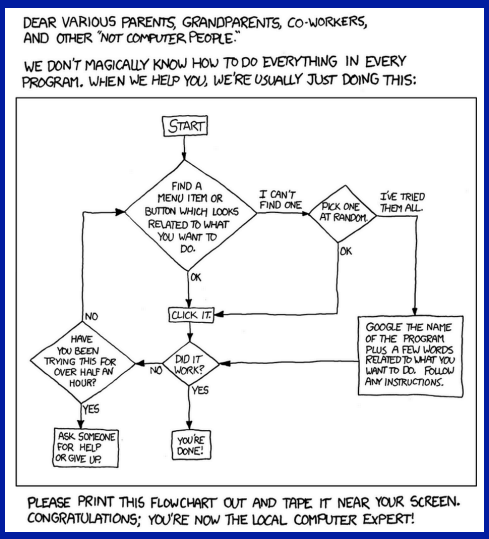
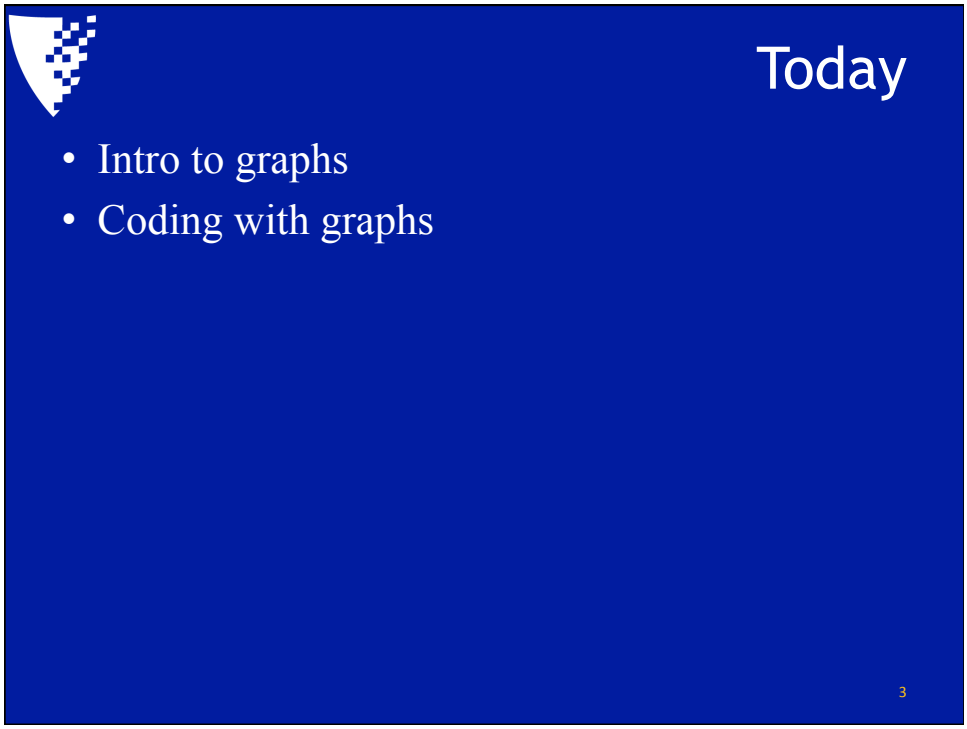


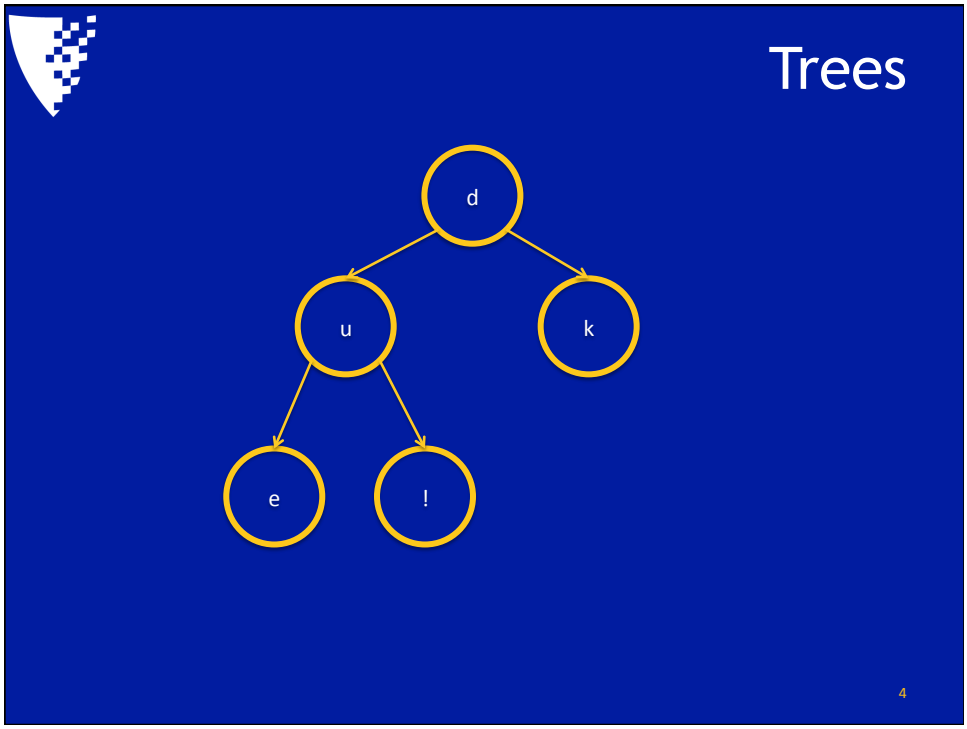
this is a flow chart, but it is similar to a graph

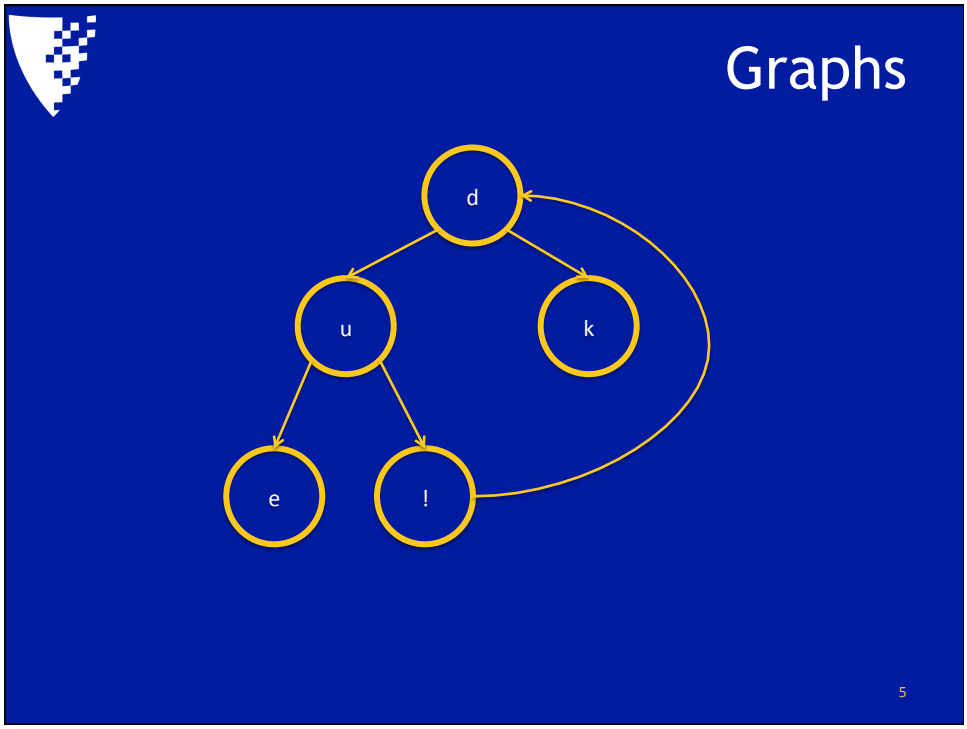
Graphs

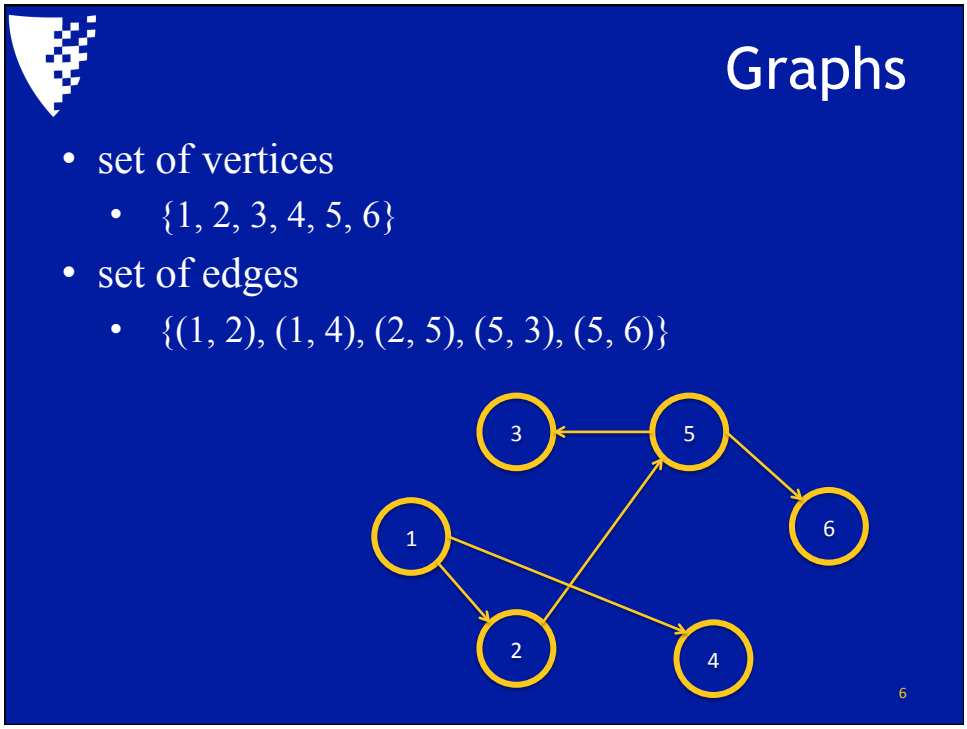
Announcements

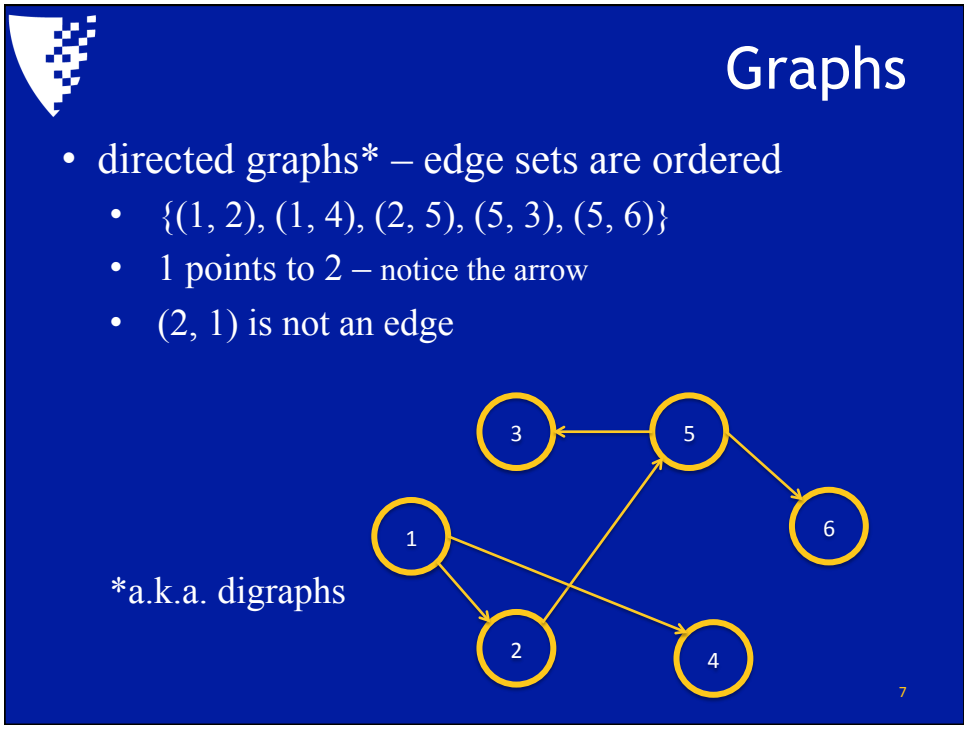
- Huffman – Due April 16
  - You may have a partner
- Burrows-Wheeler – Due April 18
  - You may use the same partner
- Family Tree

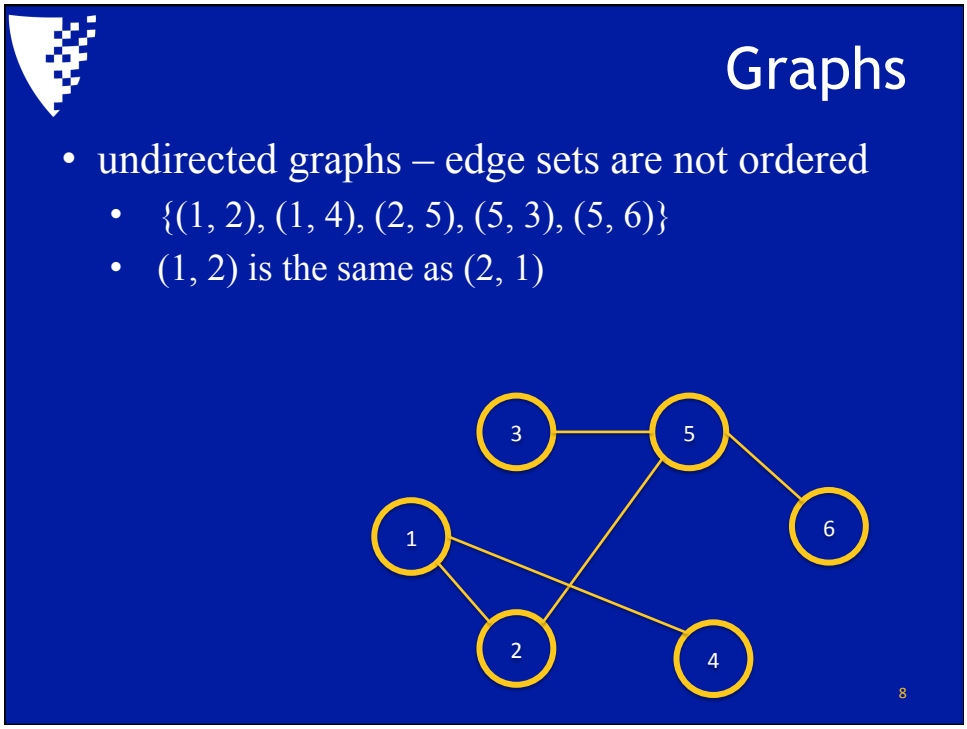

Today
• Intro to graphs
• Coding with graphs
3

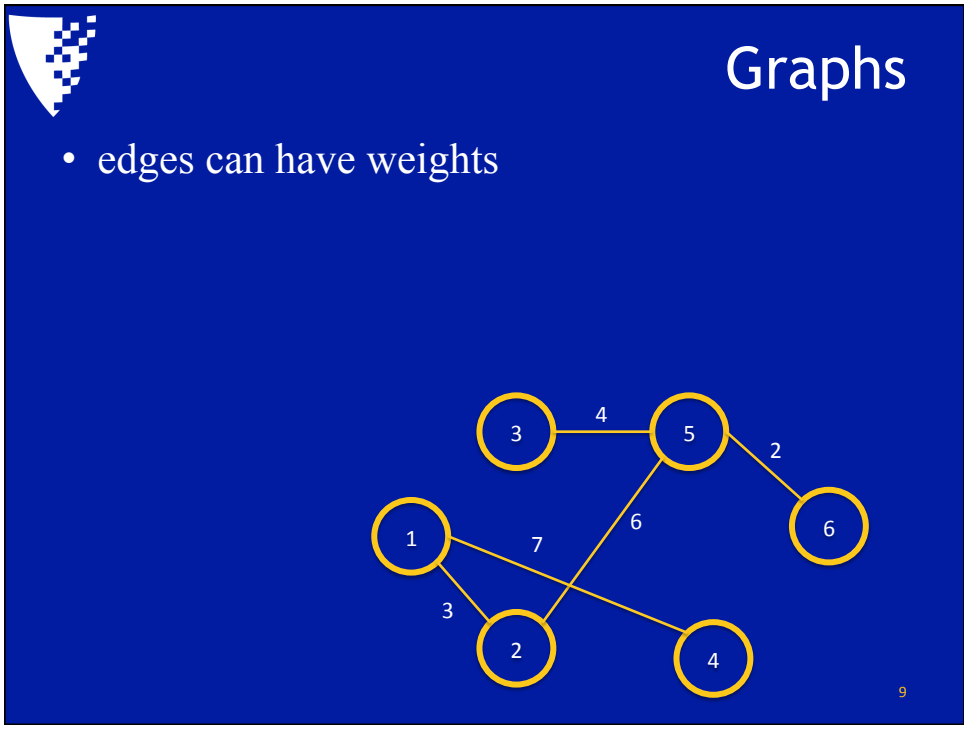

Trees
d
u
k
e
!
4

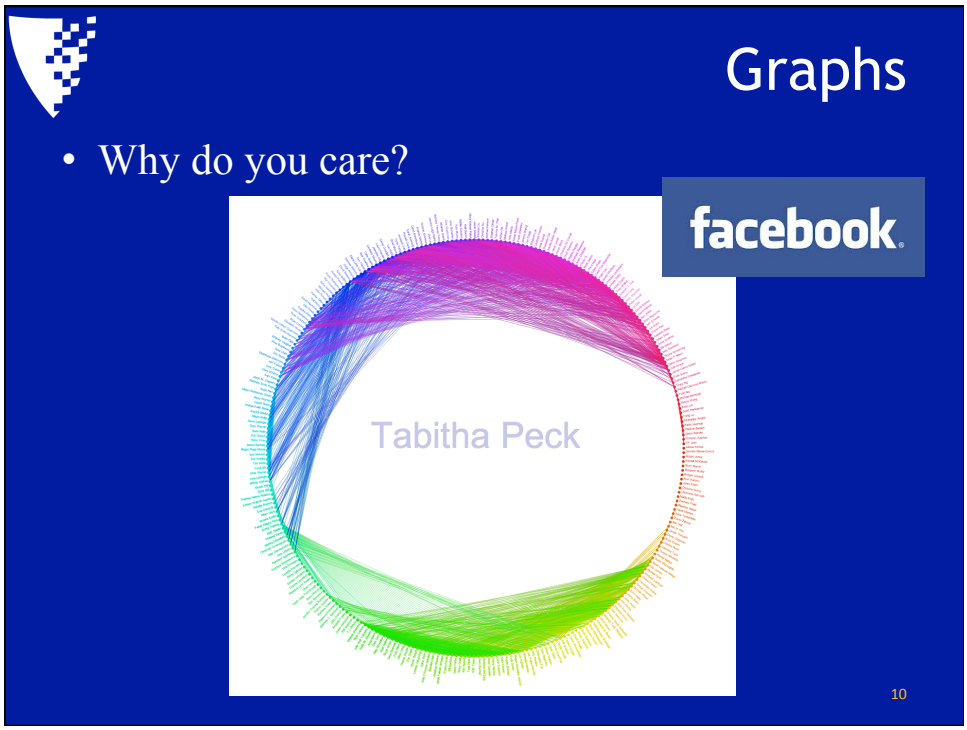

Graphs
d
u
k
e
!
5


Graphs
• set of vertices
• {1, 2, 3, 4, 5, 6}
• set of edges
• {(1, 2), (1, 4), (2, 5), (5, 3), (5, 6)}
3
5
6
1
2
4
6


Graphs
• directed graphs* – edge sets are ordered
• {(1, 2), (1, 4), (2, 5), (5, 3), (5, 6)}
• 1 points to 2 – notice the arrow
• (2, 1) is not an edge
*a.k.a. digraphs
3
5
1
6
2
4
7

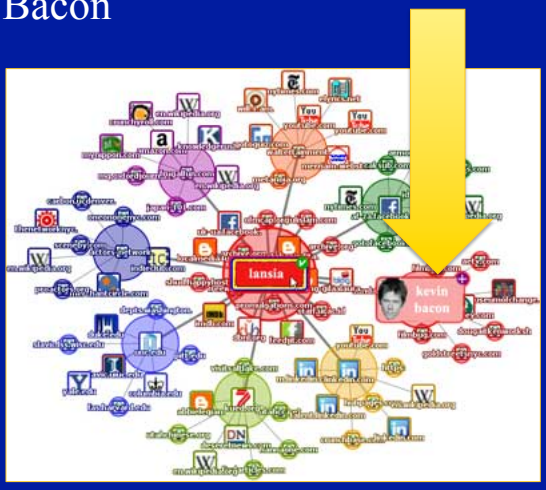

Graphs
• undirected graphs – edge sets are not ordered
• {(1, 2), (1, 4), (2, 5), (5, 3), (5, 6)}
• (1, 2) is the same as (2, 1)
3
5
1
6
2
4
8


Graphs
• edges can have weights
3
4
5
2
1
7
6
6
3
2
4
9


Graphs
• Why do you care?
facebook
Tabitha Peck
10

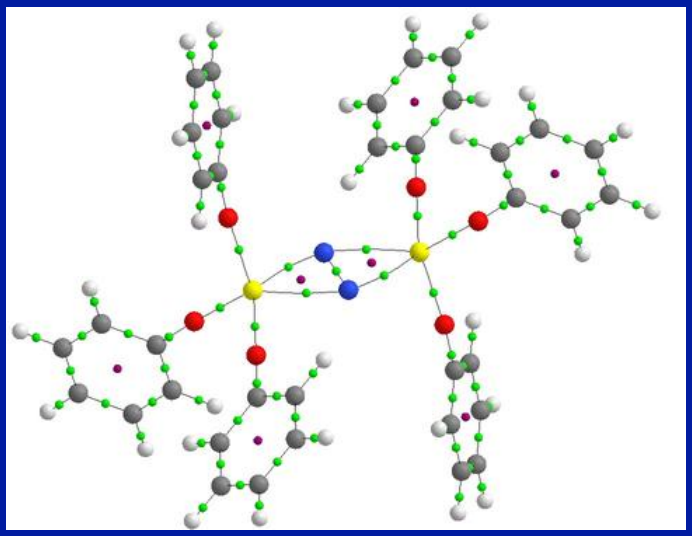
Graphs

- Kevin Bacon



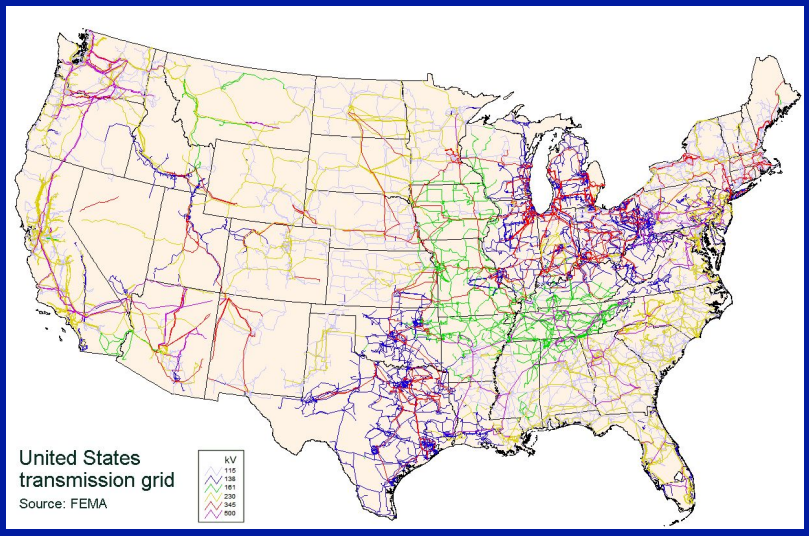
11

Graphs



12

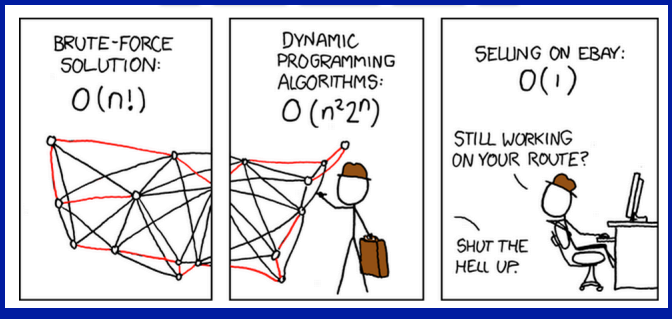
Graphs



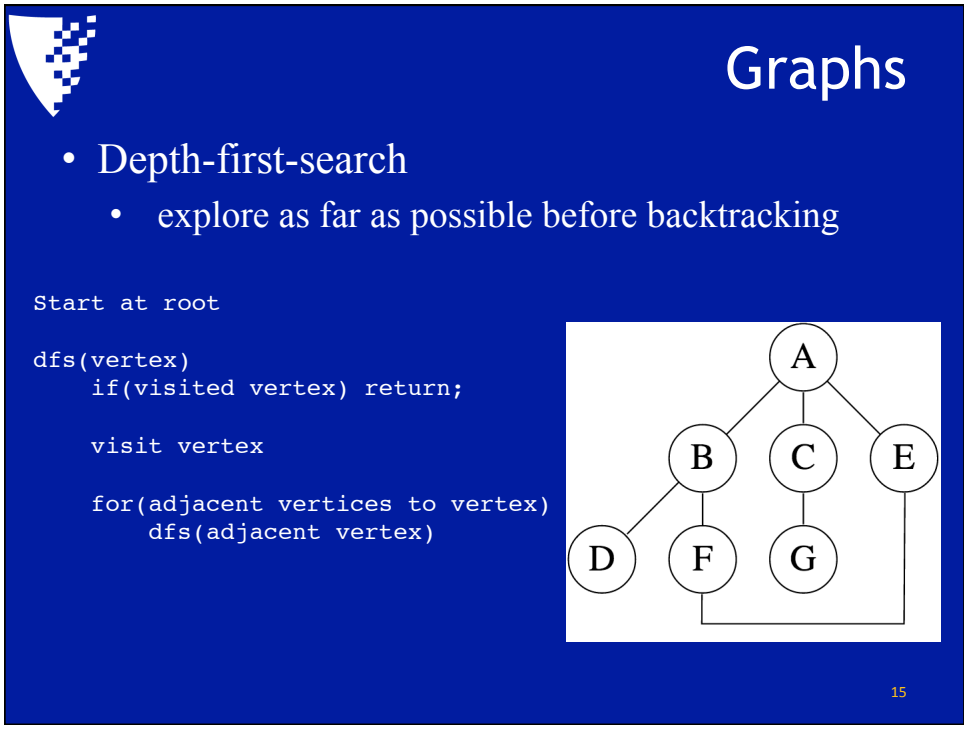
13

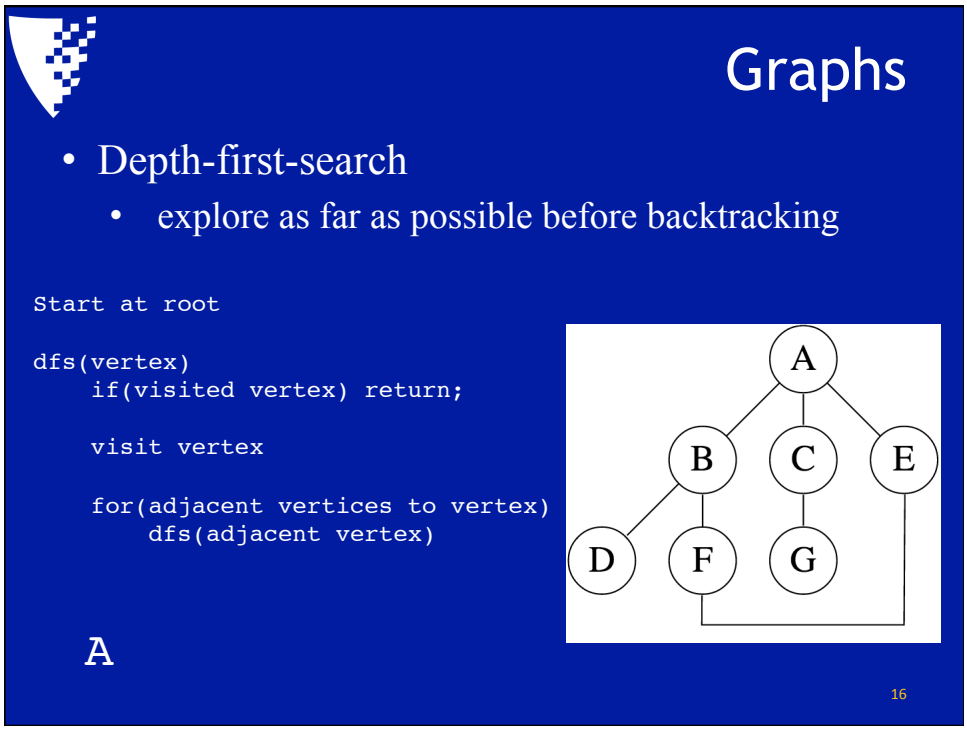
Graphs

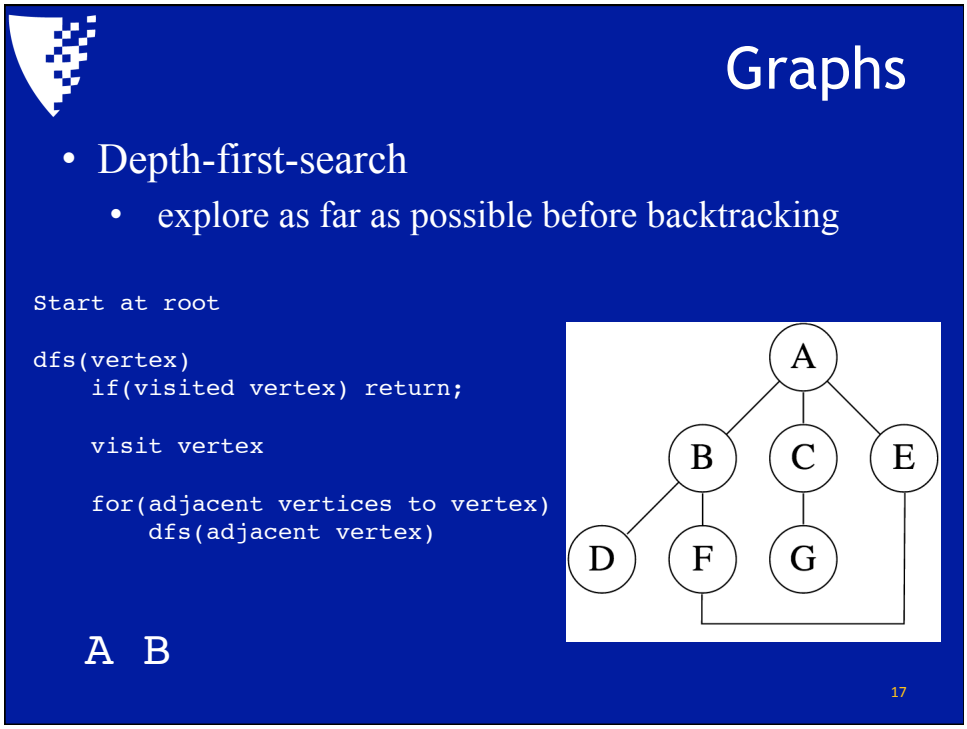
- Traveling salesperson problem
  - Given a list of cities and the distance between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the original city?

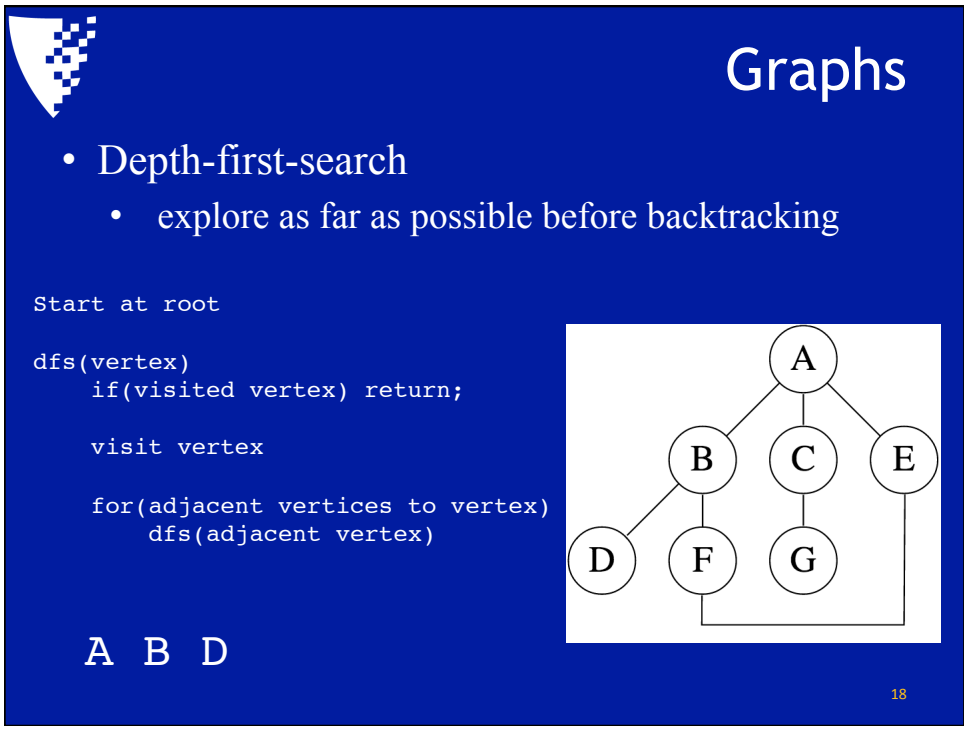


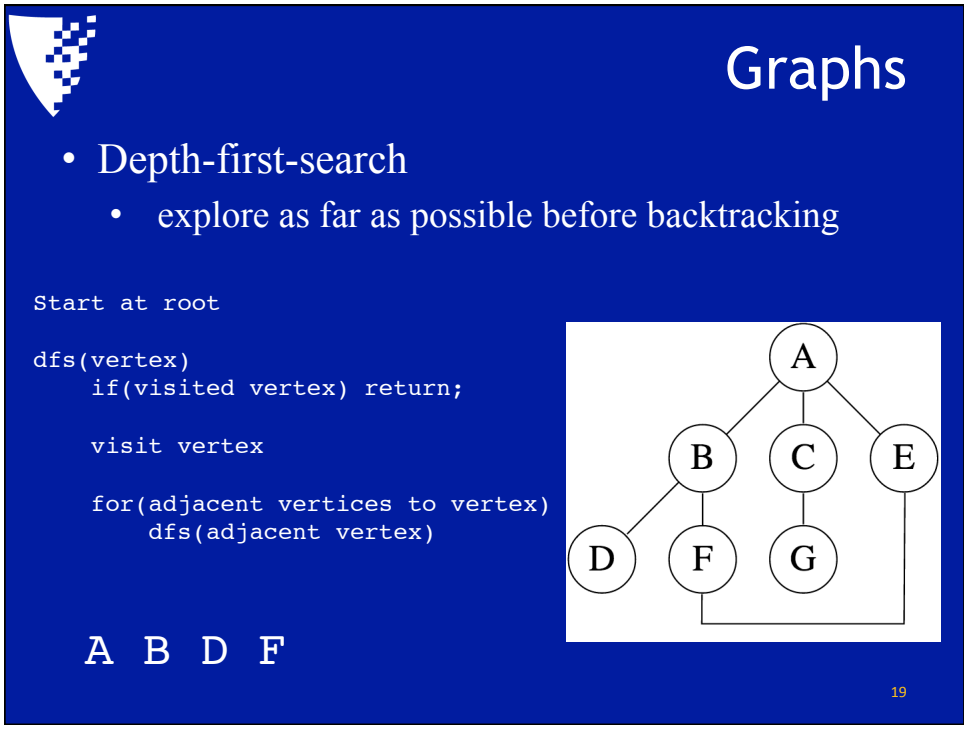
14

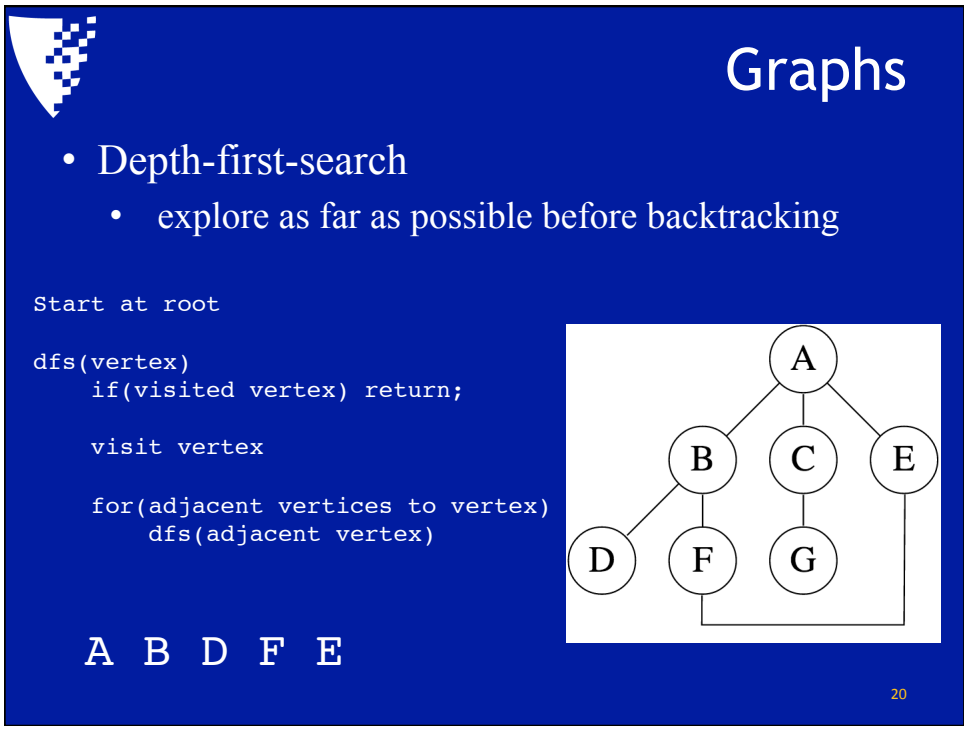

Graphs
• Depth-first-search
• explore as far as possible before backtracking
Start at root
dfs(vertex)
if(visited vertex) return;
visit vertex
for(adjacent vertices to vertex)
dfs(adjacent vertex)
A
B
C
E
D
F
G
15

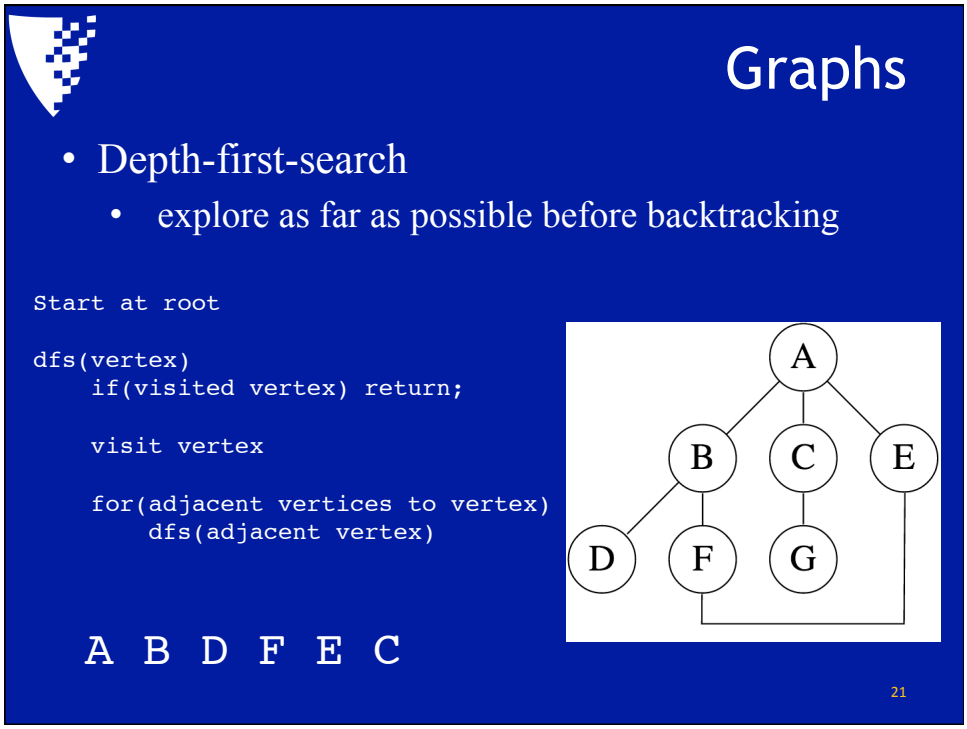

Graphs
• Depth-first-search
• explore as far as possible before backtracking
Start at root
dfs(vertex)
if(visited vertex) return;
visit vertex
for(adjacent vertices to vertex)
dfs(adjacent vertex)
A
A
B
C
E
D
F
G
16

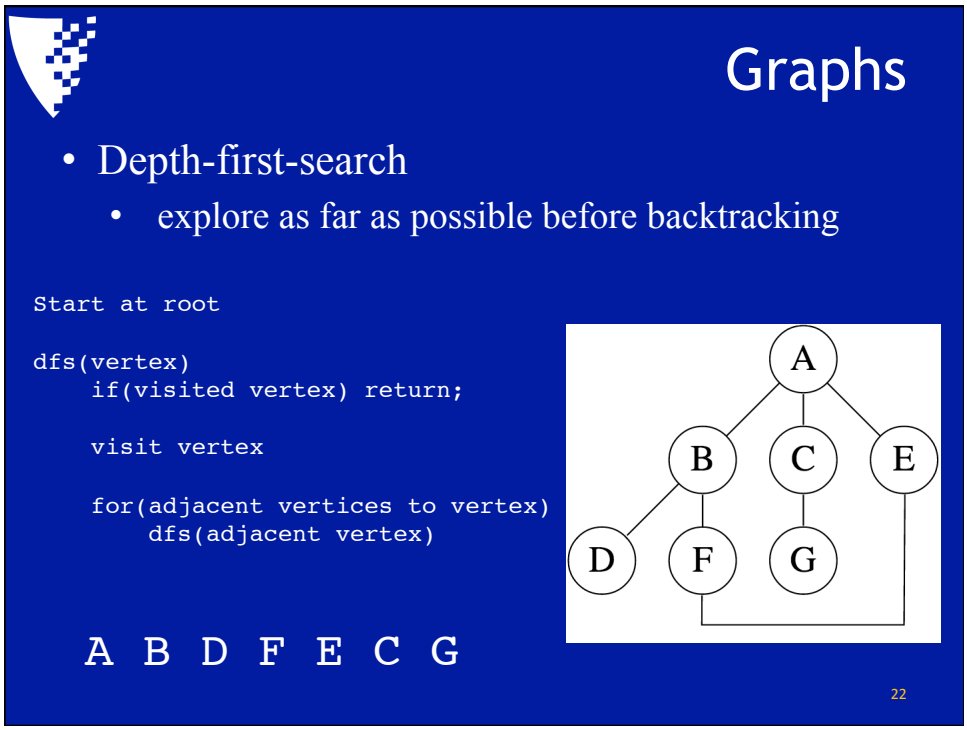

Graphs
• Depth-first-search
• explore as far as possible before backtracking
Start at root
dfs(vertex)
if(visited vertex) return;
visit vertex
for(adjacent vertices to vertex)
dfs(adjacent vertex)
A B
A
B
C
E
D
F
G
17

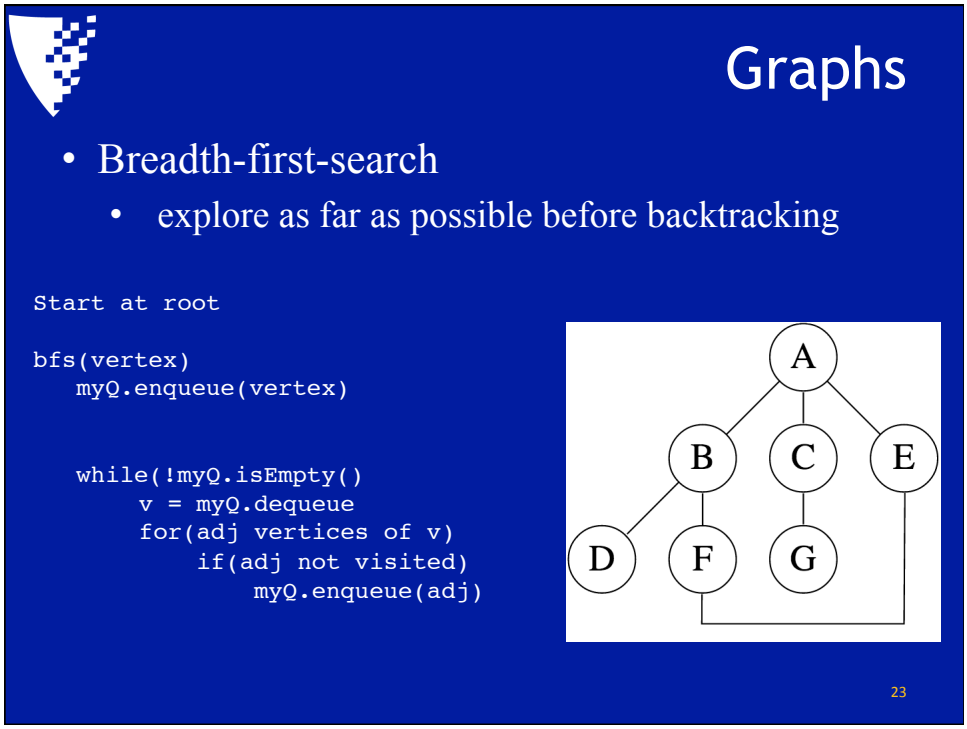

Graphs
• Depth-first-search
• explore as far as possible before backtracking
Start at root
dfs(vertex)
if(visited vertex) return;
visit vertex
for(adjacent vertices to vertex)
dfs(adjacent vertex)
A B D
A
B
C
E
D
F
G
18


Graphs
• Depth-first-search
• explore as far as possible before backtracking
Start at root
dfs(vertex)
if(visited vertex) return;
visit vertex
for(adjacent vertices to vertex)
dfs(adjacent vertex)
A B D F
A
B C E
D F G
19


Graphs
• Depth-first-search
• explore as far as possible before backtracking
Start at root
dfs(vertex)
if(visited vertex) return;
visit vertex
for(adjacent vertices to vertex)
dfs(adjacent vertex)
A B D F E
A
B C E
D F G
20


Graphs
• Depth-first-search
• explore as far as possible before backtracking
Start at root
dfs(vertex)
if(visited vertex) return;
visit vertex
for(adjacent vertices to vertex)
dfs(adjacent vertex)
A B D F E C
A
B C E
D F G
21


Graphs
• Depth-first-search
• explore as far as possible before backtracking
Start at root
dfs(vertex)
if(visited vertex) return;
visit vertex
for(adjacent vertices to vertex)
dfs(adjacent vertex)
A B D F E C G
A
B C E
D F G
22

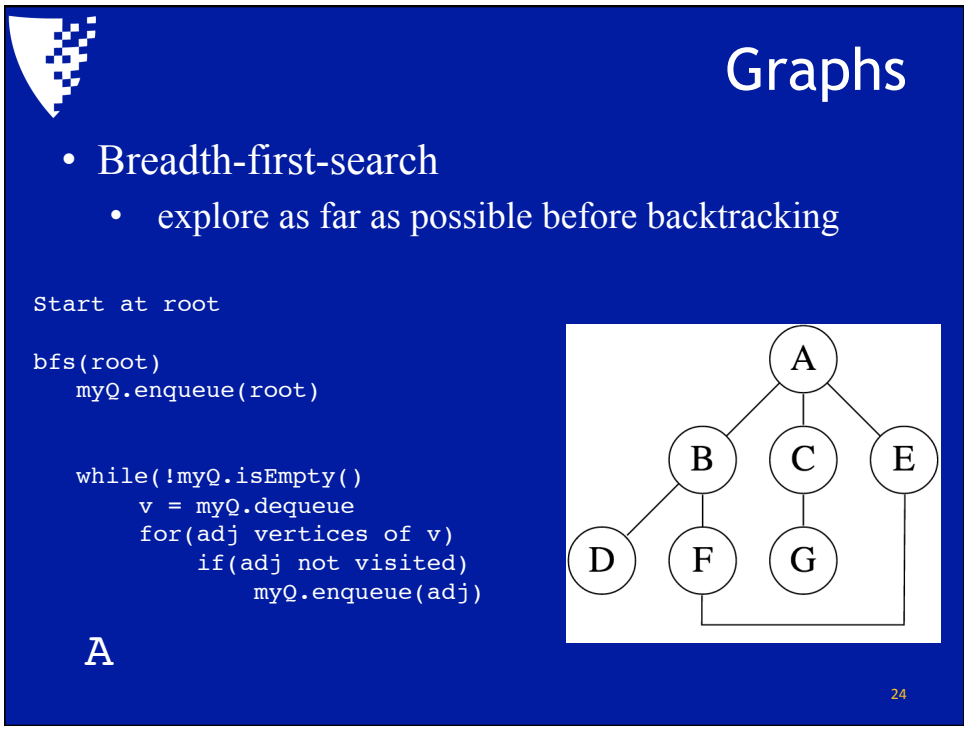


Graphs

- Breadth-first-search
  - explore as far as possible before backtracking

```
Start at root

bfs(vertex)
   myQ.enqueue(vertex)


   while(!myQ.isEmpty()
        v = myQ.dequeue
        for(adj vertices of v)
            if(adj not visited)
                myQ.enqueue(adj)
```


23



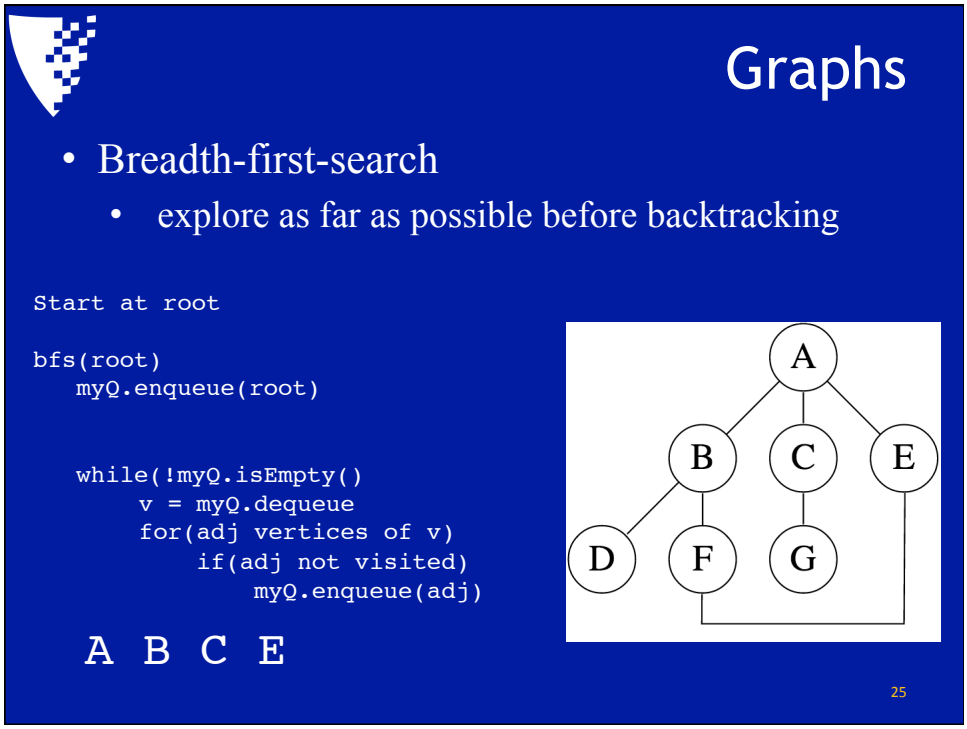
Graphs

- Breadth-first-search
  - explore as far as possible before backtracking

```
Start at root

bfs(root)
   myQ.enqueue(root)


   while(!myQ.isEmpty()
        v = myQ.dequeue
        for(adj vertices of v)
            if(adj not visited)
                myQ.enqueue(adj)
```

A

24



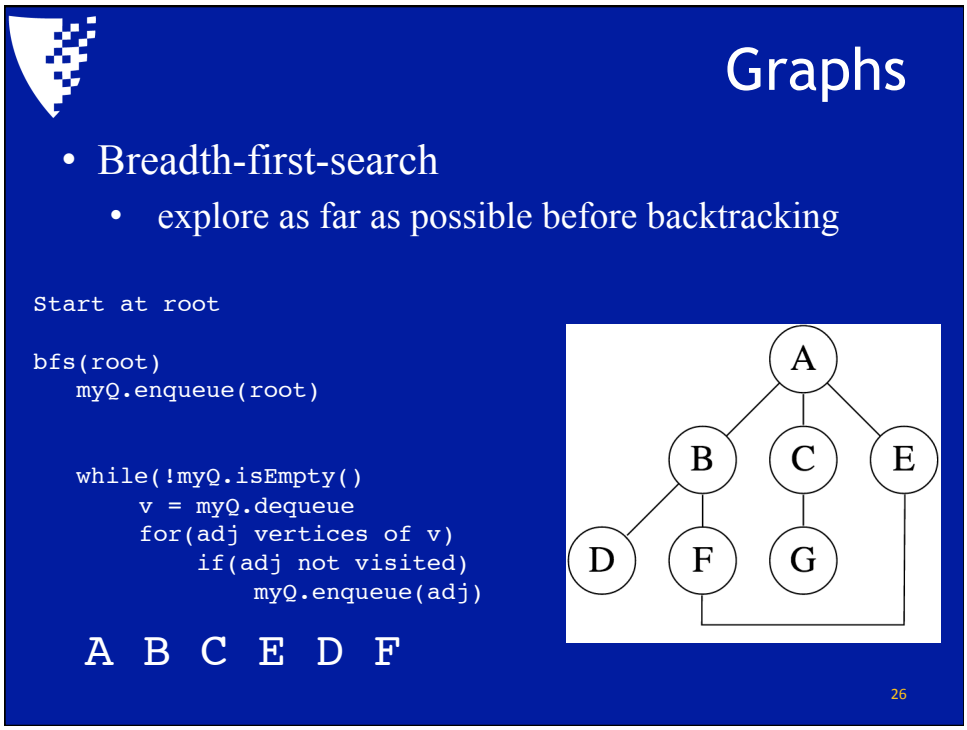
Graphs

- Breadth-first-search
  - explore as far as possible before backtracking

```
Start at root

bfs(root)
   myQ.enqueue(root)


   while(!myQ.isEmpty()
        v = myQ.dequeue
        for(adj vertices of v)
            if(adj not visited)
                myQ.enqueue(adj)
```

A B C E

25



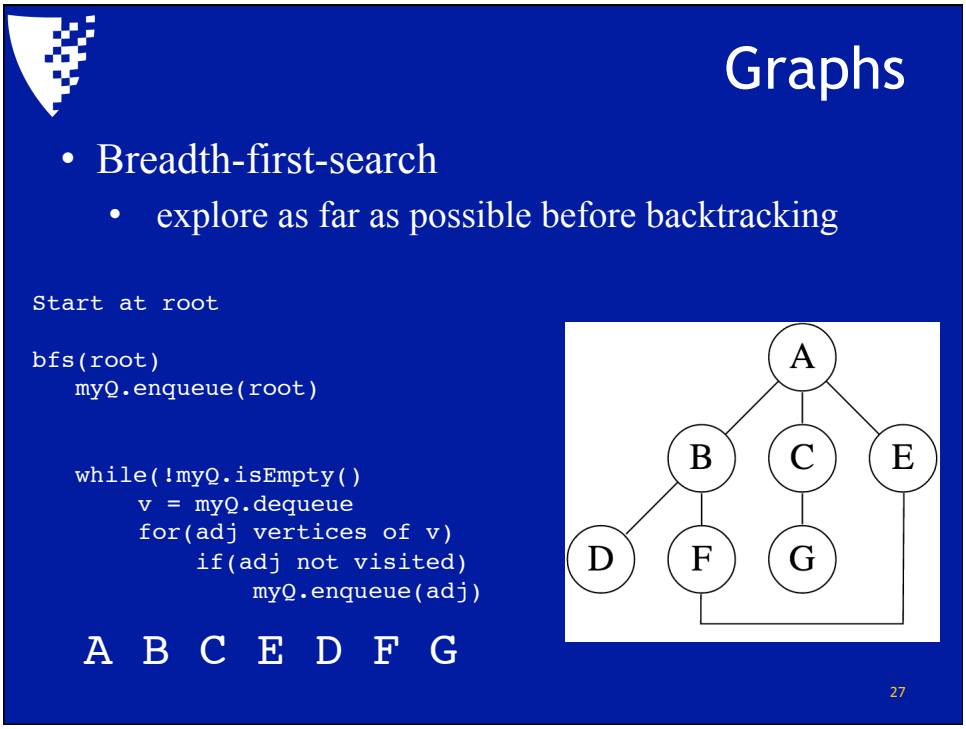
Graphs

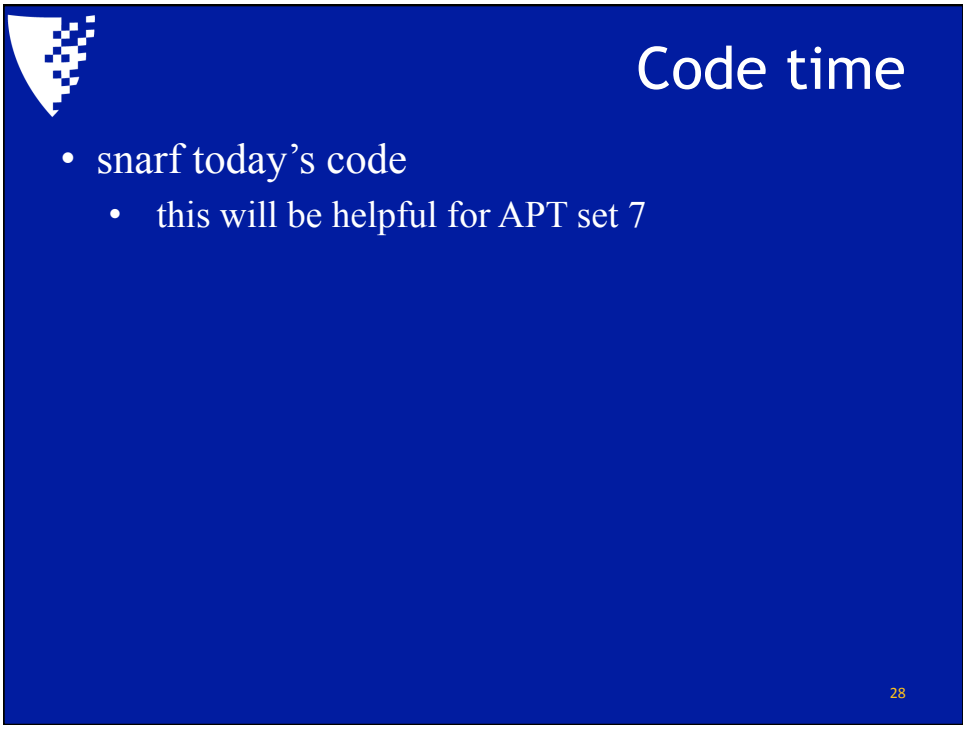
- Breadth-first-search
  - explore as far as possible before backtracking

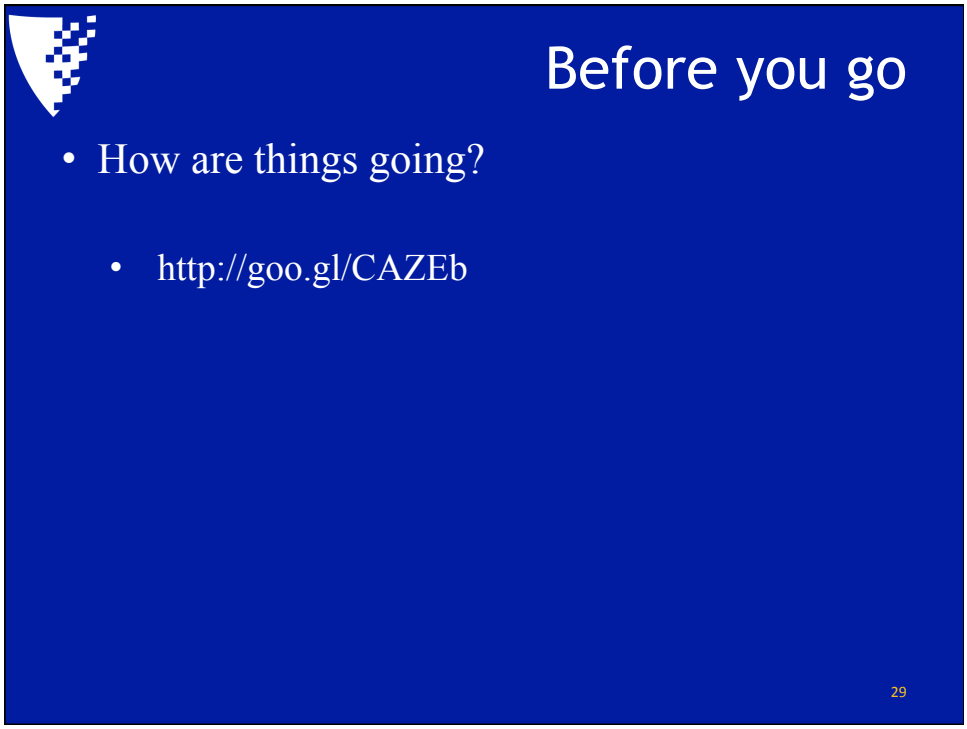
```
Start at root

bfs(root)
   myQ.enqueue(root)


   while(!myQ.isEmpty()
        v = myQ.dequeue
        for(adj vertices of v)
            if(adj not visited)
                myQ.enqueue(adj)
```

A B C E D F

26


Graphs
• Breadth-first-search
• explore as far as possible before backtracking
Start at root
bfs(root)
myQ.enqueue(root)
while(!myQ.isEmpty()
v = myQ.dequeue
for(adj vertices of v)
if(adj not visited)
myQ.enqueue(adj)
A B C E D F G
A
B C E
D F G
27


Code time
• snarf today's code
• this will be helpful for APT set 7
28


Before you go
• How are things going?
• http://goo.gl/CAZEb
29