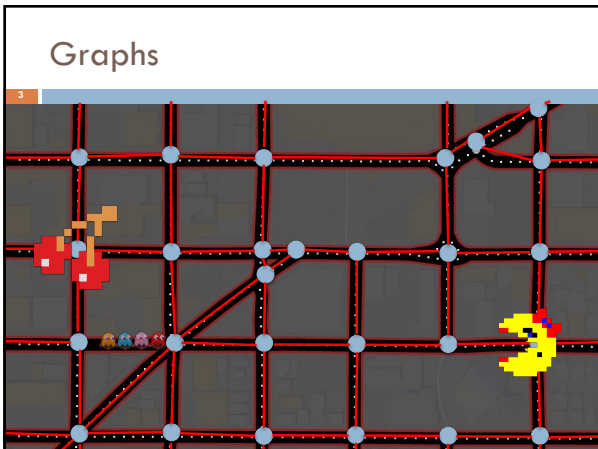


Announcements

- A5 due tonight
- A6 is out, remember to get started early
- For the next lecture, you **MUST** watch the tutorial on the shortest path algorithm beforehand: <http://www.cs.cornell.edu/courses/cs2110/2017fa/online/shortestPath/shortestPath.html>
- The class on 4/10 **will assume** that you understand it. Watch the tutorial once or twice and execute the algorithm on a small graph.
- Complete Quiz 4 by 4/9



Representing Graphs

Adjacency List

```

1 → 2 → 4
2 → 3
3
4 → 2 → 3
    
```

Adjacency Matrix

	1	2	3	4
1	0	1	0	1
2	0	0	1	0
3	0	0	0	0
4	0	1	1	0

Graph Algorithms

- Search
 - Depth-first search
 - Breadth-first search
- Shortest paths
 - Dijkstra's algorithm
- Spanning trees
 - Algorithms based on properties
 - Minimum spanning trees
 - Prim's algorithm
 - Kruskal's algorithm

Search on Graphs

- Given a graph (V, E) and a vertex $u \in V$
- We want to "visit" each node that is reachable from u

There are many paths to some nodes.

How do we visit all nodes efficiently, without doing extra work?

Depth-First Search

Intuition: Recursively visit all vertices that are reachable along unvisited paths.

```

/** Visit all nodes reachable on unvisited paths from u.
Precondition: u is unvisited.
*/
public static void dfs(int u)
{
    visit(u);
    for all edges (u,v):
        if(!visited[v]):
            dfs(v);
}
    
```

dfs(1) visits the nodes in this order: 1, 2, 3, 5, 7, 8

Depth-First Search

Intuition: Recursively visit all vertices that are reachable along unvisited paths.

```

/** Visit all nodes reachable on unvisited paths from u.
Precondition: u is unvisited.
*/
public static void dfs(int u)
{
    visit(u);
    for all edges (u,v):
        if(!visited[v]):
            dfs(v);
}
    
```

Suppose there are n vertices that are reachable along unvisited paths and m edges:
 Worst-case running time? $O(n + m)$
 Worst-case space? $O(n)$

DFS Quiz

In what order would a DFS visit the vertices of this graph? Break ties by visiting the lower-numbered vertex first.

- 1, 2, 3, 4, 5, 6, 7, 8
- 1, 2, 5, 6, 3, 6, 7, 4, 7, 8
- 1, 2, 5, 3, 6, 4, 7, 8
- 1, 2, 5, 6, 3, 7, 4, 8

Depth-First Search in Java

```

public class Node {
    boolean visited;
    List<Node> neighbors;
}

/** Visit all nodes reachable on unvisited paths from this node.
Precondition: this node is unvisited.
*/
public void dfs() {
    visited = true;
    for (Node n: neighbors) {
        if (!n.visited) n.dfs();
    }
}
    
```

Each vertex of the graph is an object of type Node

No need for a parameter. The object is the node.

Depth-First Search Iteratively

Intuition: Visit all vertices that are reachable along unvisited paths from the current node.

```

/** Visit all nodes reachable on unvisited paths from u.
Precondition: u is unvisited.
*/
public static void dfs(int u) {
    Stack s = (u); // Not Java!
    while (s is not empty) {
        u = s.pop();
        if (u not visited) {
            visit u;
            for each edge (u, v):
                s.push(v);
        }
    }
}
    
```

Stack: 8, 5, 3

Breadth-First Search

Intuition: Iteratively process the graph in "layers" moving further away from the source node.

```

/** Visit all nodes reachable on unvisited paths from u.
Precondition: u is unvisited.
*/
public static void bfs(int u) {
    Queue q = (u); // Not Java!
    while (q is not empty) {
        u = q.remove();
        if (u not visited) {
            visit u;
            for each (u, v):
                q.add(v);
        }
    }
}
    
```

Queue: 2, 5, 7, 3, 5, 8, 5

Analyzing BFS

13

Intuition: Iteratively process the graph in "layers" moving further away from the source node.

```

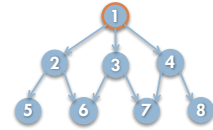
/** Visit all nodes reachable on
unvisited paths from u.
Precondition: u is unvisited. */
public static void bfs(int u) {
    Queue q= (u); // Not Java!
    while ( ) {
        u= q.remove();
        if (u not visited) {
            visit u;
            for each (u, v):
                q.add(v);
        }
    }
}
    
```

Suppose there are n vertices that are reachable along unvisited paths and m edges:
 Worst-case running time? $O(n + m)$
 Worst-case space? $O(m)$

BFS Quiz

14

- In what order would a BFS visit the vertices of this graph? Break ties by visiting the lower-numbered vertex first.



- 1, 2, 3, 4, 5, 6, 7, 8
- 1, 2, 3, 4, 5, 6, 6, 7, 7, 8
- 1, 2, 5, 3, 6, 4, 7, 8
- 1, 2, 5, 6, 3, 7, 4, 8

Comparing Search Algorithms

15

DFS

- Visits: 1, 2, 3, 5, 7, 8
- Time: $O(n + m)$
- Space: $O(n)$

BFS

- Visits: 1, 2, 5, 7, 3, 8
- Time: $O(n + m)$
- Space: $O(m)$

