CSE 421
Algorithms

Richard Anderson

Lecture 5

Graph Theory

Announcements

- Monday's class will be held in CSE 305
- Reading
  - Chapter 3
  - Start on Chapter 4

Explain that there will be some review from 326

Graph Theory

By default |V| = n and |E| = m

- G = (V, E)
  - V – vertices
  - E – edges
- Undirected graphs
  - Edges sets of two vertices {u, v}
- Directed graphs
  - Edges ordered pairs (u, v)
- Many other flavors
  - Edge / vertices weights
  - Parallel edges
  - Self loops

Definitions

- Path: $v_1, v_2, \ldots, v_k$, with (v_i, v_{i+1}) in E
  - Simple Path
  - Cycle
  - Simple Cycle
- Distance
- Connectivity
  - Undirected
  - Directed (strong connectivity)
- Trees
  - Rooted
  - Unrooted

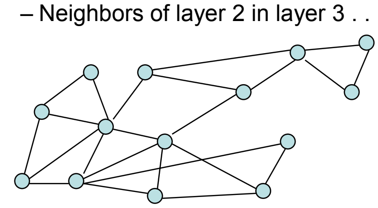
Graph search

- Find a path from s to t

```
S = {s}
While there exists (u, v) in E with u in S and v not in S
        Pred[v] = u
        Add v to S
        if (v = t) then path found
```

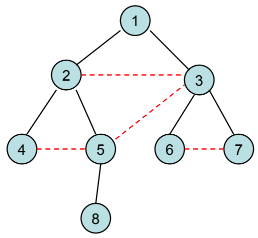
Breadth first search

- Explore vertices in layers
  - s in layer 1
  - Neighbors of s in layer 2
  - Neighbors of layer 2 in layer 3 . . .



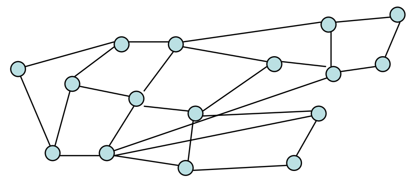
Key observation

- All edges go between vertices on the same layer or adjacent layers



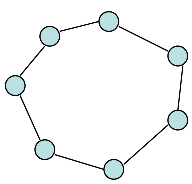
Bipartite

- A graph V is bipartite if V can be partitioned into V_1, V_2 such that all edges go between V_1 and V_2
- A graph is bipartite if it can be two colored



Testing Bipartiteness

- If a graph contains an odd cycle, it is not bipartite



Algorithm

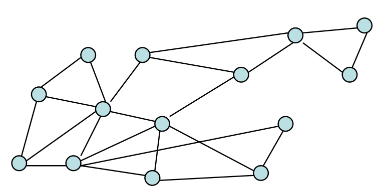
- Run BFS
- Color odd layers red, even layers blue
- If no edges between the same layer, the graph is bipartite
- If edge between two vertices of the same layer, then there is an odd cycle, and the graph is not bipartite

Corollary

- A graph is bipartite if and only if it has no Odd Length Cycle

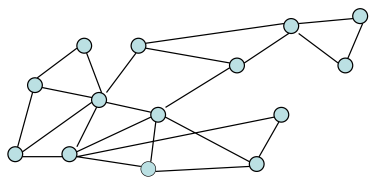
Depth first search

- Explore vertices from most recently visited



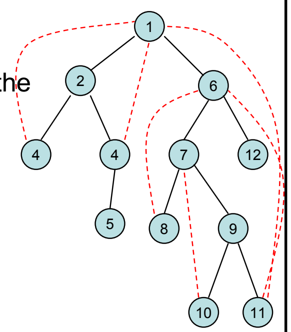
Recursive DFS

```
DFS(u)
    Mark u as “Explored”
    Foreach v in Neighborhood of u
        If v is not “Explored”,  DFS(v)
```



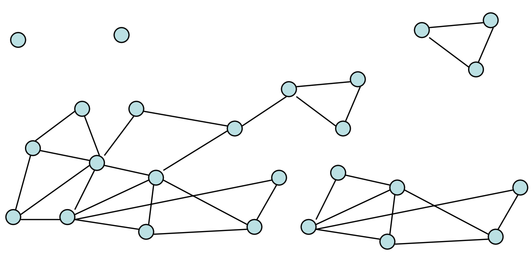
Key observation

- Each edge goes between vertices on the same branch
- No cross edges



Connected Components

- Undirected Graphs



Strongly Connected Components

- Directed Graphs

There is an O(n+m) algorithm that we will not be covering