A more careful analysis

```
function Fib1(n)
if n = 1 return 1
if n = 2 return 1
return Fib1(n-1) + Fib1(n-2)
```

```
function Fib2(n)
Create an array fib[1..n]
fib[1] = 1
fib[2] = 1
for i = 3 to n:
    fib[i] = fib[i-1] + fib[i-2]
return fib[n]
```

Problem: we cannot count these additions as single operations!

How many bits does F_n have?

Addition of *n*-bit numbers takes $O(n)$ time.

Fib1: $O(n\ 2^{0.7n})$ time

Fib2: $O(n^2)$ time

Addition

Adding two *n*-bit numbers takes *O(n)* simple operations:

E.g. 22 + 13:

[22]	1	0	1	1	0
[13]		1	1	0	1

Big-O notation

```
function Fib2(n)
Create an array fib[1..n]
fib[1] = 1
fib[2] = 1
for i = 3 to n:
    fib[i] = fib[i-1] + fib[i-2]
return fib[n]
```

Running time is proportional to n^2.

But what is the constant: is it $2n^2$ or $3n^2$ or what?

The constant depends upon:

The units of time – minutes, seconds, milliseconds,…

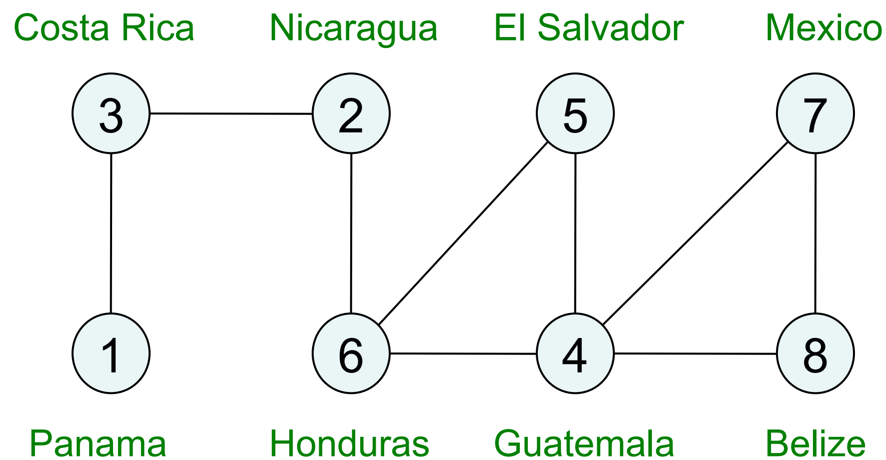
Specifics of the computer architecture.

It is *much* too hairy to figure out exactly. Moreover it is nowhere as important as the huge gulf between n^2 and 2^n.

So we simply say the running time is $O(n^2)$.

Why graphs?

A cartographer's problem



Graph specified by *nodes* and *edges*.

node	=	country
edge	=	neighbors

Graph coloring problem: color nodes of graph with as few colors as possible, so that there is no edge between nodes of the same color.

Exam scheduling

The registrar's problem

Pitchers (15)

#	Player	MWL Team	B	T	Ht.	Wt.	Born	Residence	'08 Stats (as of 6/3/08)
	Bryan Augenstein	South Bend	R	R	6-6	232	7/11/1986	Sebastian, FL	3-1, 2.09 ERA, 73.1 IP, 9BB
	Randy Boone	Lansing	R	R	6-3	215	8/6/1984	Yoakum, TX	6-2, 2.56 ERA, 59.2 IP, 13BB
	Mark Diapoules	Quad Cities	R	R	6-2	200	5/31/1988	Palm City, FL	4-0, 1.83 ERA, .226 BAA
	Edgar Estanga	Lansing	L	L	5-10	185	10/18/1985	Maturin Managas, VZ	5-1, 0.66 ERA, .182 BAA
	Alfredo Figaro	West Michigan	R	R	6-0	173	7/7/1984	Samanna, DR	7-2, 1.22 ERA, .176 BAA
	Jeff Jeffords	Dayton	R	R	6-1	200	11/4/1984	Lamar, SC	2-1, 2.83 ERA, 28.2 IP, 37K
	Jon Kibler	West Michigan	L	L	6-4	215	8/10/1986	Freeland, MD	5-2, 2.20 ERA, .159 BAA
	Steven Johnson	Great Lakes	R	R	6-1	212	8/31/1987	Kingsville, MD	6-2, 2.60 ERA, .214 BAA
	Joseph Krebs	Dayton	L	L	6-0	200	9/14/1984	Bridgeport, TX	5-2, 2.43 ERA, 4 saves
	Derek McDaid	Fort Wayne	R	R	6-1	200	9/27/1983	Barrie, ON, Canada	3-0, 2.52 ERA, .225 BAA
	Brad Mills	Lansing	L	L	6-0	185	3/5/1985	Mesa, AZ	4-2, 2.62 ERA, 58.1IP, 68K
	Luis Montano	Dayton	R	R	6-0	180	3/20/1985	Santo Domingo, DR	6-3, 4.45 ERA, 56.2 IP, 13BB
	Jarrod Parker	South Bend	R	R	6-1	190	11/24/1988	Ossian, IN	4-2, 2.45 ERA, .236 BAA
	Miguel Ramirez	Great Lakes	R	R	5-11	165	7/15/1983	Fondo Negro, DR	1-3, 0.37 ERA, 12 saves
	Evan Scribner	South Bend	R	R	6-3	190	7/19/1985	New Britain, CT	2-3, 2.05 ERA, 26.1IP, 40K

Catchers (2)

#	Player	MWL Team	B	T	Ht.	Wt.	Born	Residence	'08 Stats (as of 6/3/08)
	Sean Coughlin^	South Bend	L	R	6-1	206	5/14/1985	Morrison, CO	.265, 6 HR, 24 RBI, .541SLG
	Kenley Jansen	Great Lakes	B	R	6-4	225	9/30/1987	Curacao, Netherlands Antilles	.204, 5 HR, 11 RBI

Infielders (11)

#	Player	MWL Team	B	T	Ht.	Wt.	Born	Residence	'08 Stats (as of 6/3/08)
	Kevin Ahrens	Lansing	B	R	6-1	190	4/26/1989	Houston, TX	.260, HR, 24 RBI, 14 doubles
	Chris Carlson^	West Michigan	R	R	6-4	225	1/7/1984	Topeka, KS	.291, 7 HR, 32 RBI
	Felix Carrasco	Fort Wayne	B	R	6-1	244	2/14/1987	Bani, DR	.255, 9 HR, 37 RBI
	Justin Jackson	Lansing	R	R	6-2	175	12/11/1988	Asheville, NC	.250, 3 HR, 23 RBI, 40 R
	Pete Kozma^	Quad Cities	R	R	6-0	170	4/11/1988	Owasso, OK	.274, 3 HR, 19 RBI
	Mike Mee	South Bend	L	R	6-0	188	10/14/1983	Richfield, MN	.260, 2 HR, 21 RBI, .379 OBP
	Andy Parrino^	Fort Wayne	B	R	6-0	177	10/31/1985	Brockport, NY	.276, 3 HR, 15 RBI, .387 OBP
	Manny Rodriguez^	Lansing	L	L	6-3	190	1/6/1985	Chitre, Panama	.310, 4 HR, 37 RBI,19 doubles
	John Tolisano	Lansing	B	R	5-11	180	10/7/1988	Estero, FL	.274, HR, 23 RBI, 6 triples
	Joe Tucker	West Michigan	R	R	5-11	170	1/25/1984	Canton, OH	.273, 15 RBI, 13 K in 132 AB
	Brandon Waring^	Dayton	R	R	6-4	195	1/2/1986	West Columbia, SC	.267, 11 HR, 32 RBI, .497 SLG

Outfielders (5)

#	Player	MWL Team	B	T	Ht.	Wt.	Born	Residence	'08 Stats (as of 6/3/08)
	Evan Frey^	South Bend	L	L	5-11	171	6/7/1986	Edwardsville, IL	.333, 22 RBI, 13 SB, .384 OBP
	Charlie Kingrey^	Quad Cities	L	L	6-2	210	1/19/1985	Kinder, LA	.308, 5 HR, 32 RBI,15 doubles
	Andrew Lambo^	Great Lakes	L	L	6-3	200	8/11/1988	Newbury Park, CA	.267, 7 HR, 41 RBI,14 doubles
	Denis Phipps	Dayton	R	R	6-2	176	7/22/1985	San Pedro de Macoris, DR	.261, 4 HR, 29 RBI,12 doubles
	Casper Wells	West Michigan	R	R	6-2	210	11/23/1984	Schenectady, NY	.237, 10 HR, 26 RBI, 15 SB

Schedule final exams:

- use as few time slots as possible

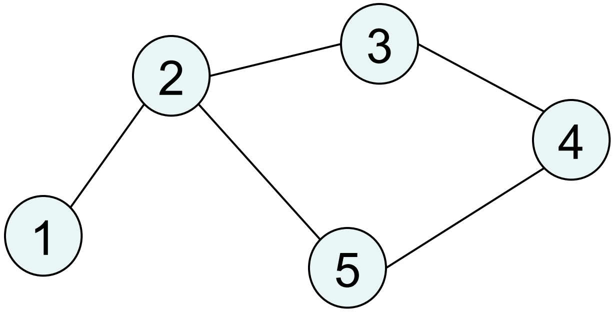
- can't schedule two exams in the same slot if there's a student taking both classes.

This is also graph coloring!

Node = exam

Edge = some student is taking both endpoint-exams

Color = time slot



Animal crossing

Animals need to be ferried across a river

- Use as few boats as possible
- Cannot put two animals in the same boat if one will eat the other

This is, yet again, graph coloring!

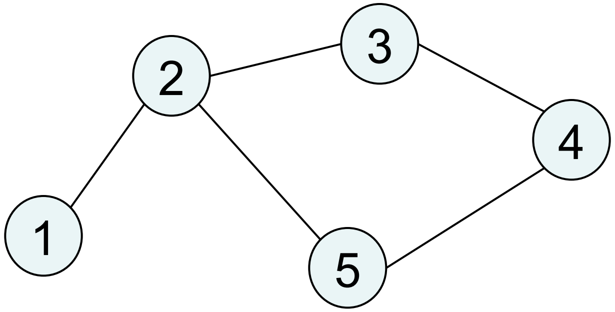
Node = animal
Edge = one endpoint-animal will eat the other
Color = boat

Graph representations

G = (V,E) where

V: vertices/nodes

E: edges



V = {1,2,3,4,5}

E = {{1,2}, {2,3}, {3,4}, {2,5}, {4,5}}

Undirected edges: symmetric relationship

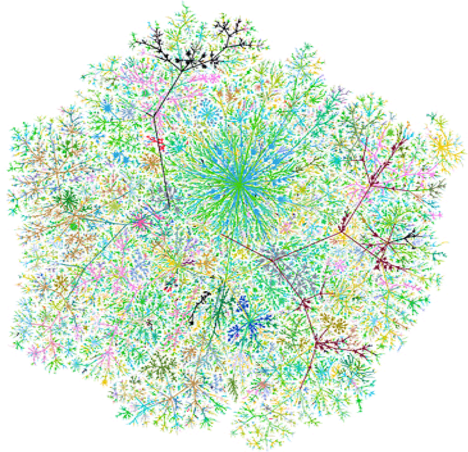
Directed graphs

(x,y): edge *from* x *to* y

e.g.World wide web

node	URL
edge (u,v)	u points to v

Billions of nodes and edges!



How are graphs stored on a computer?

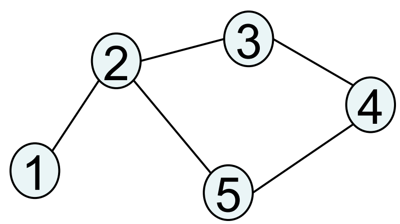
Adjacency matrix

V x V matrix A

A(i,j) = 1 if (i,j) is in E
0 otherwise

Symmetric if G undirected

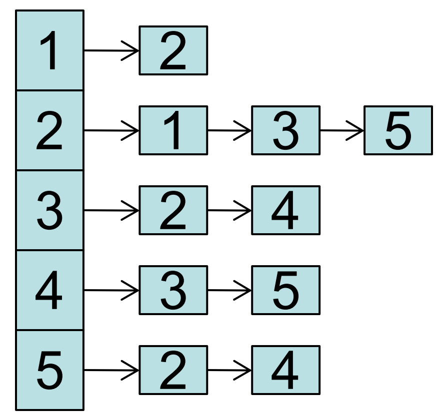
$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$



PRO check for an edge in O(1) time
CON uses up $O(V^2)$ space

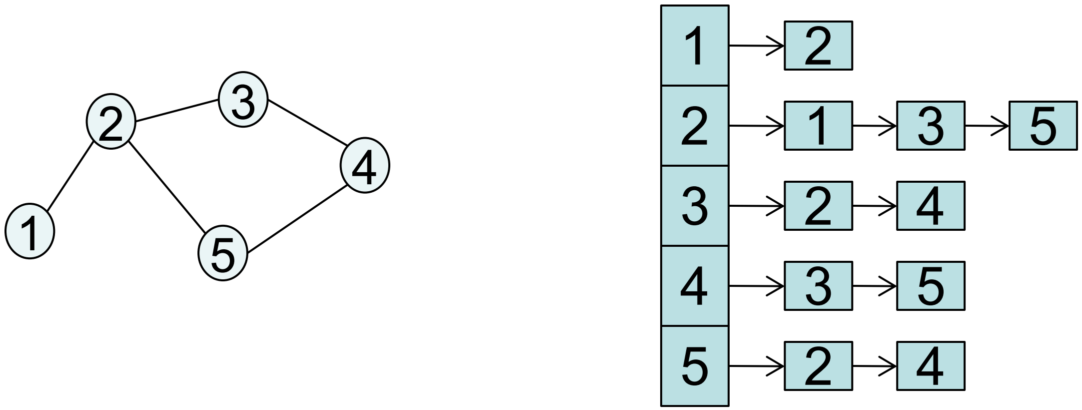
Adjacency list

For each node, list of outgoing edges

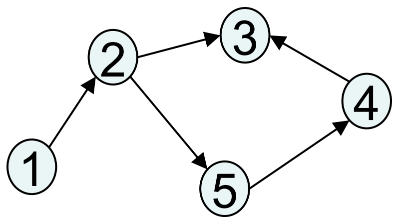


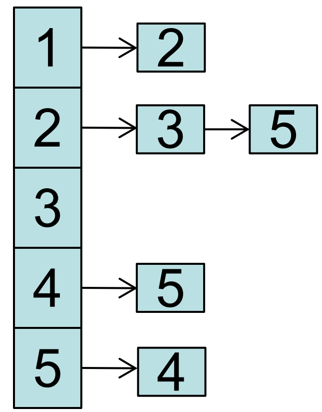
PRO just O(V + E) space
CON check for an edge in O(V) time
PRO easily iterate through node's neighbors

Undirected graphs: adjacency list



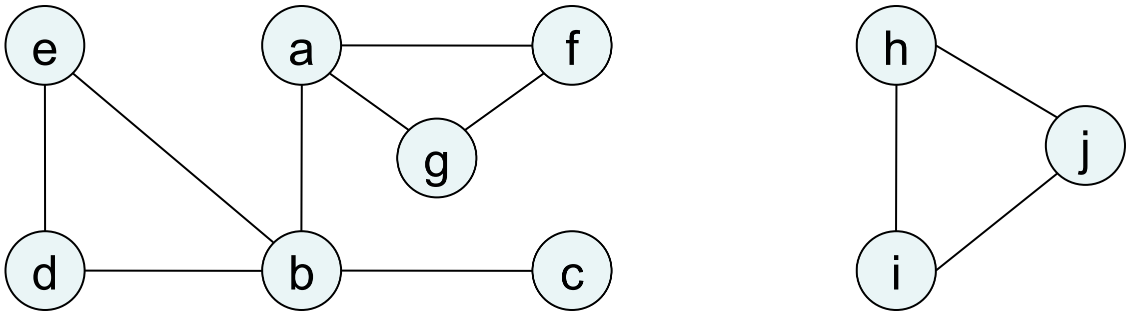
Directed graphs: adjacency list





Reachability in undirected graphs

What parts of a graph are reachable from a given vertex?



With an adjacency list representation, this is like navigating a maze...

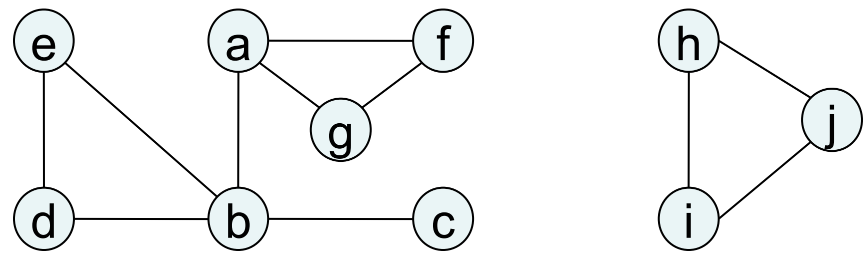
Potential difficulty	**Don't go round in circles**	**Don't miss anything**
Classical solution	Piece of chalk to mark visited junctions	Ball of string – leads back to starting point
Cyber-analog	Boolean variable for each vertex: visited or not	STACK

An exploration procedure

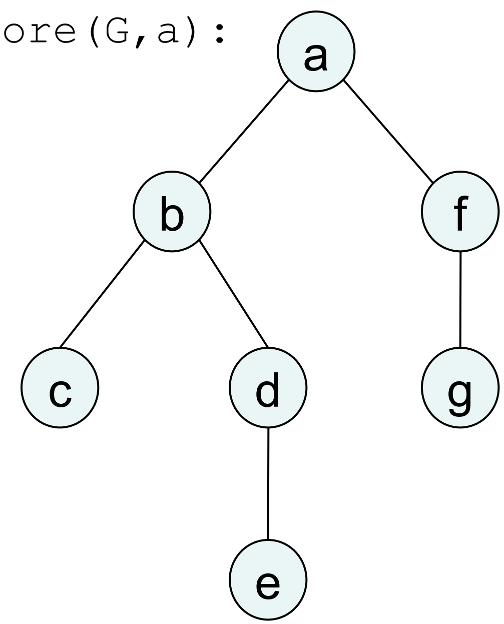
```
procedure explore(G,v)

input: graph G = (V,E); node v in V
output: visited[u] is set to true
   for all u reachable from v

visited[v] = true
for each edge (v,u) in E:
   if not visited[u]:
       explore(G,u)
```



explore(G,a):



Does "explore" work?

```
procedure explore(G,v)
visited[v] = true
for each edge (v,u) in E:
   if not visited[u]:
       explore(G,u)
```

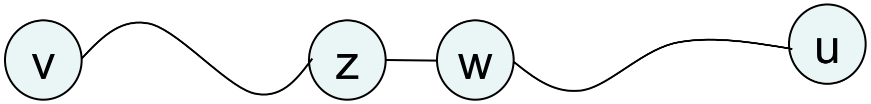
Does it actually halt?

For any node u, explore(G,u) is called at most once; thereafter visited[u] is set.

Does it visit everything reachable from v?

Suppose it misses node u reachable from v; we'll derive a contradiction.

Pick any path from v to u, and let z be the last node on the path that was visited.



But w would not have been overlooked during explore(G,z); this is a contradiction.

Alternative proof

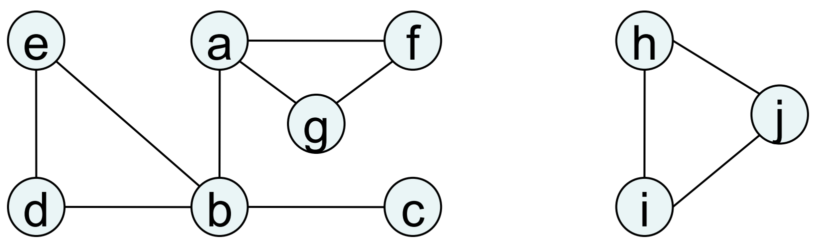
```
procedure explore(G,v)
visited[v] = true
for each edge (v,u) in E:
   if not visited[u]:
        explore(G,u)
```

Does explore(G,v) visit everything reachable from v?

Do a proof by induction.

Undirected connectivity

An undirected graph is *connected* if there is a path between any pair of nodes.



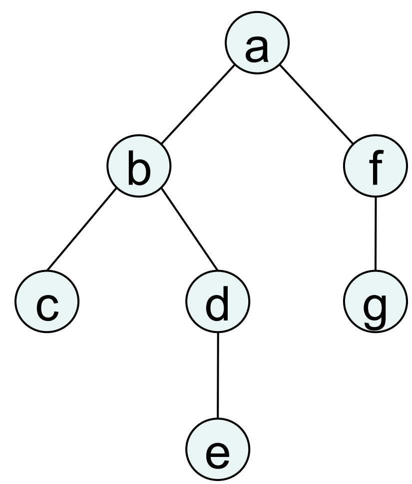
This graph has 2 *connected components*.

explore(G,v) returns the connected component containing v.
To explore the rest of the graph, restart explore() elsewhere.

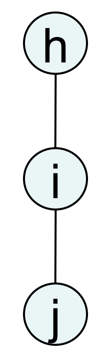
DFS decomposes an undirected graph into its connected components!

```
procedure dfs(G)
for all v in V:
   visited[v] = false
for all v in V:
   if not visited[v]:
        explore(G,v)
```

explore(G,a)



explore(G,h)



Running time analysis

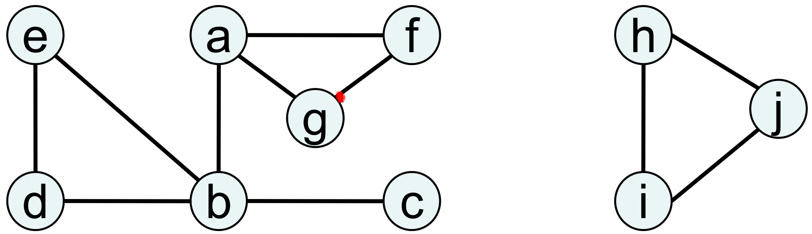
```
procedure explore(G,v)
visited[v] = true
for each edge (v,u) in E:
   if not visited[u]:
        explore(G,u)
```

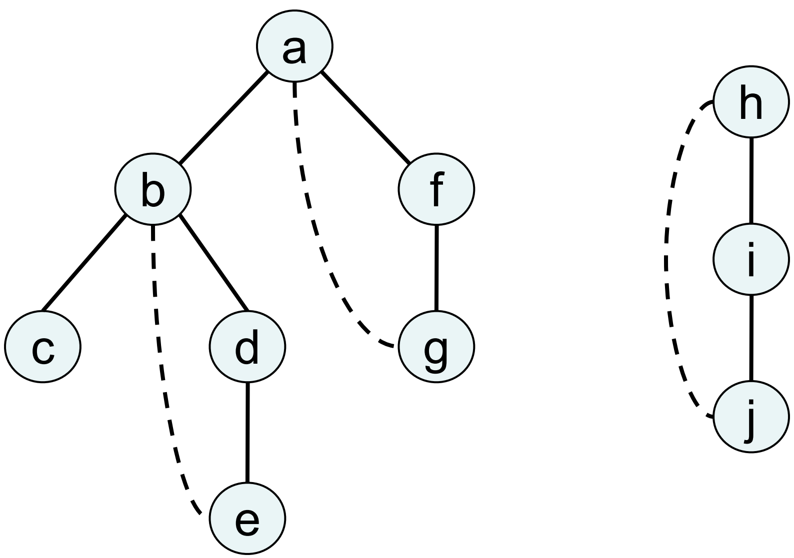
```
procedure dfs(G)
for all v in V:
   visited[v] = false
for all v in V:
   if not visited[v]:
        explore(G,v)
```

How long does dfs(G) take?

explore(G,v) is called exactly once for each node v.

DFS search forest





Terminology:
DFS search forest consisting of two *DFS search trees*

tree edge: traversed by DFS

back edge: not traversed (led to a node already visited)