Graphs

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

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

Trees
Graphs

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

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)
Graphs

• edges can have weights

Graphs

• Why do you care?
Graphs

- Kevin Bacon
• 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?
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)
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)
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 D F E
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 D F E C G
Graphs

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

Start at root

\[
\text{bfs}(\text{vertex}) \\
\quad \text{myQ.enqueue}(\text{vertex}) \\
\]

\[
\text{while(!myQ.isEmpty())} \\
\quad \text{v = myQ.dequeue} \\
\quad \text{for(adj vertices of v)} \\
\quad \quad \text{if(adj not visited)} \\
\quad \quad \quad \text{myQ.enqueue(adj)}
\]
Graphs

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

Start at root

```java
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
```

Graphs

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

Start at root

```java
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

Code time

- snarf today’s code
  - this will be helpful for APT set 7
Before you go

• How are things going?

• http://goo.gl/CAZEb