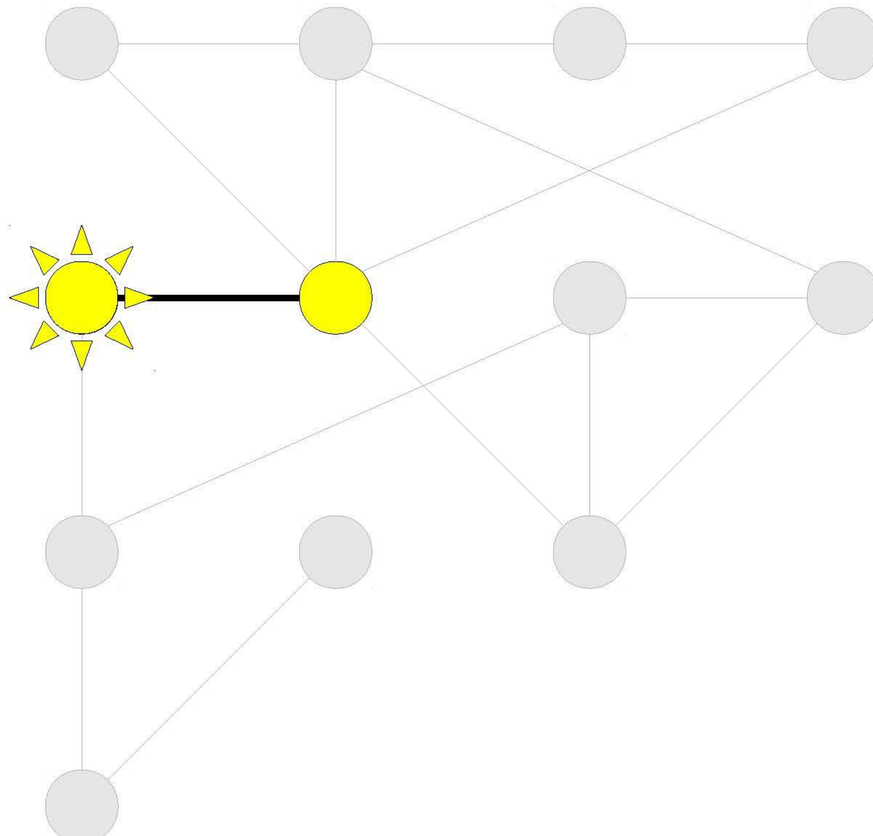
Depth-First Search



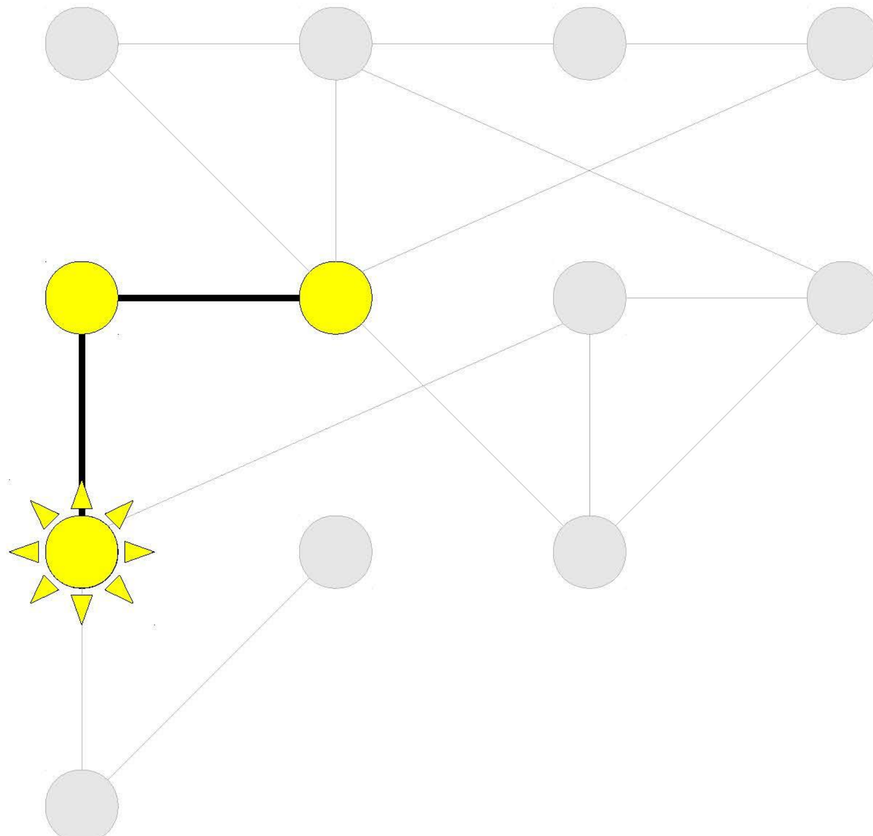
Depth-First Search



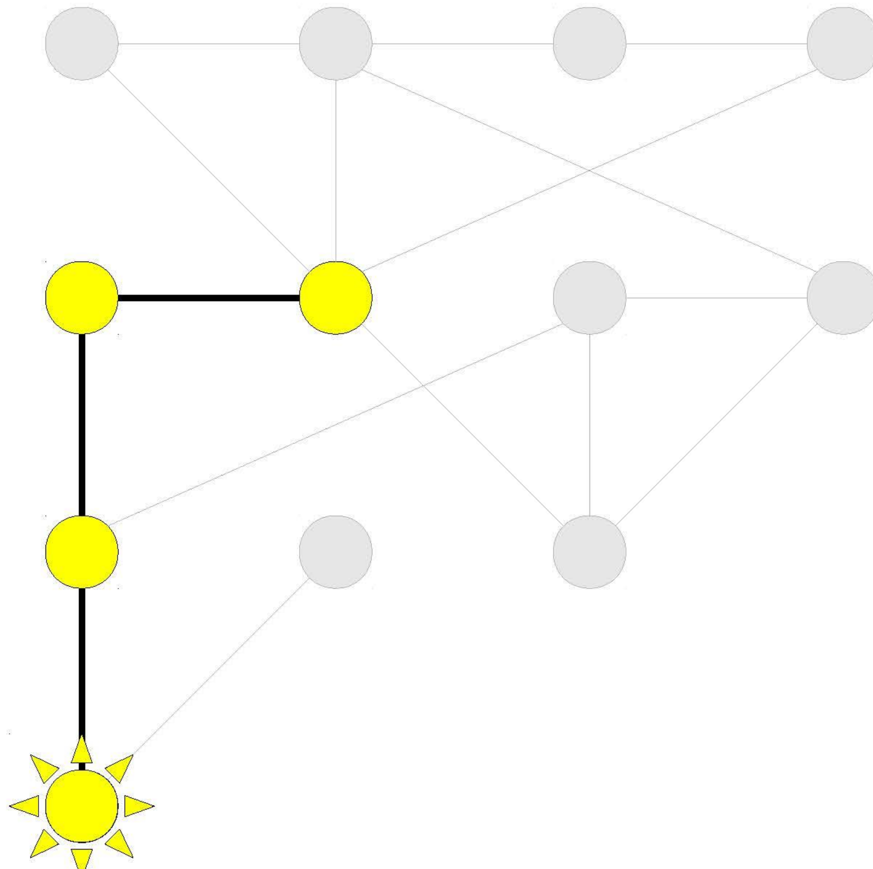
Depth-First Search



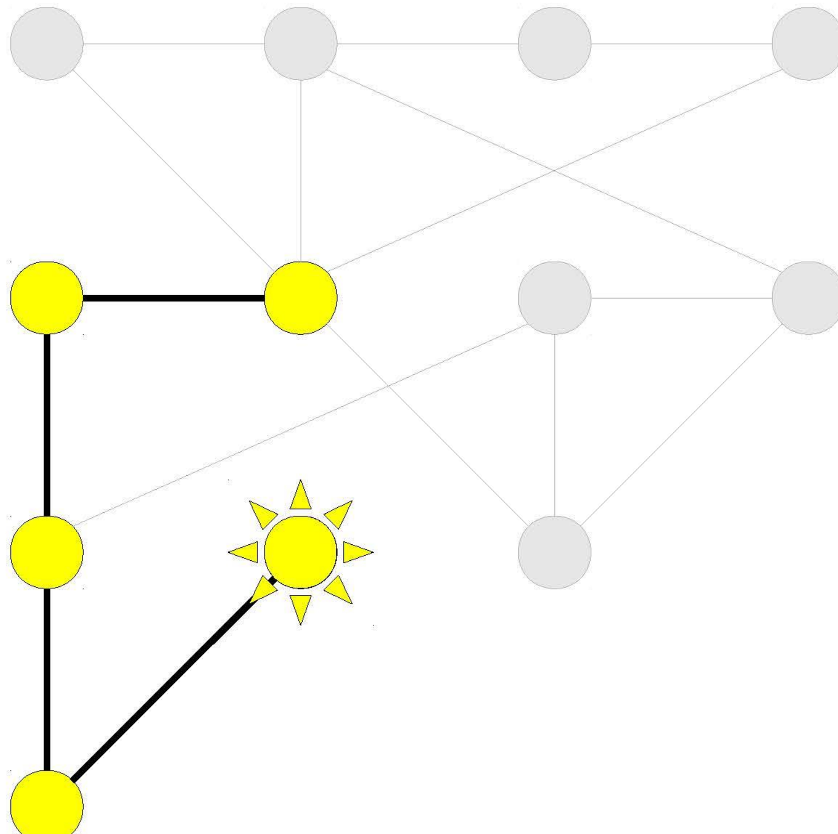
Depth-First Search



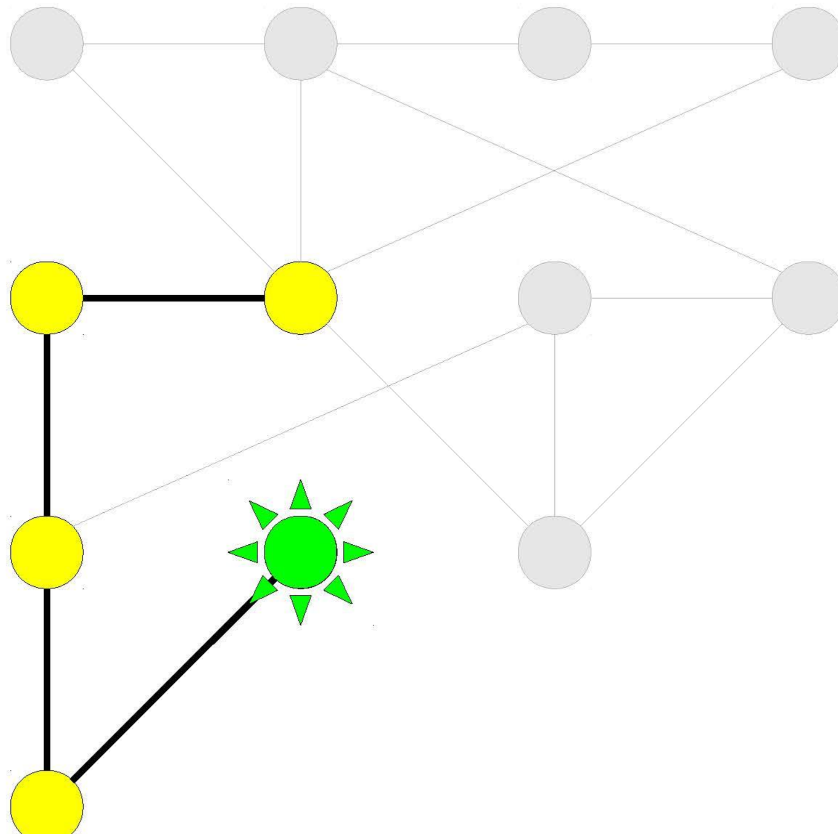
Depth-First Search



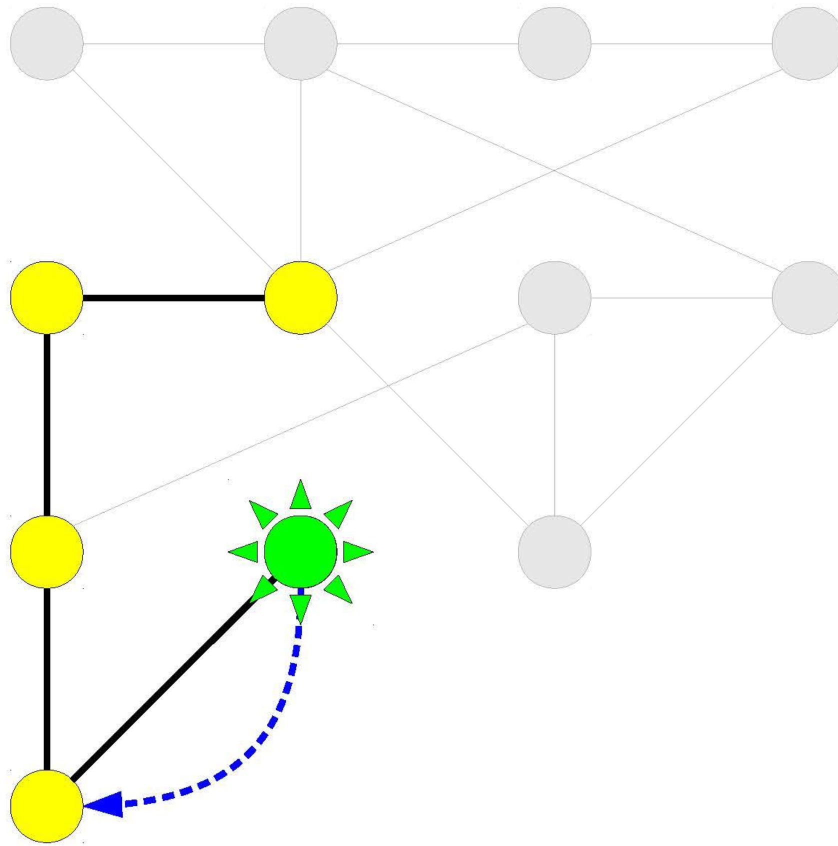
Depth-First Search



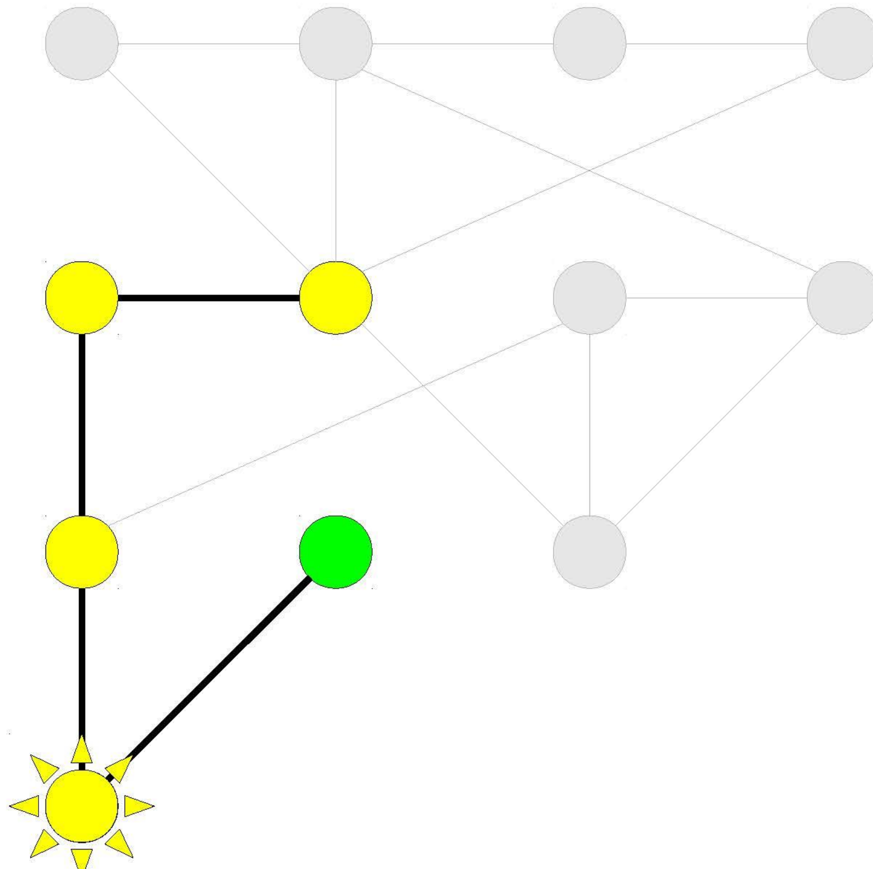
Depth-First Search



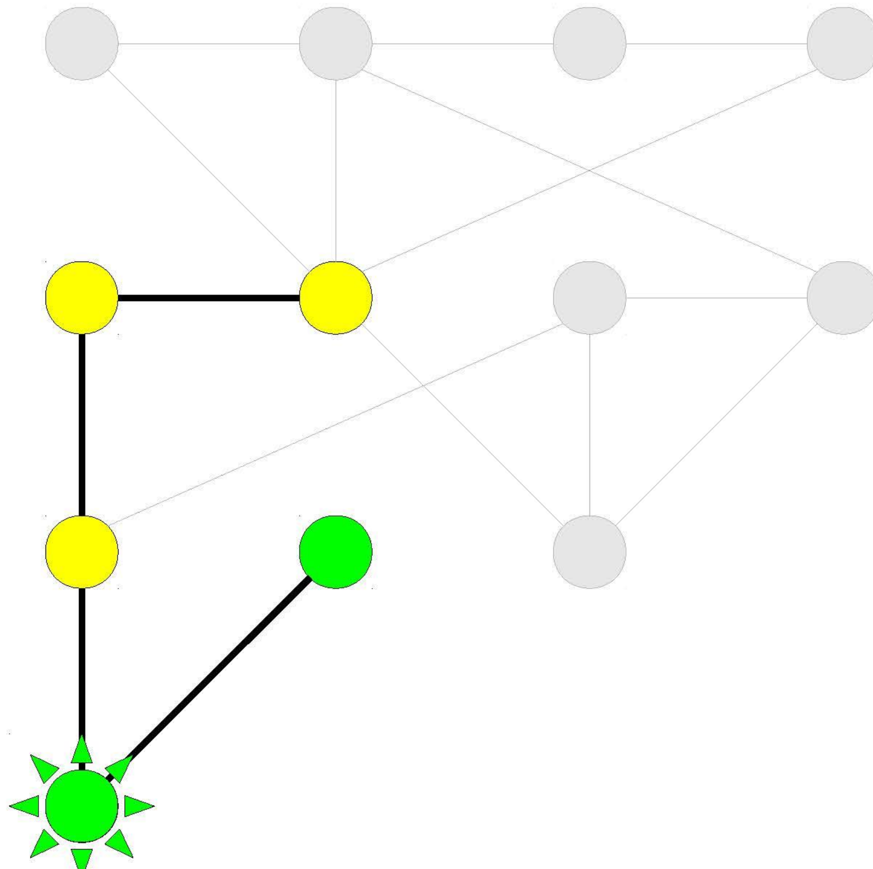
Depth-First Search



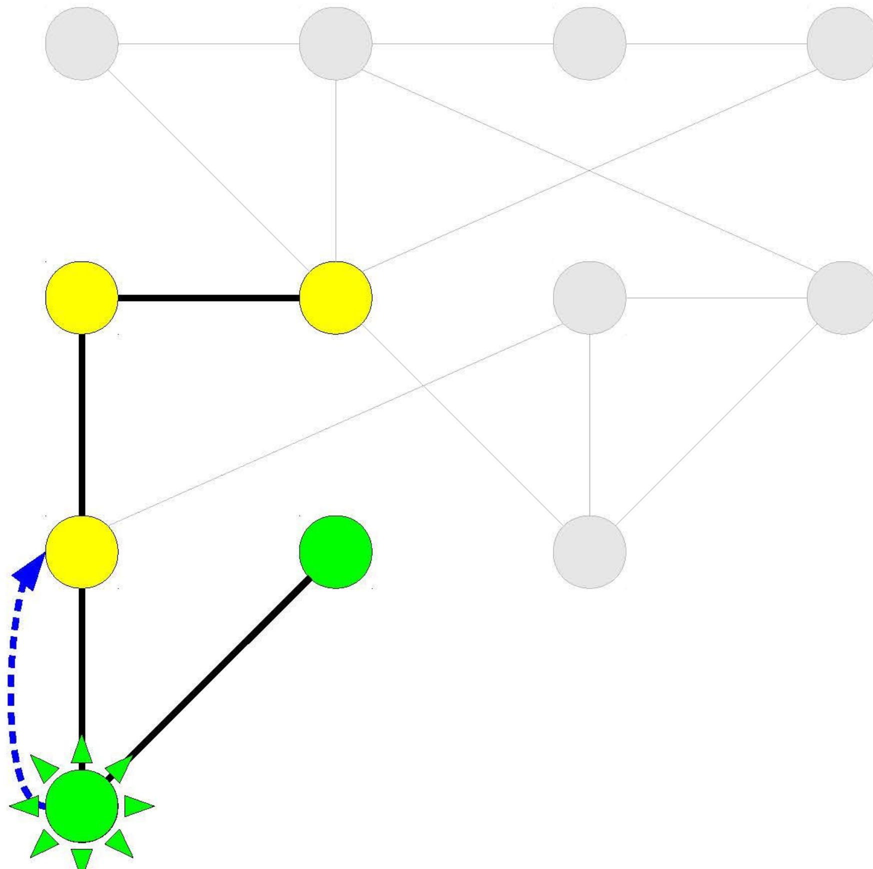
Depth-First Search



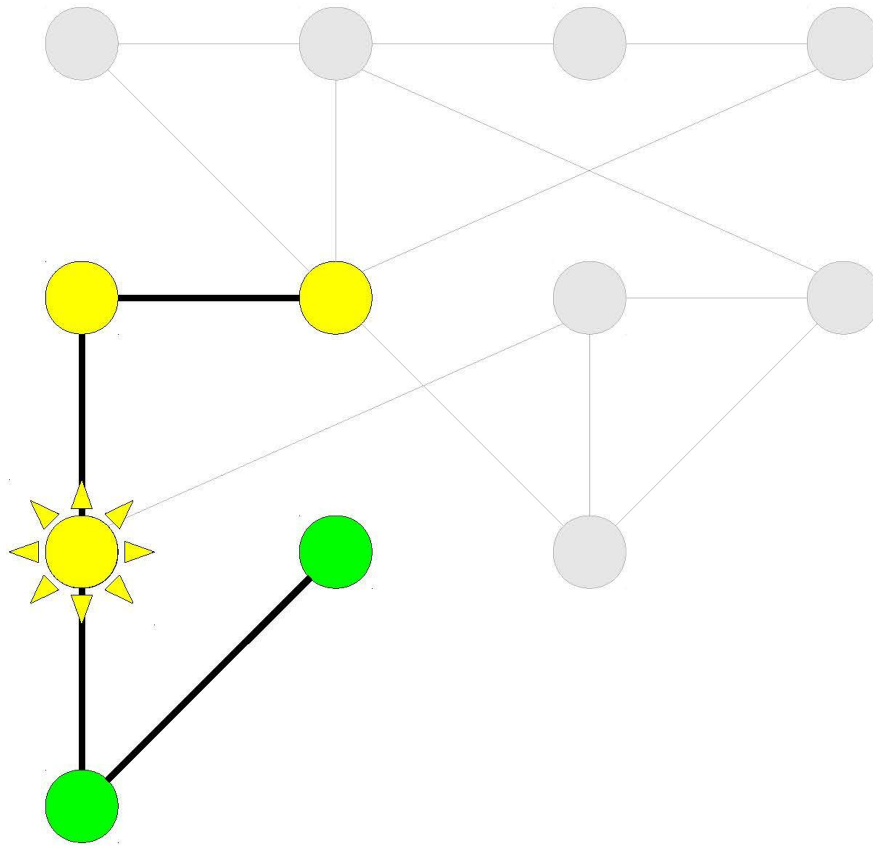
Depth-First Search



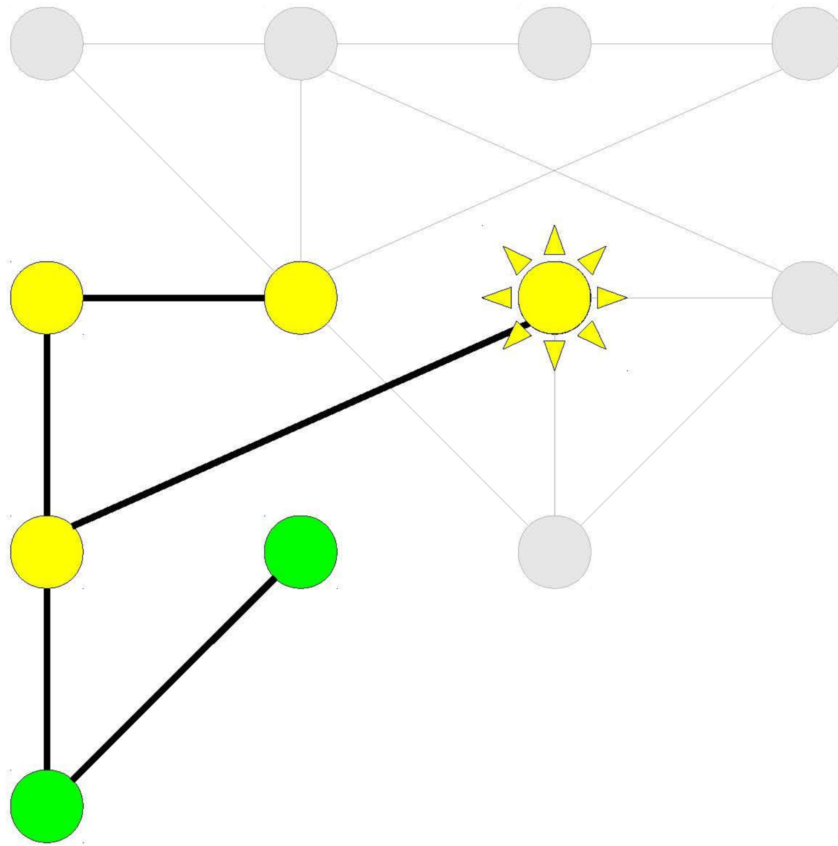
Depth-First Search



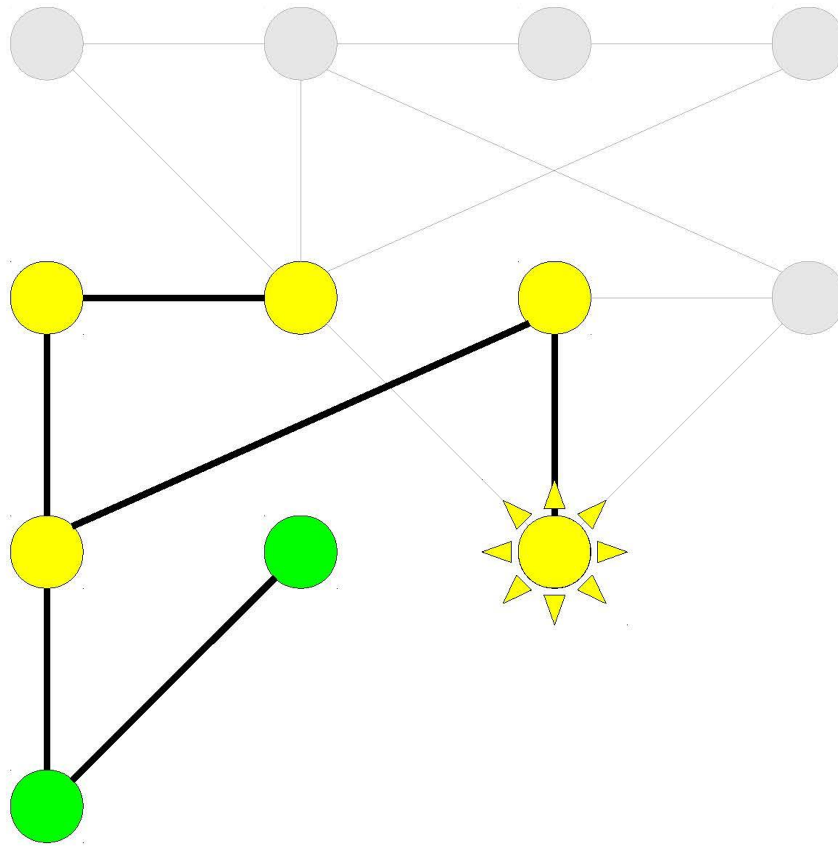
Depth-First Search



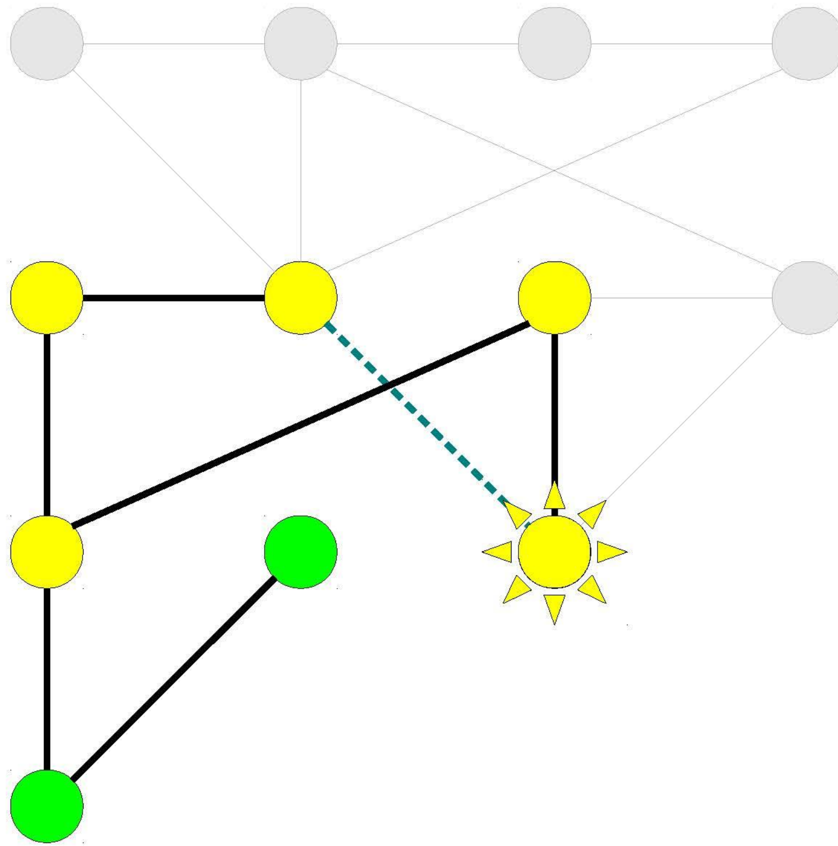
Depth-First Search



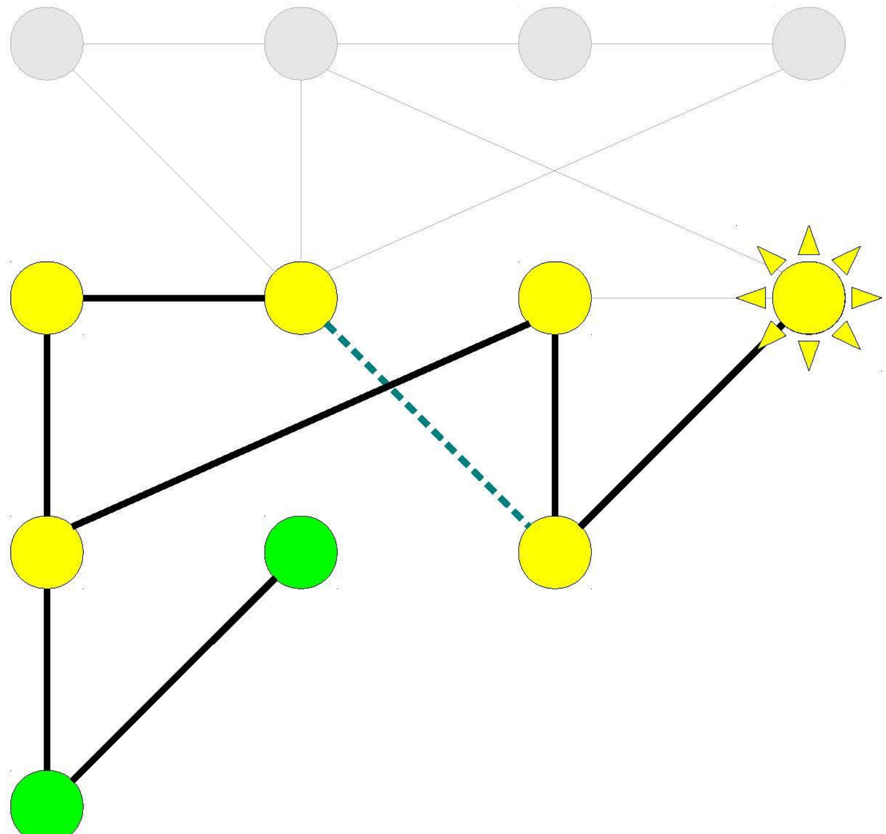
Depth-First Search



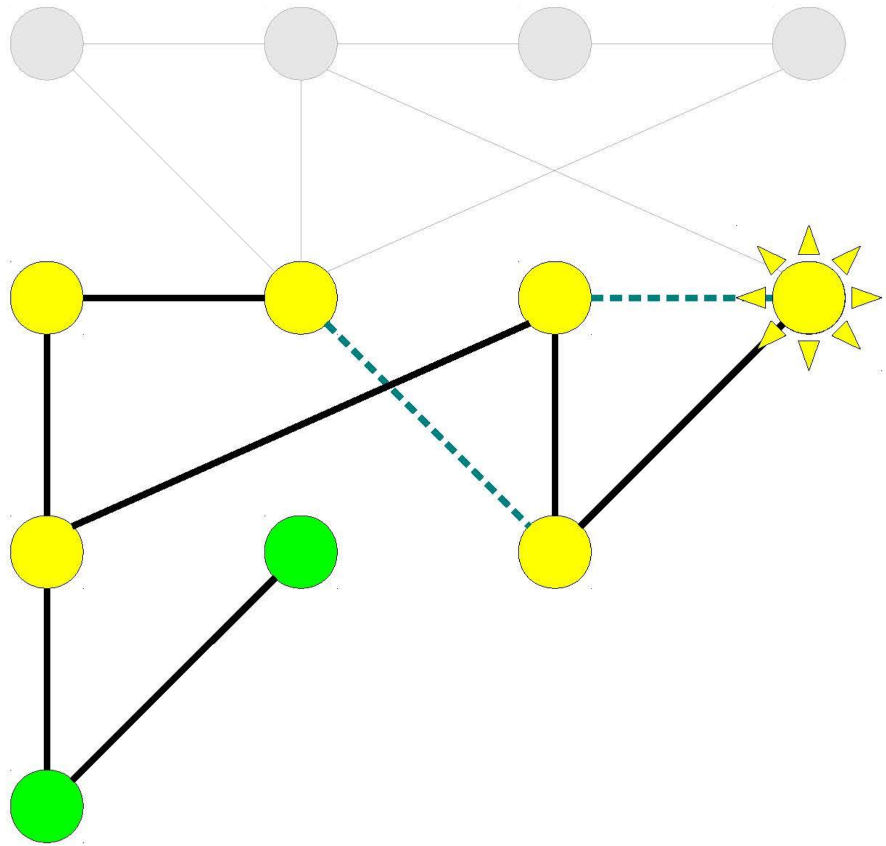
Depth-First Search



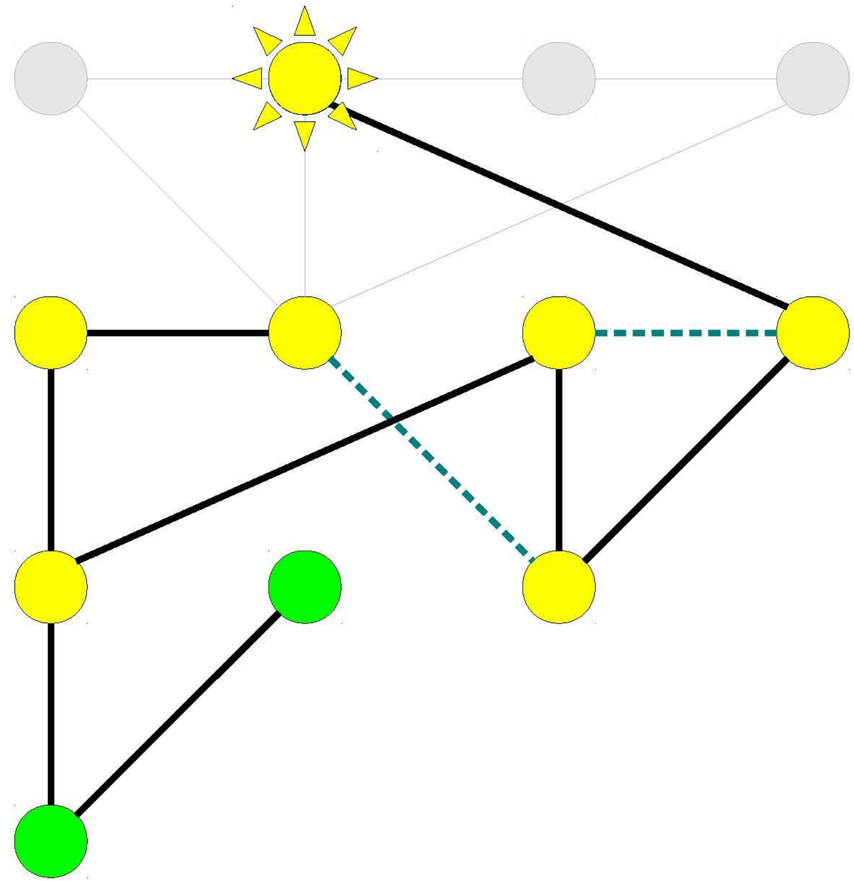
Depth-First Search



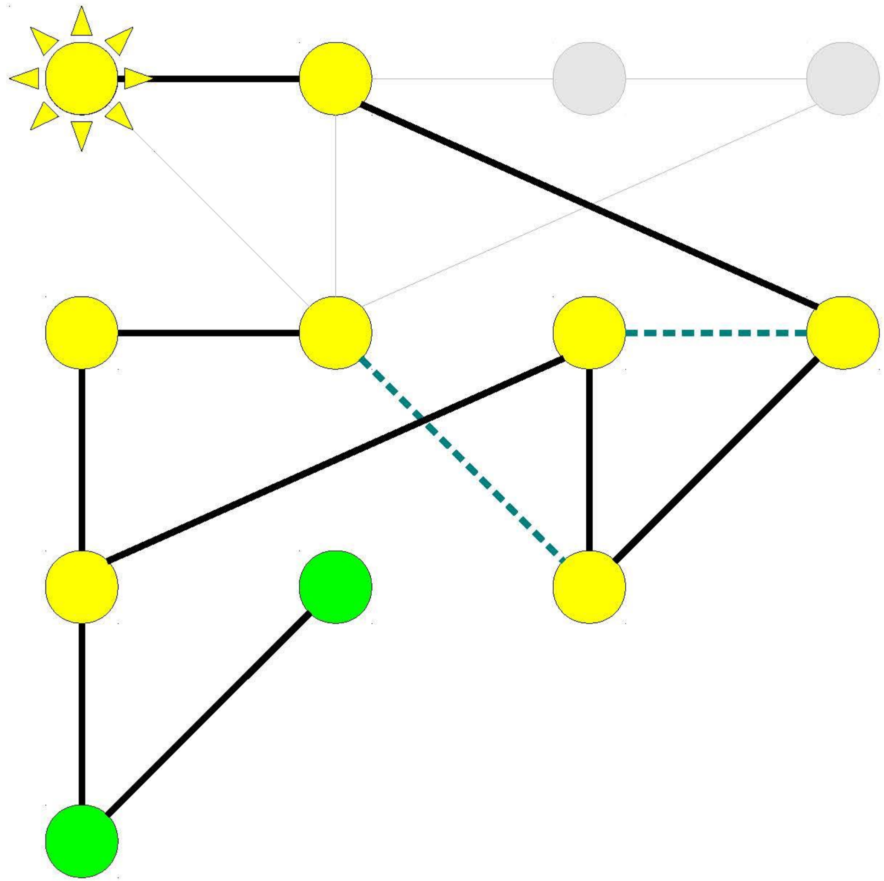
Depth-First Search



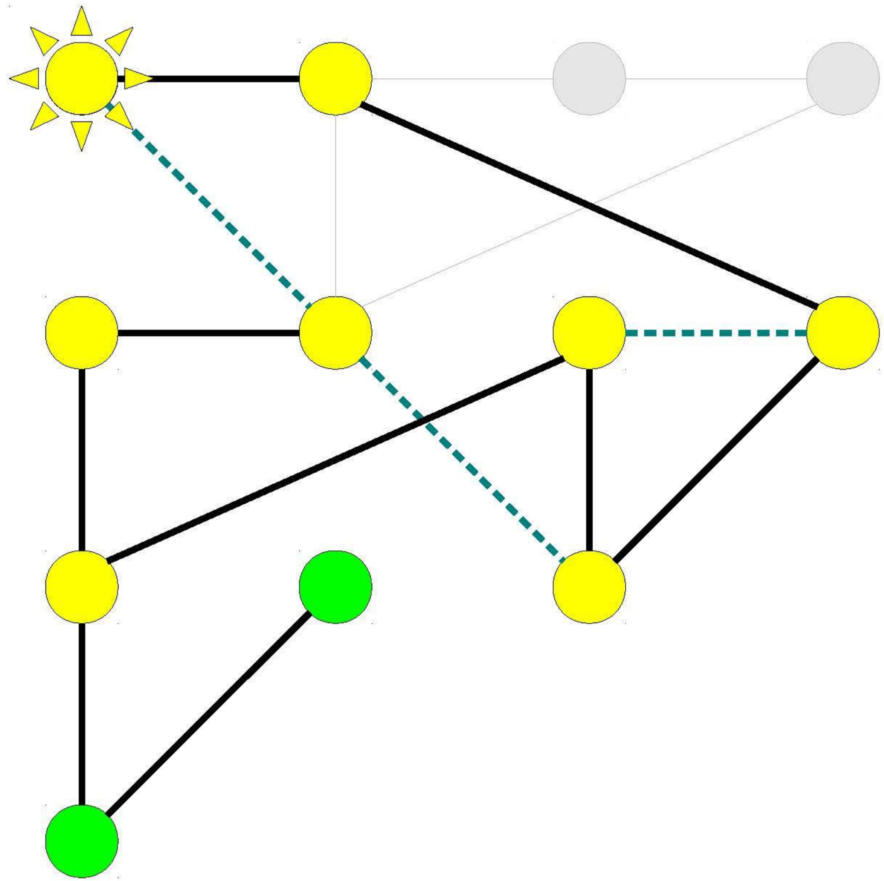
Depth-First Search



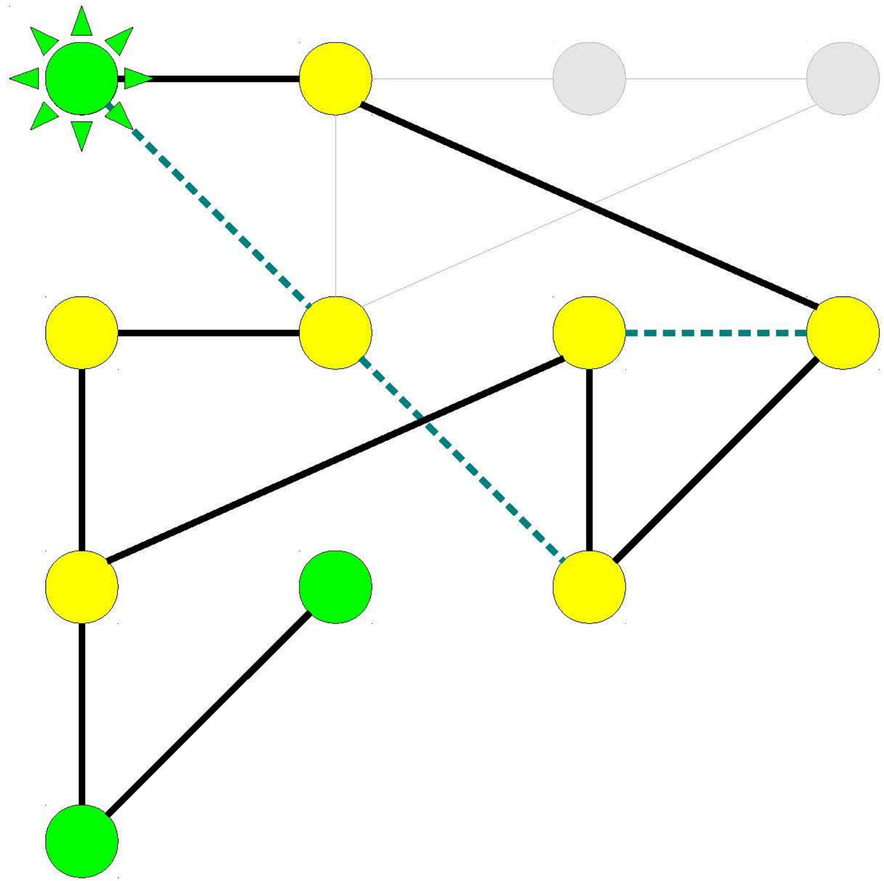
Depth-First Search



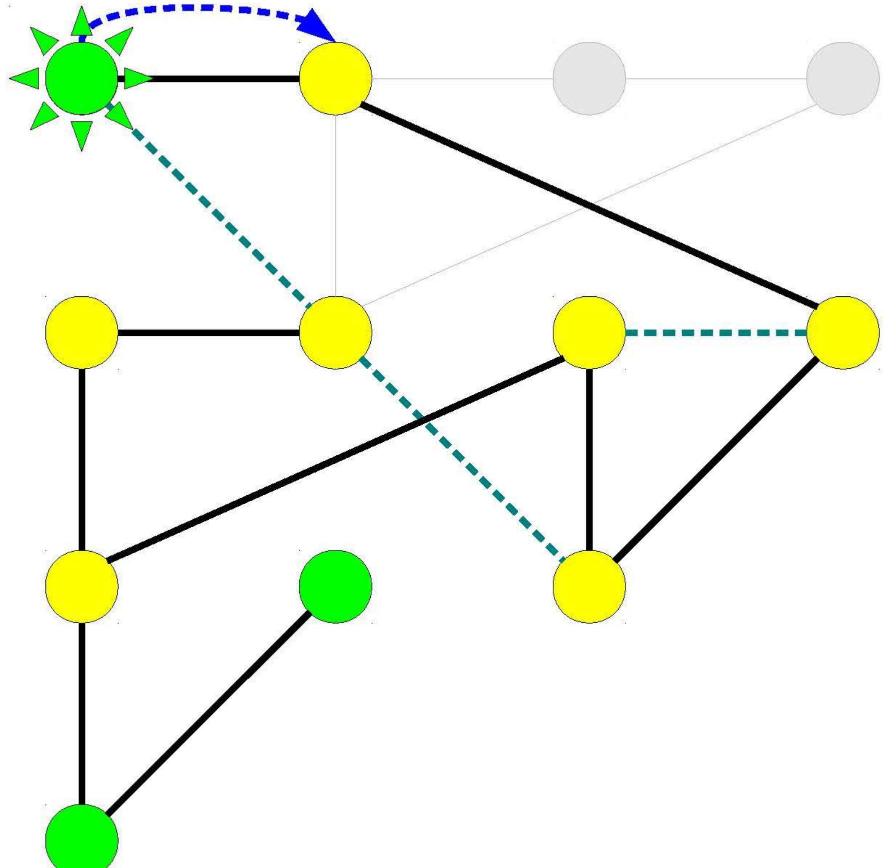
Depth-First Search



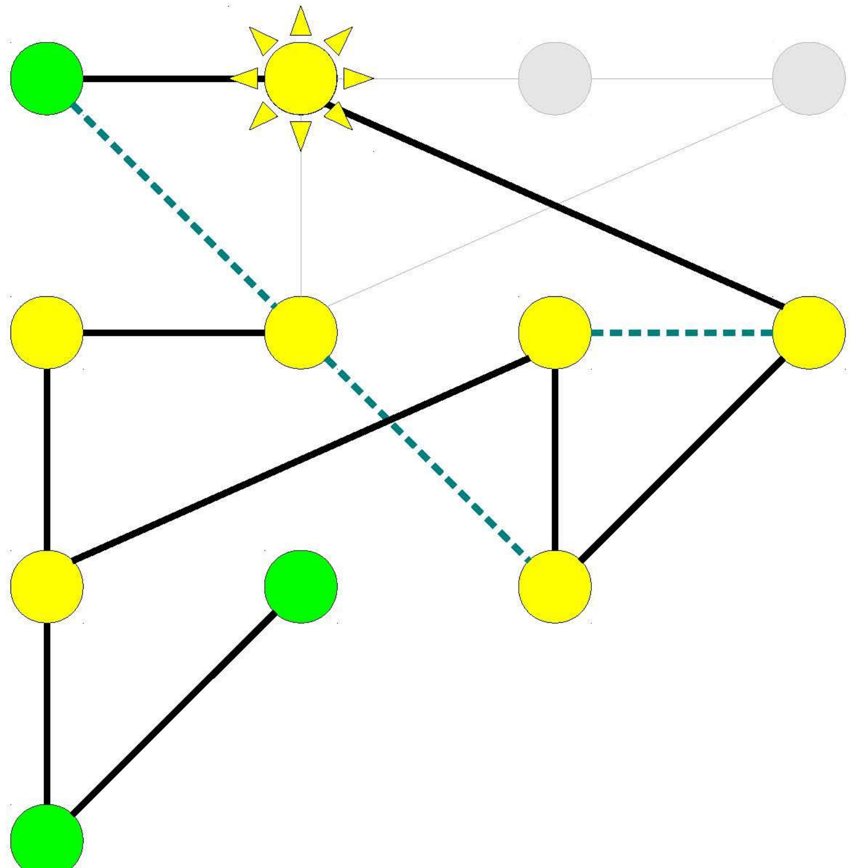
Depth-First Search



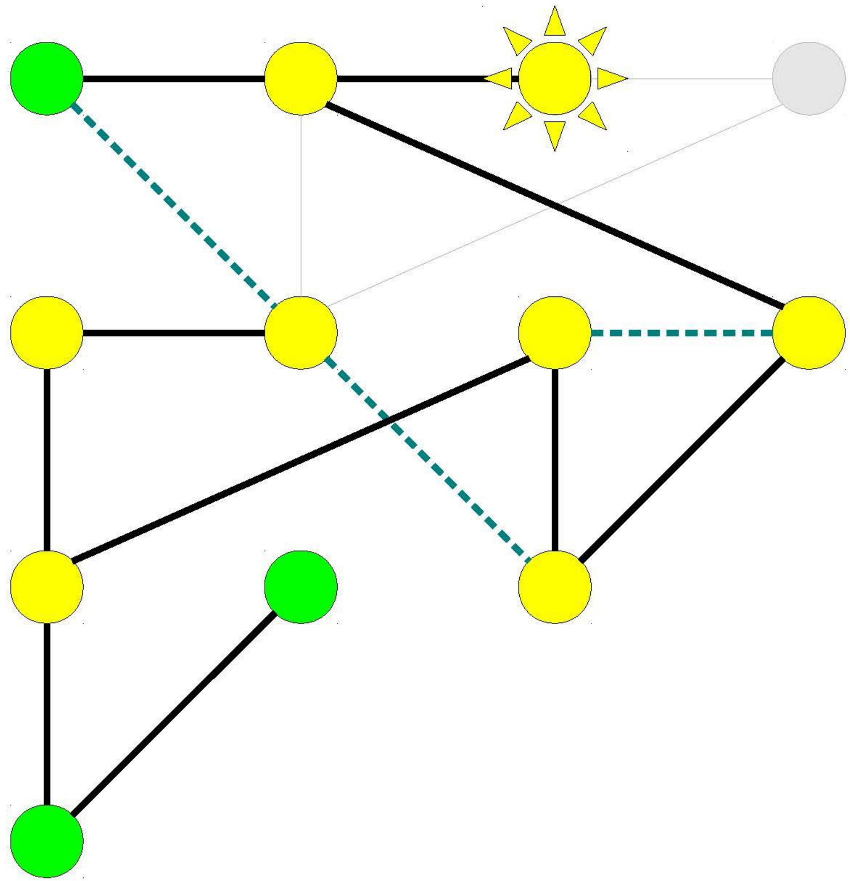
Depth-First Search



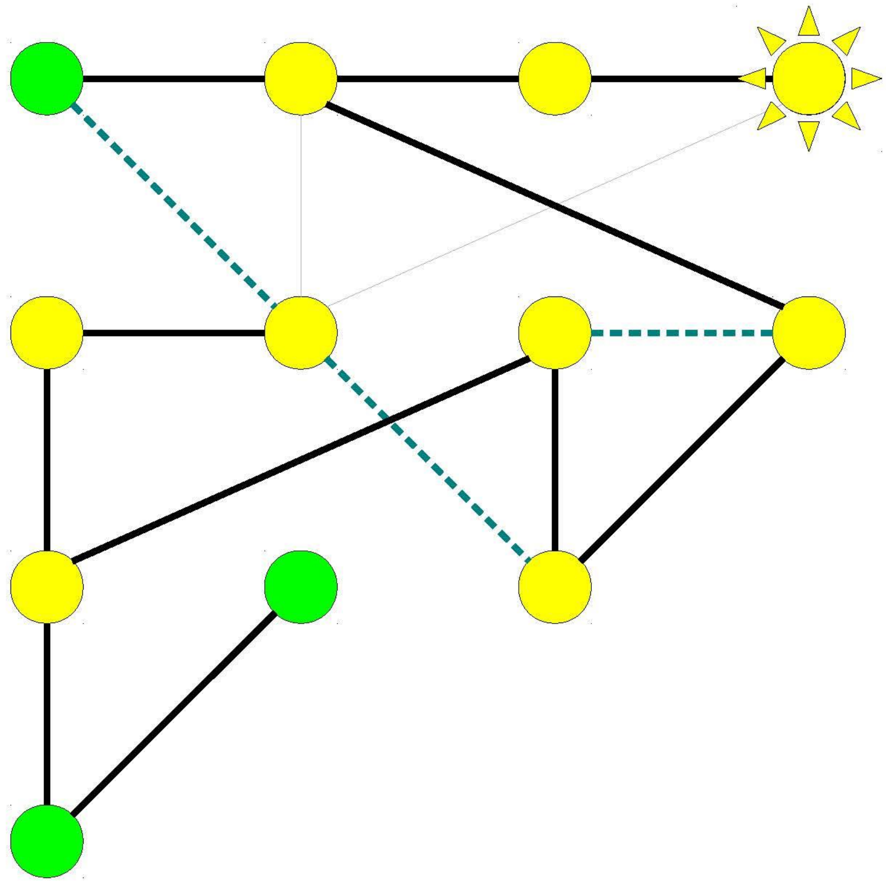
Depth-First Search



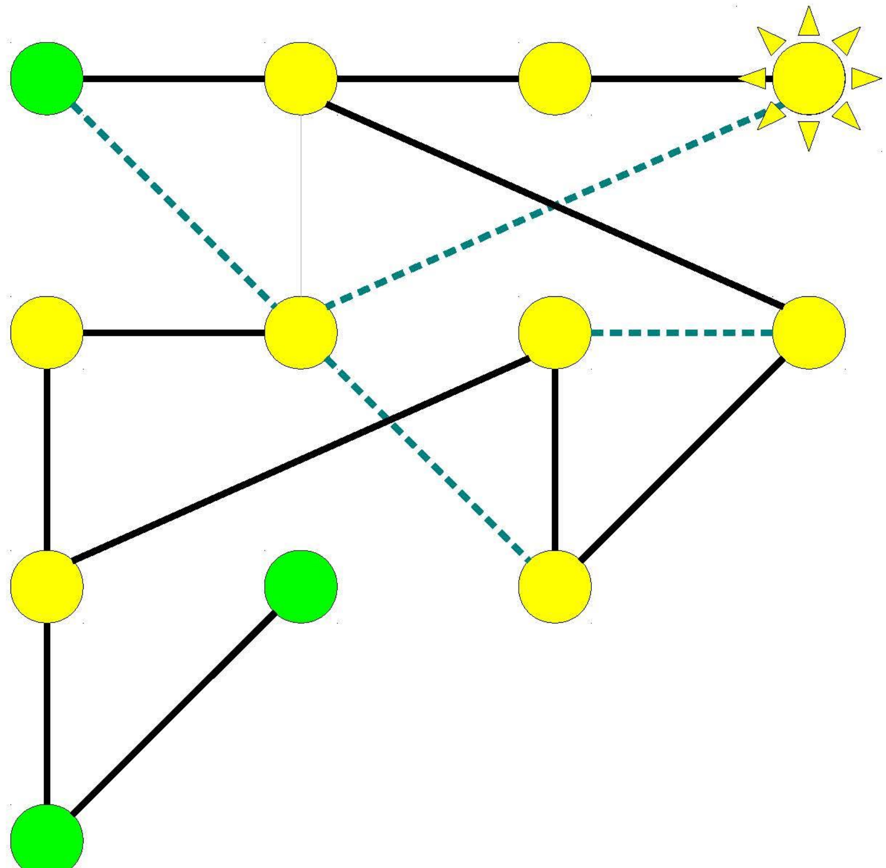
Depth-First Search



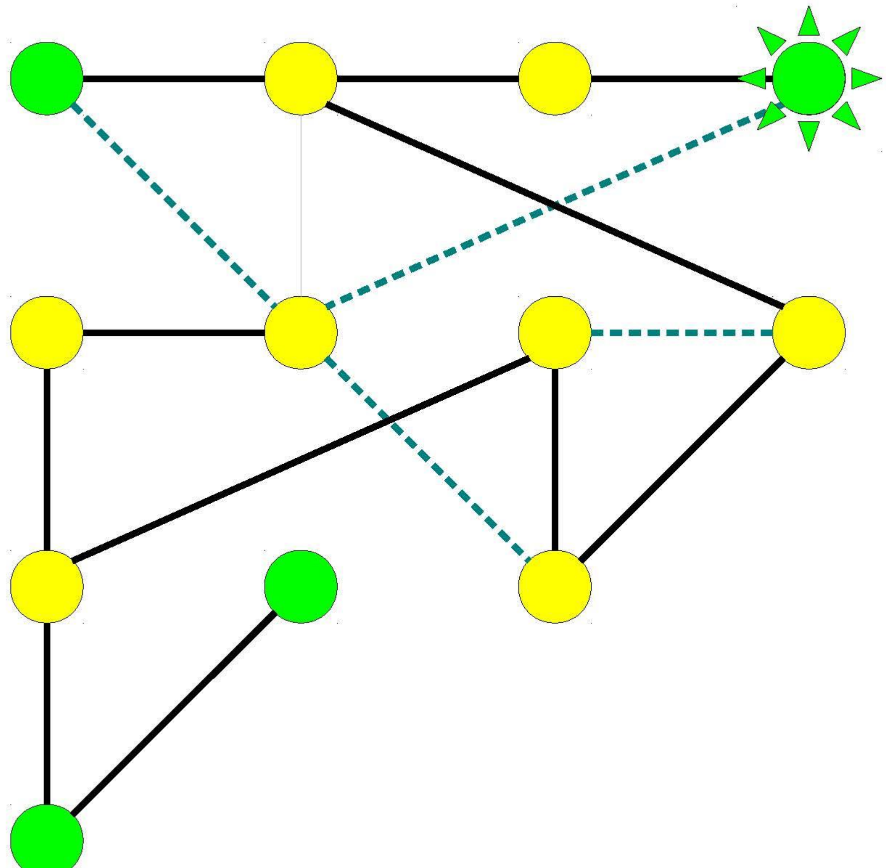
Depth-First Search



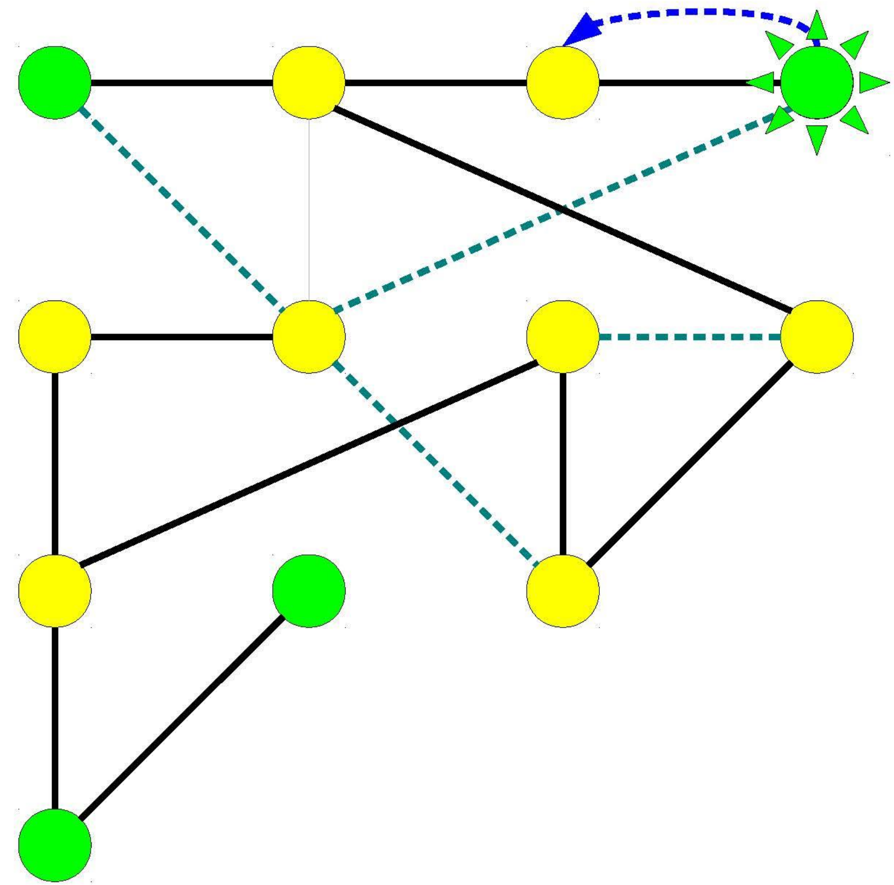
Depth-First Search



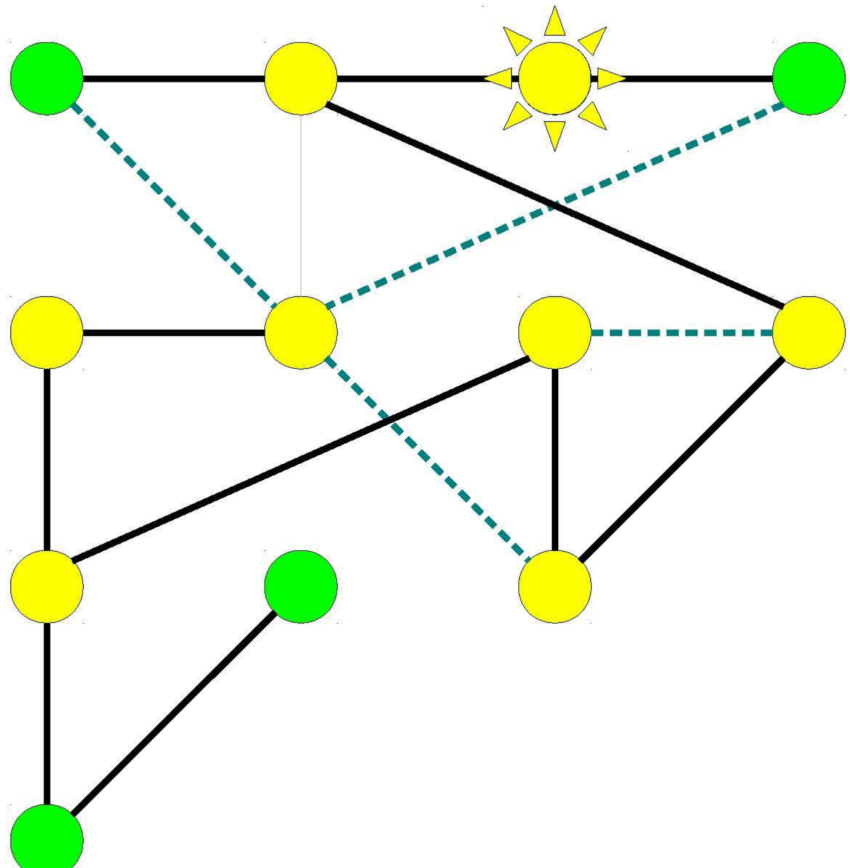
Depth-First Search



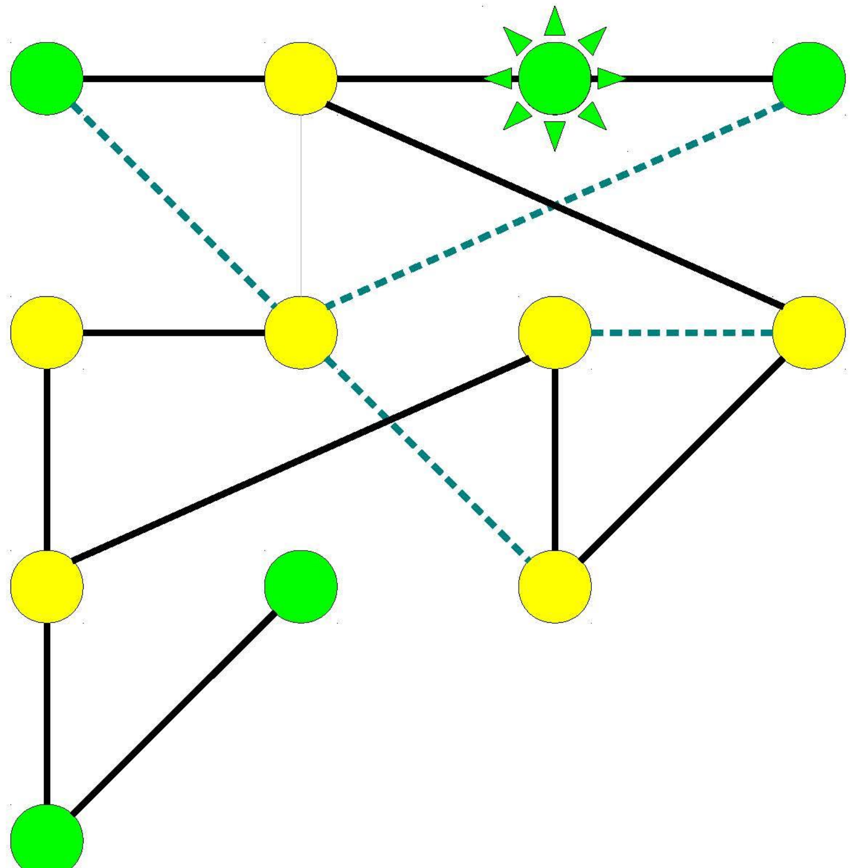
Depth-First Search



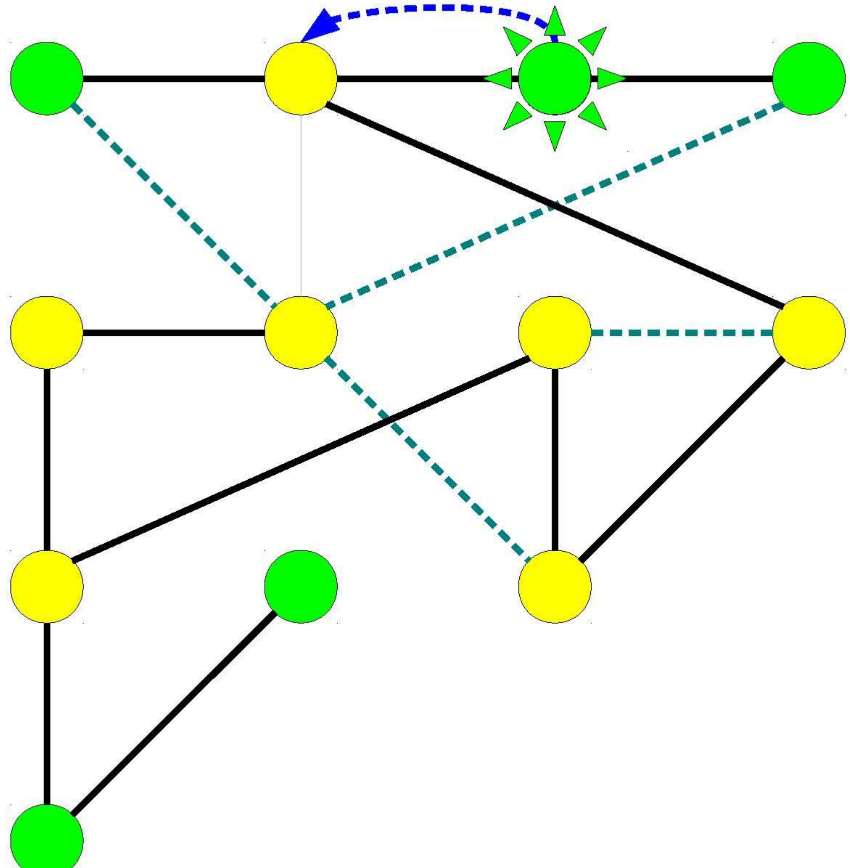
Depth-First Search



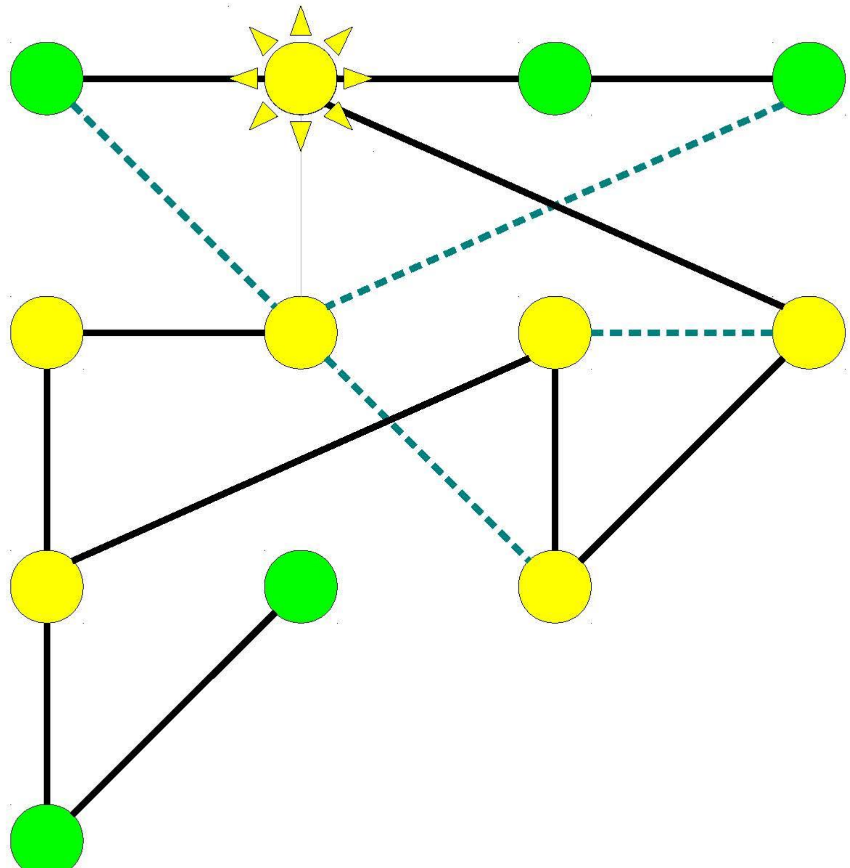
Depth-First Search



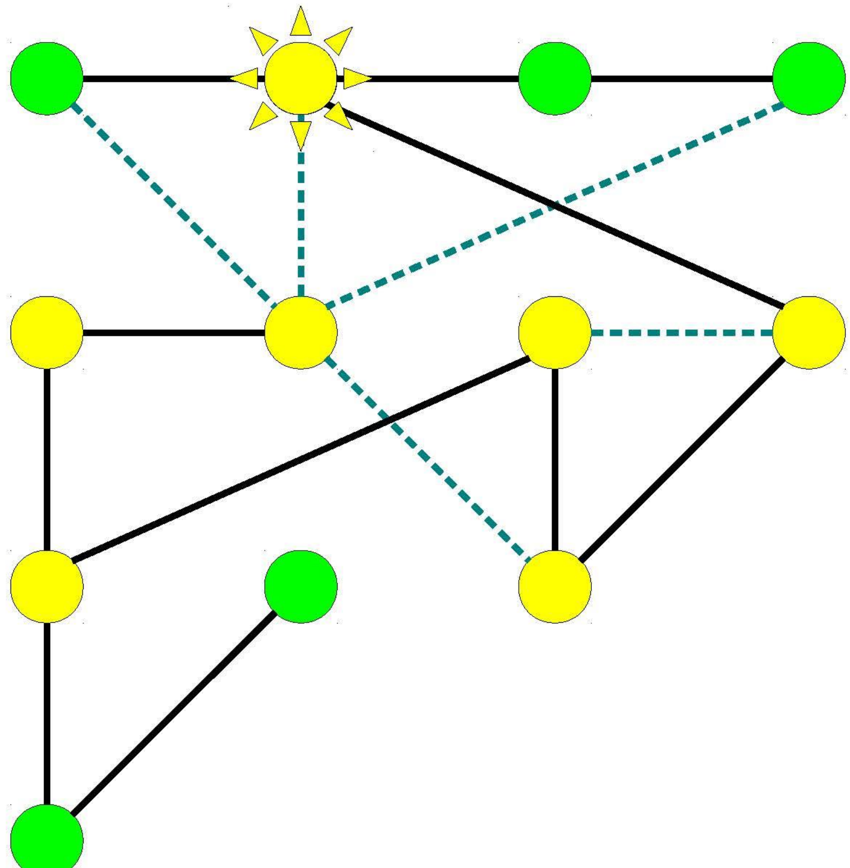
Depth-First Search



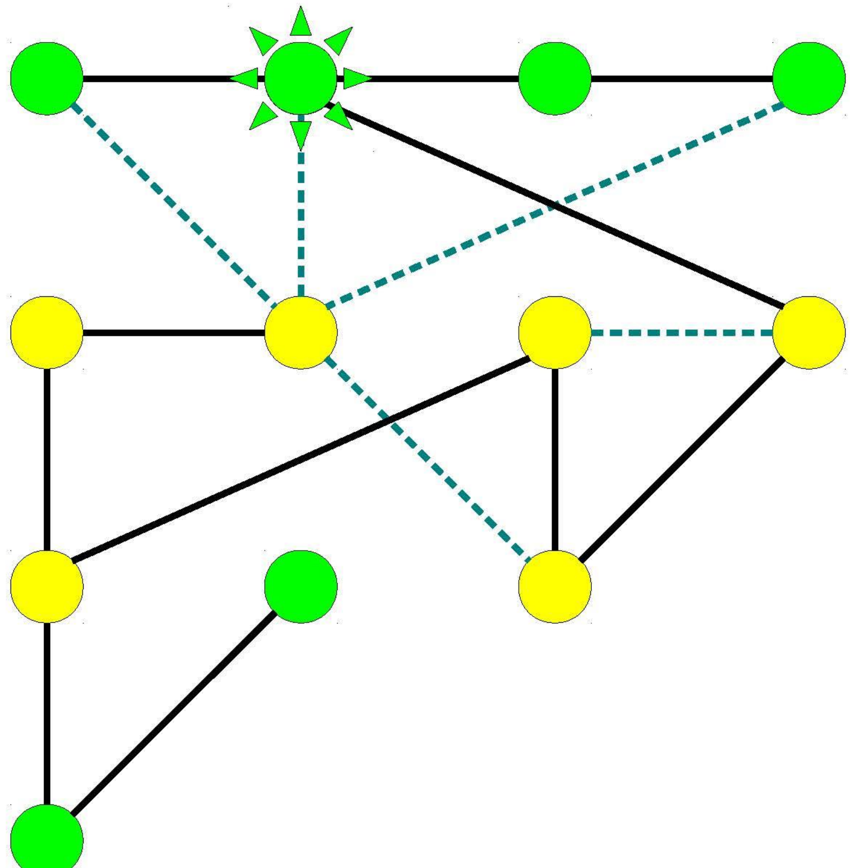
Depth-First Search



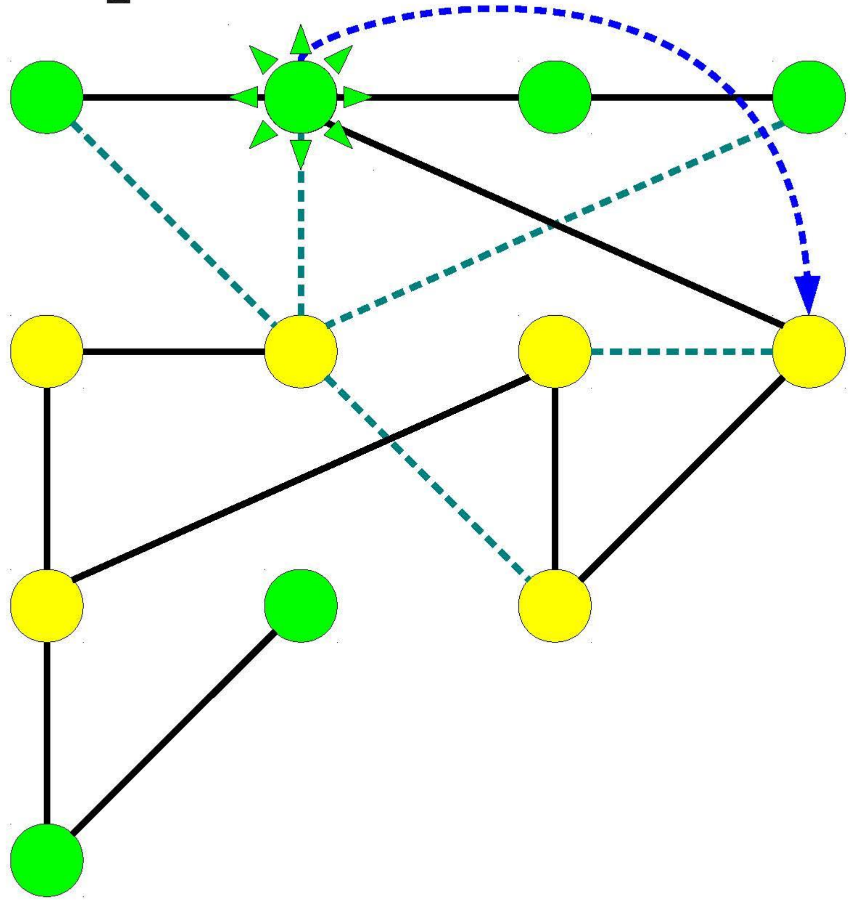
Depth-First Search



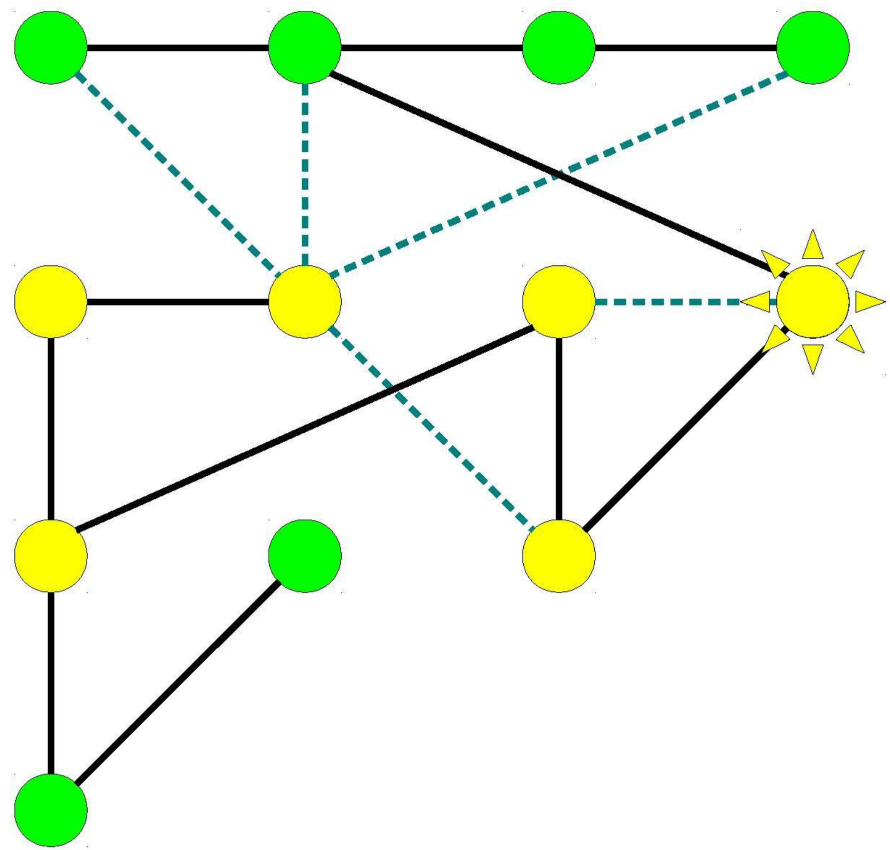
Depth-First Search



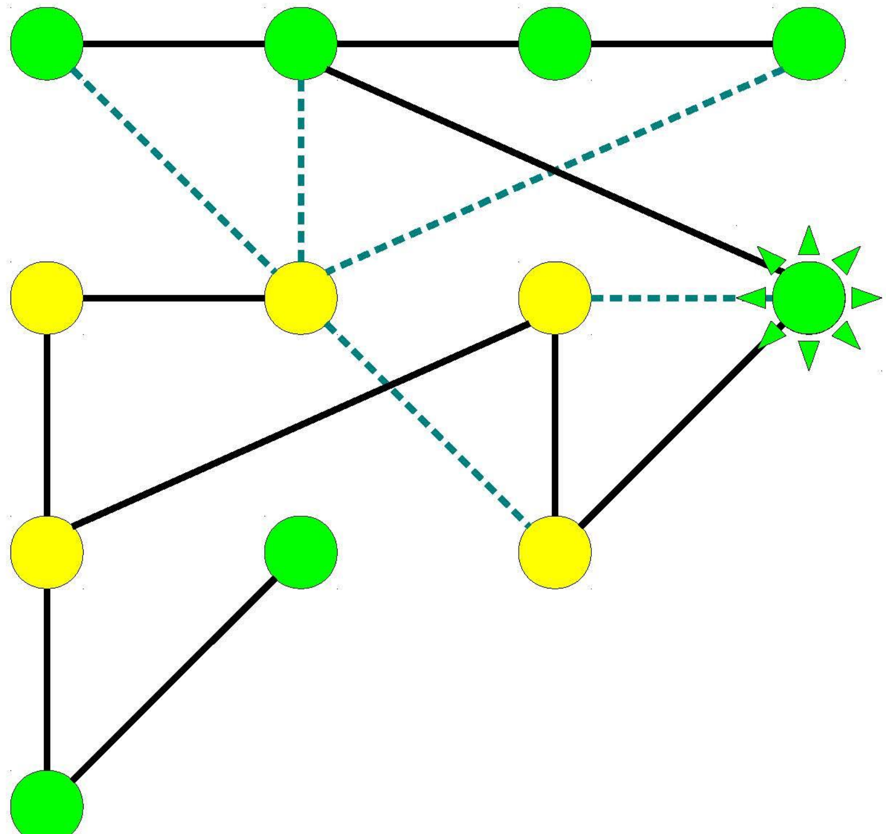
Depth-First Search



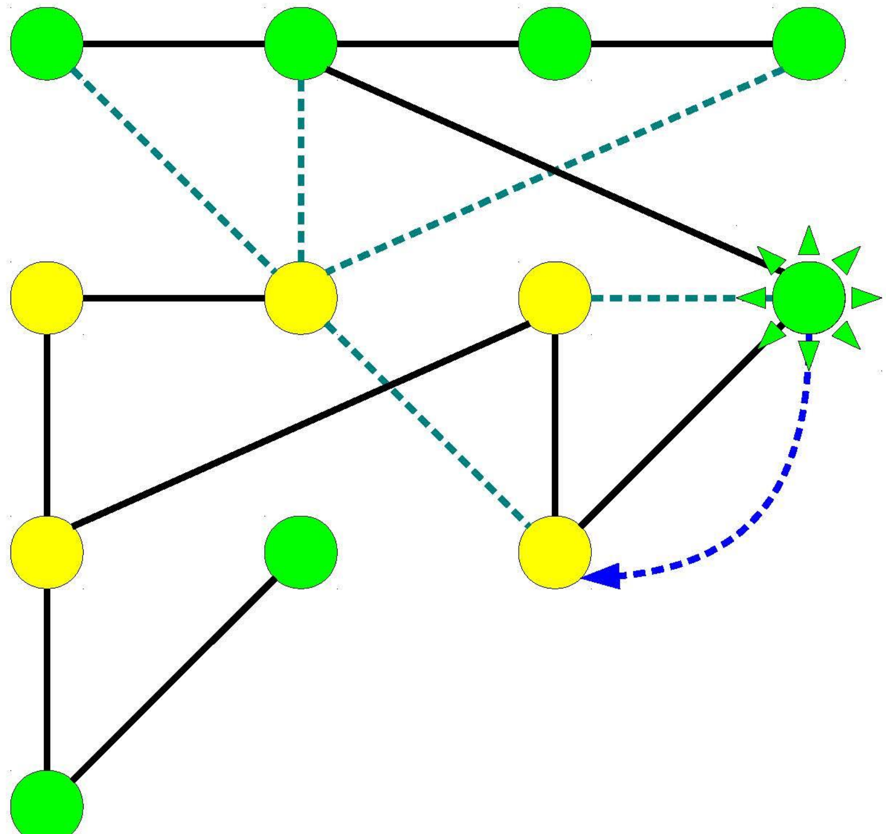
Depth-First Search



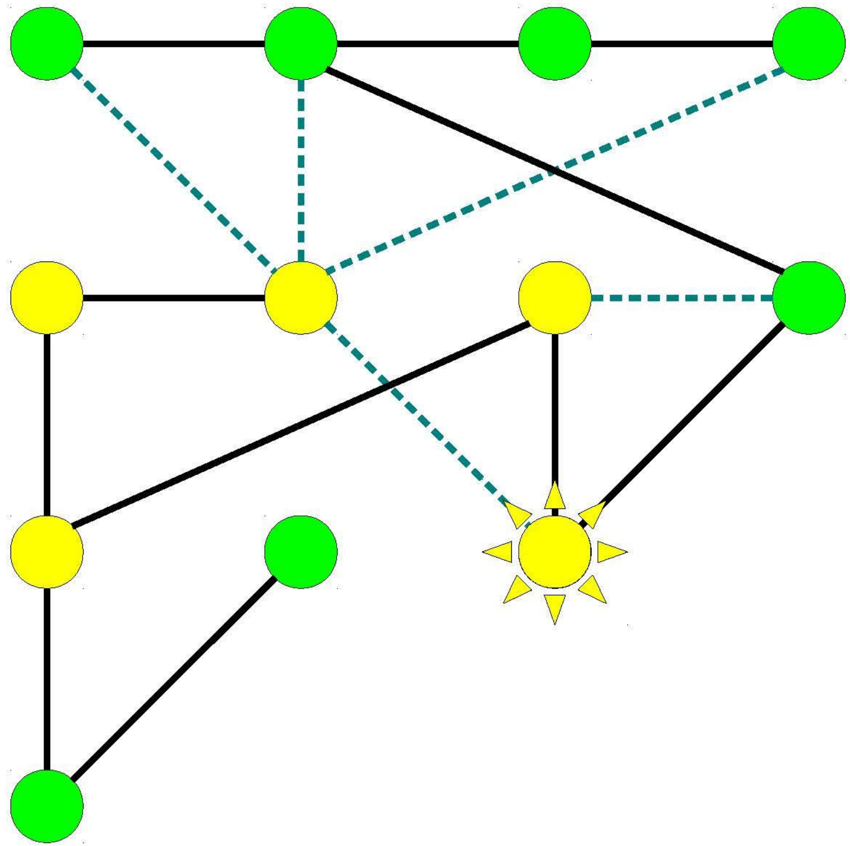
Depth-First Search



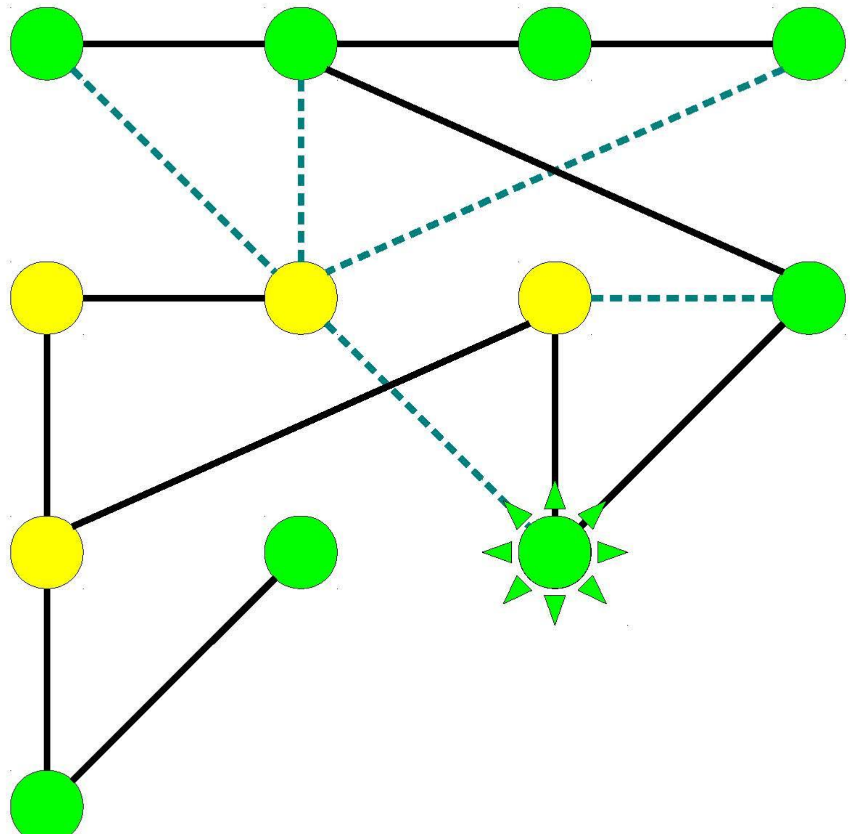
Depth-First Search



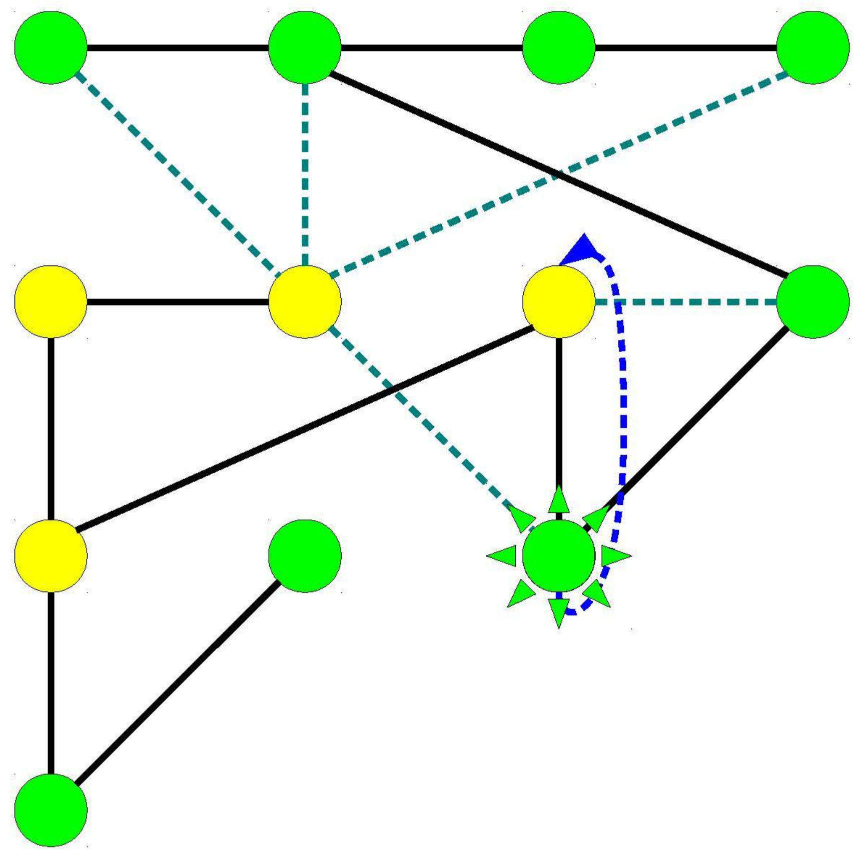
Depth-First Search



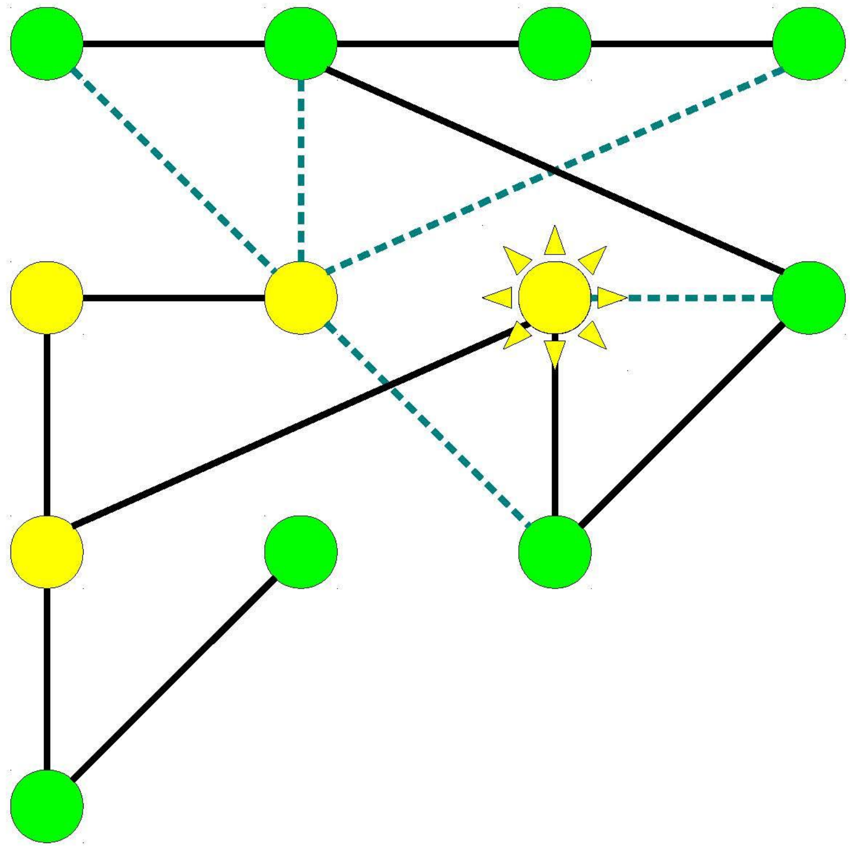
Depth-First Search



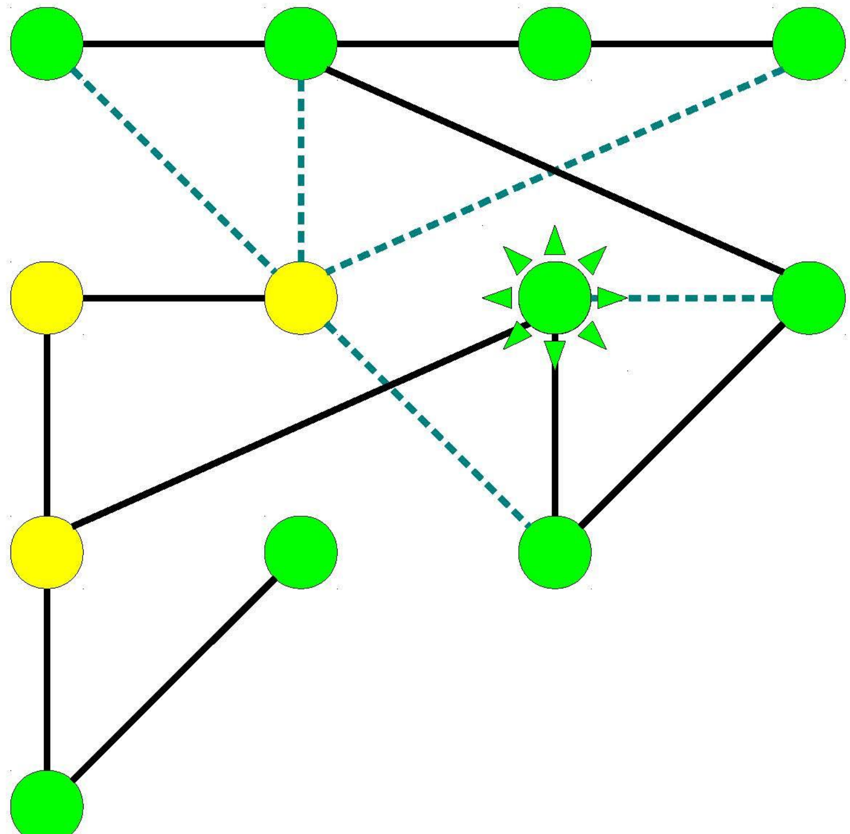
Depth-First Search



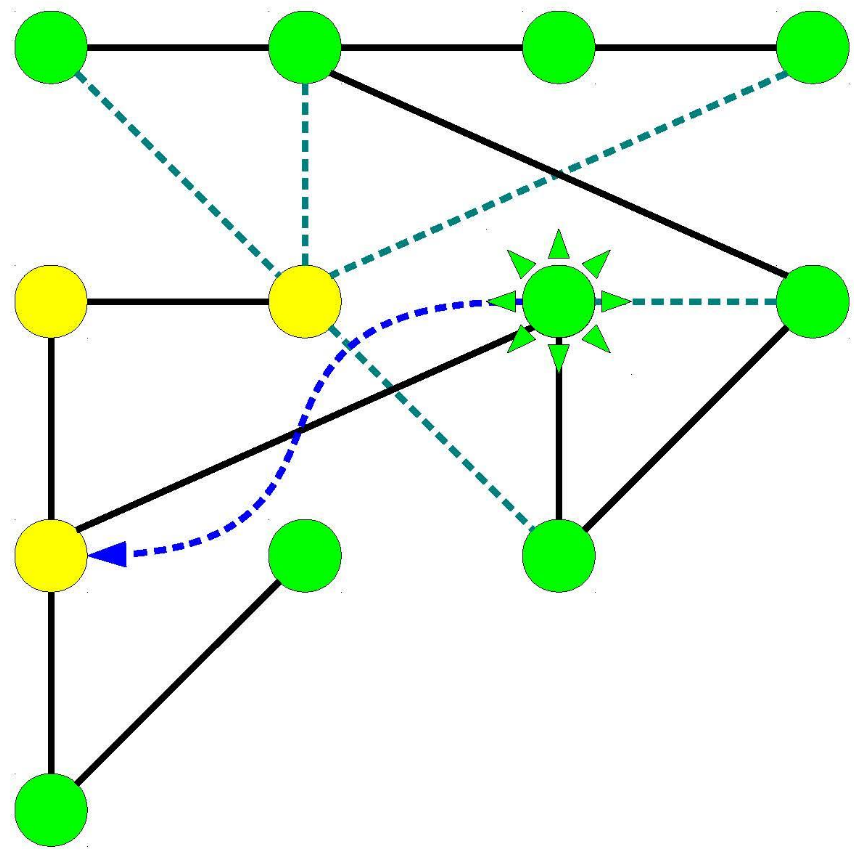
Depth-First Search



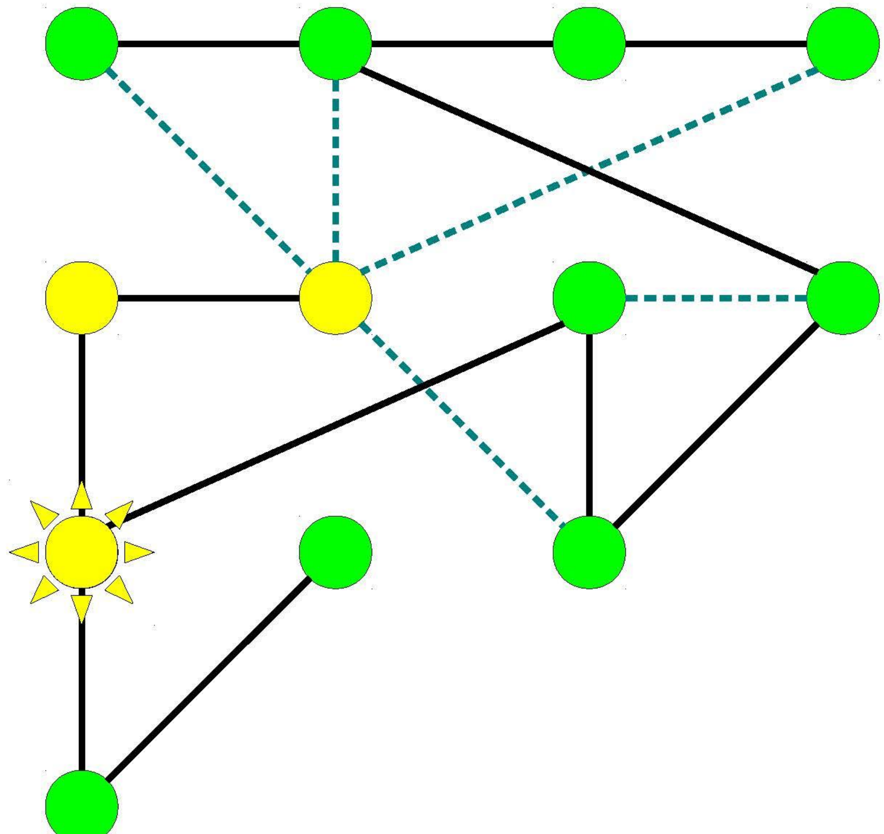
Depth-First Search



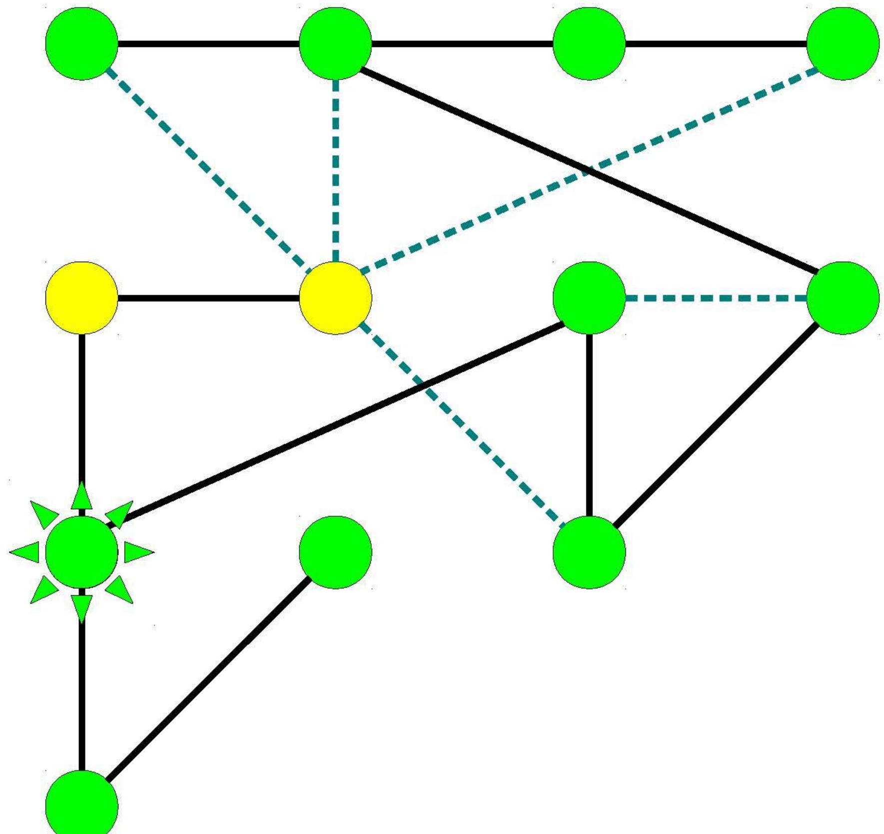
Depth-First Search



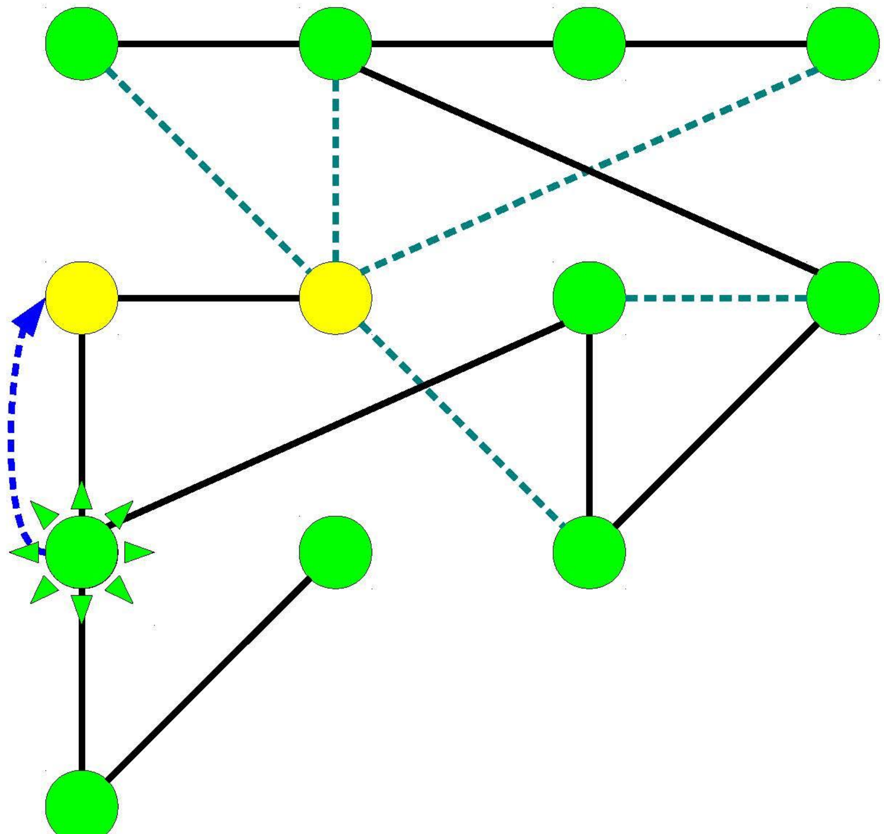
Depth-First Search



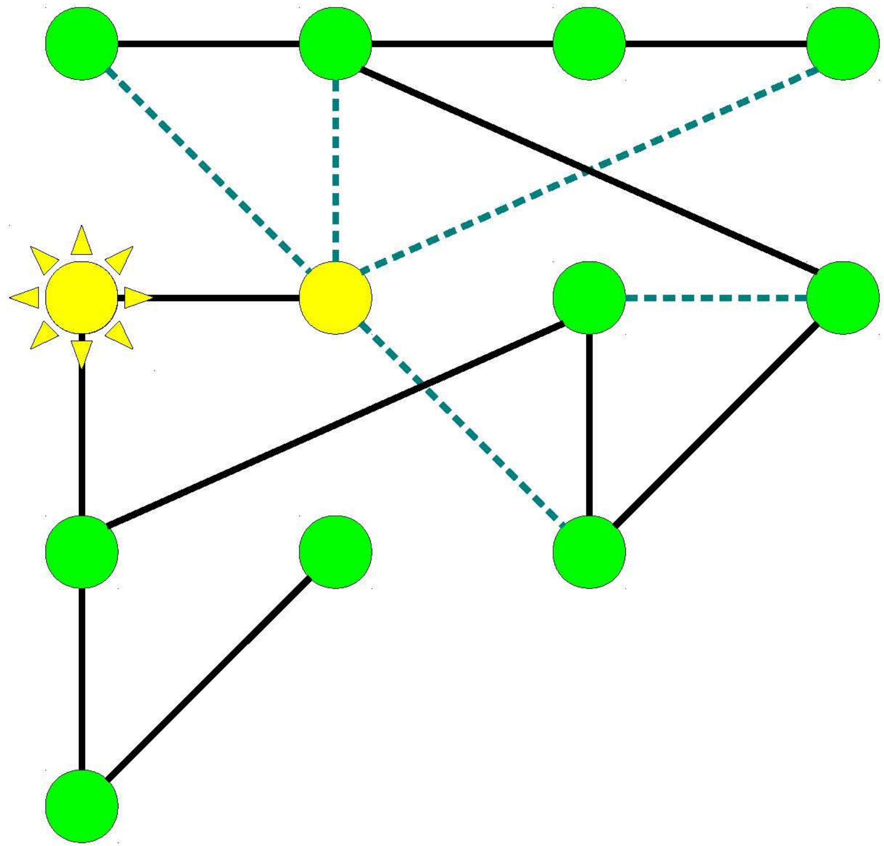
Depth-First Search



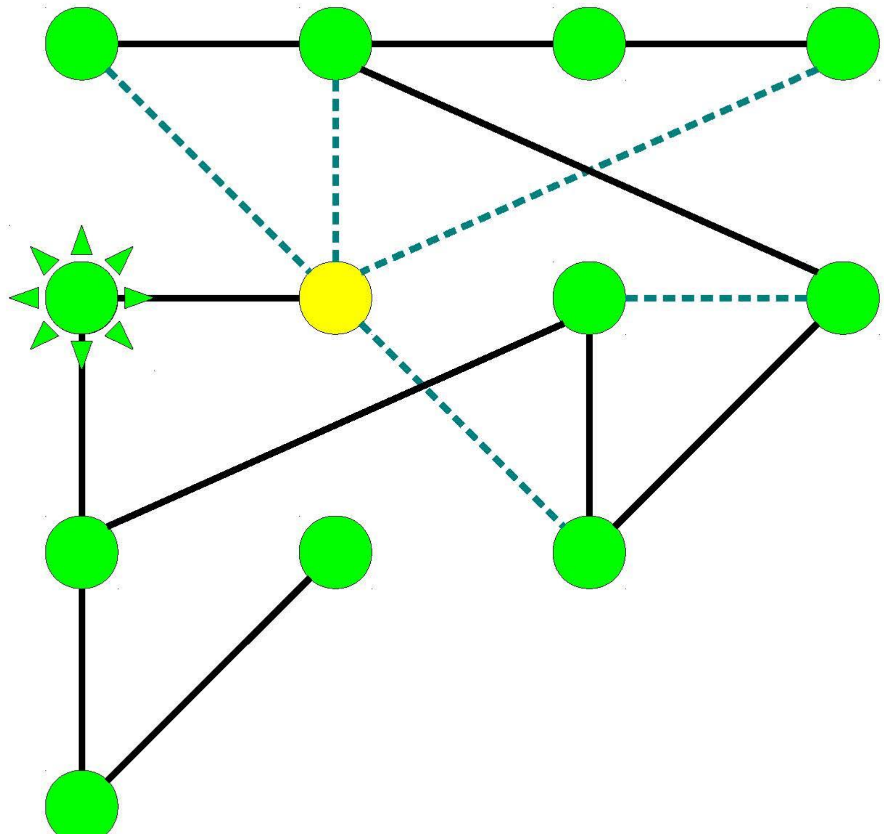
Depth-First Search



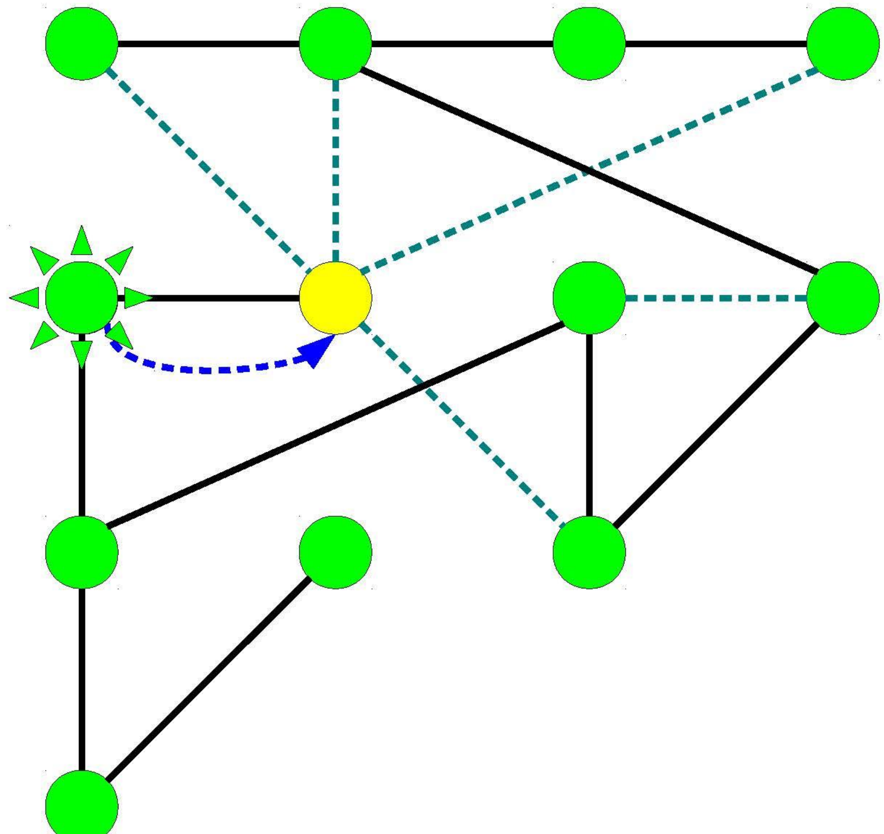
Depth-First Search



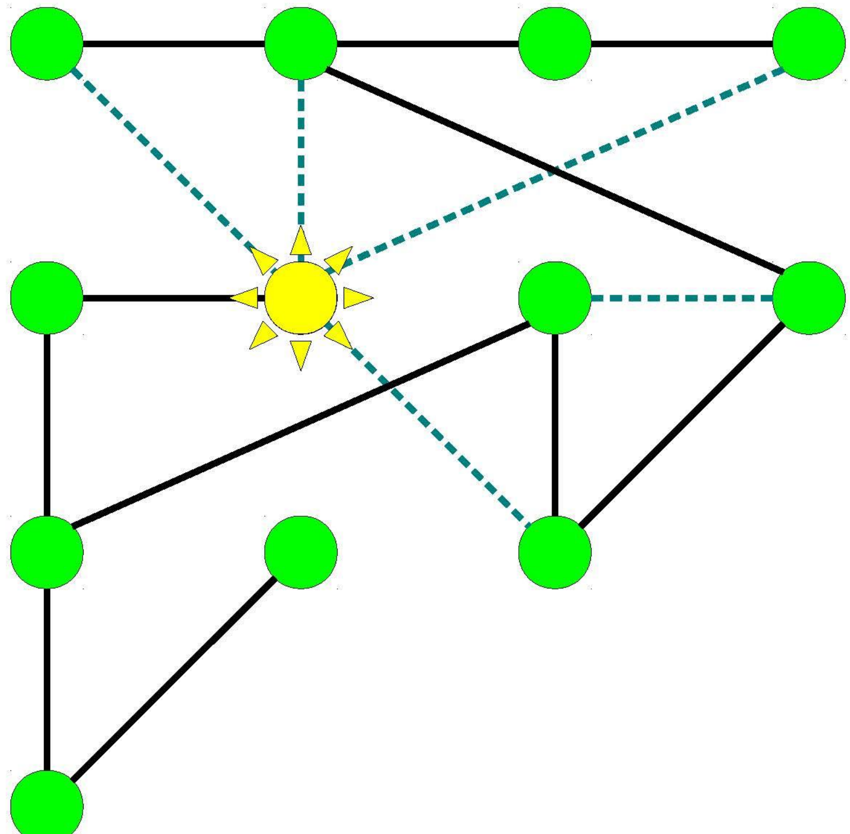
Depth-First Search



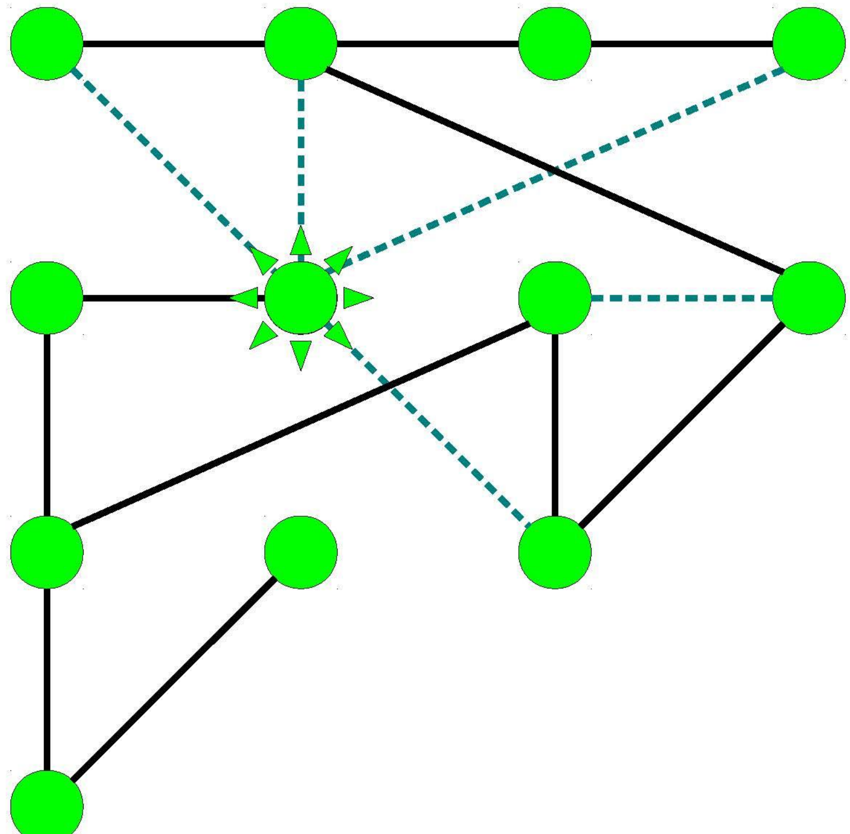
Depth-First Search



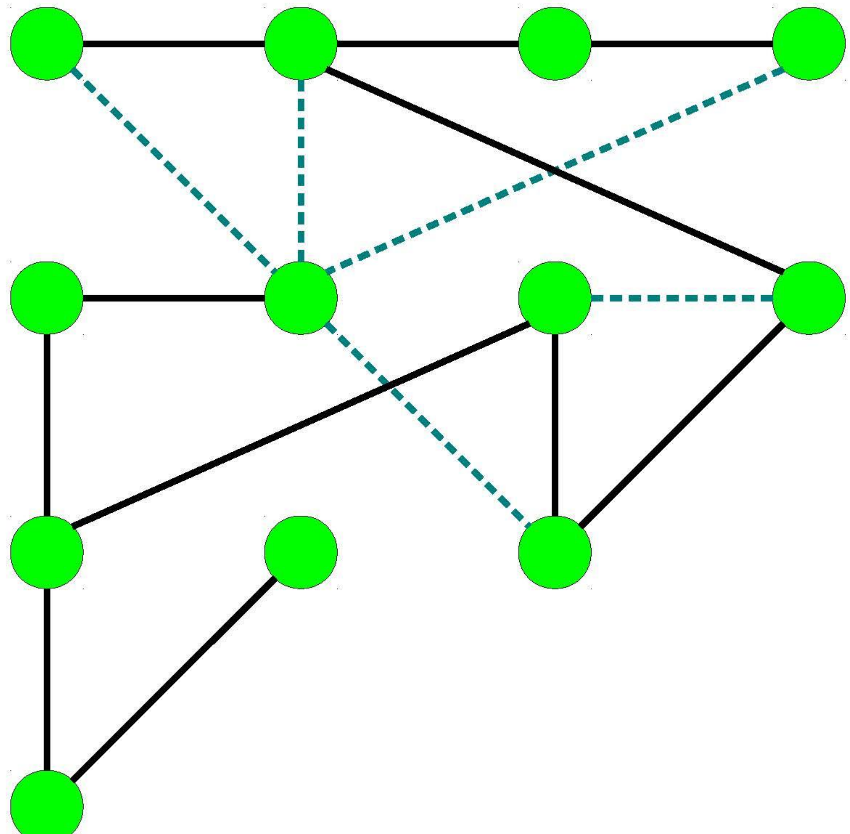
Depth-First Search



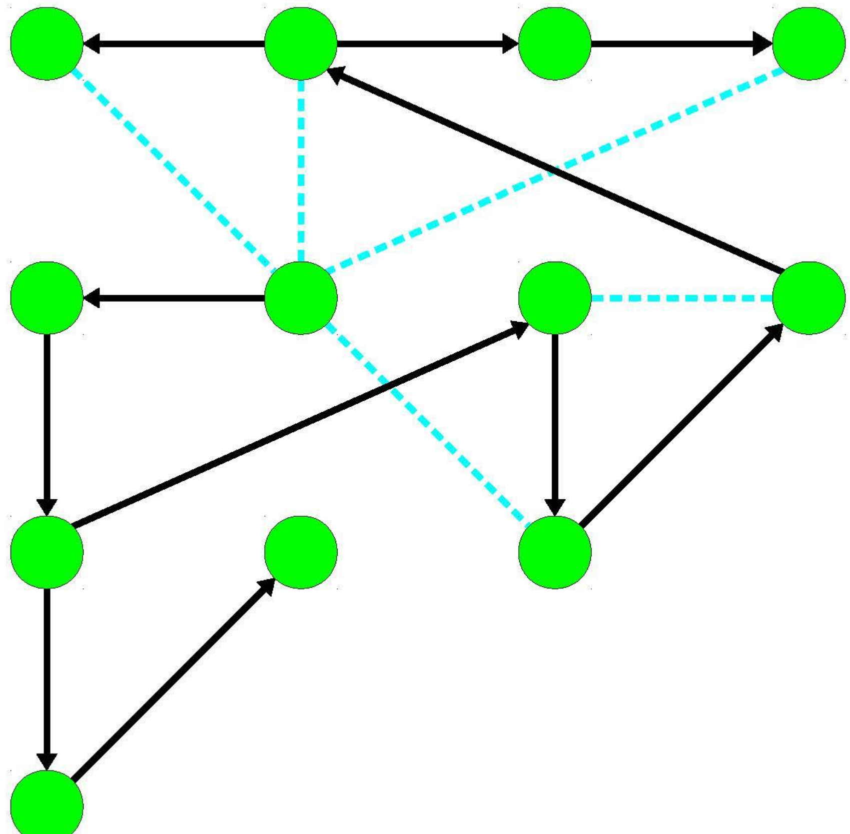
Depth-First Search



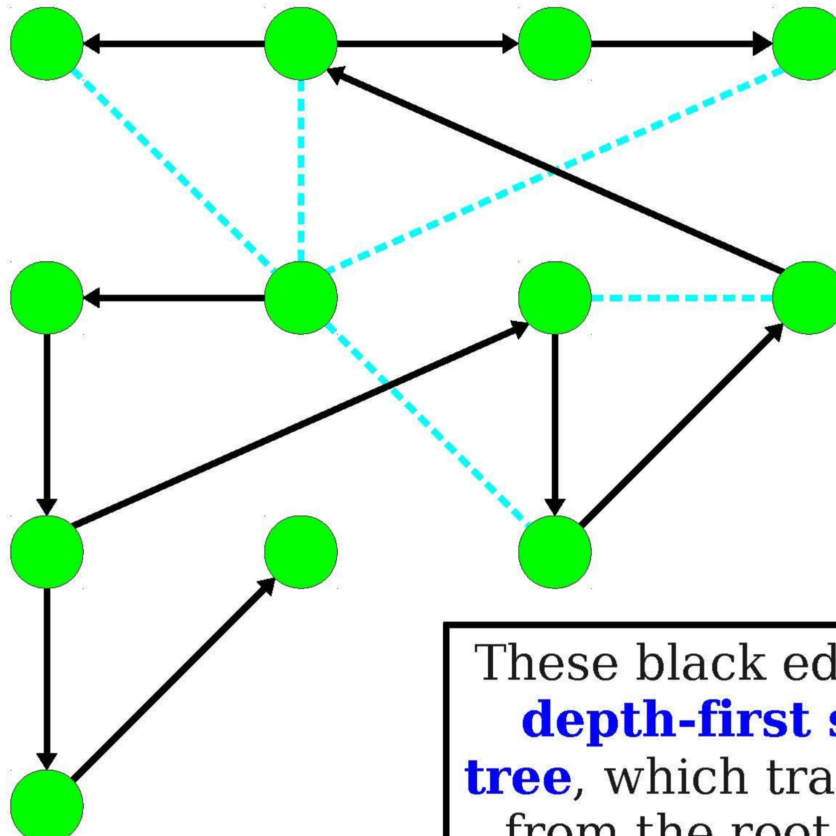
Depth-First Search



Depth-First Search



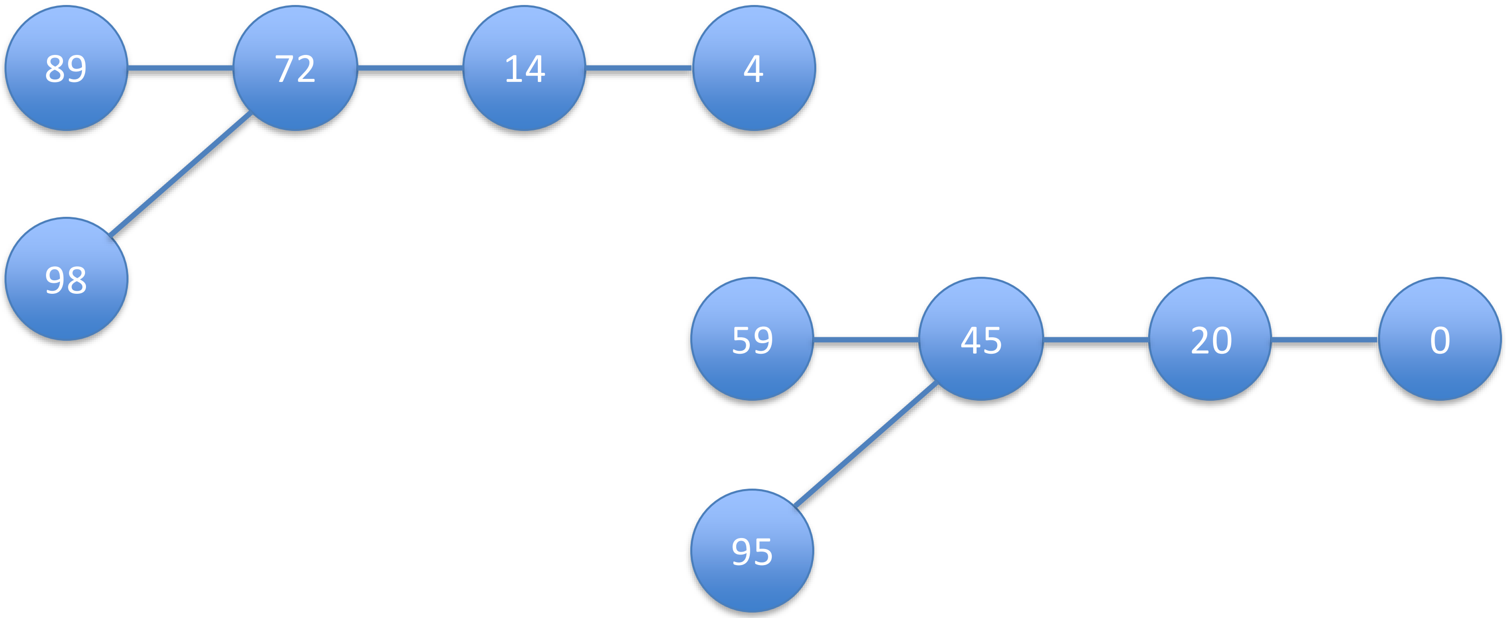
Depth-First Search



These black edges for a **depth-first search tree**, which traces paths from the root to each node in the graph.

An extra exercise

A network of numbers.



What numbers are reachable from 0?

Do you want to use BFS or DFS?