6.006 Proudly Presents

- Graph Traversal
- BFS
- DFS
- Topological Sorting
Breadth-First Search
a.k.a. BFS (not BFG)

- Fix your source
- Visit all the neighbors
- Then visit all the neighbors’ neighbors
- Then all the neighbors’ neighbors’ neighbors’
- ...

Diagram:
```
    1
   / \
  2   3
 / \ / \   
4  5 6  7
 / \ / \ / \  
8  9 10 11
```
BFS in Python: Design

- Use the `graph` module shown before, and Python’s `deque`

- Encapsulate traversal data in a class, return at the end of the traversal

- Implement traversal as stand-alone function

```python
from graph import *
from collections import deque

class BFSResults:
    def __init__(self):
        self.level = dict()
        self.parent = dict()
```
```python
def bfs(g, s):
    r = BFSResults()
    actives = deque()
    actives.append(s)
    r.parent[s] = None
    r.level[s] = 0

    while len(actives):
        v = actives.popleft()
        for n in g.neighbors(v):
            if n not in r.parent:
                r.parent[n] = v
                r.level[n] = r.level[v] + 1
                actives.append(n)
    return r
```
Depth-First Search
a.k.a. Backtracking

- Fix your source
- Move to its first neighbor
- Then to that guy’s first neighbor
- ...
- When stuck, backtrack and visit next neighbor
DFS in Python: Design

• Use the graph module shown before

• Encapsulate traversal data in a class, return at the end of the traversal

• Implement traversal as stand-alone function

```python
1 from graph import *
2
3 class DFSResults:
4    def __init__(self):
5        self.parent = dict()
6        self.time = dict()
7        self.vertices = list()
8        self.t = 0
```
DFS in Python: Code

```python
def dfs(g):
    results = DFSResults()
    for vertex in g.itervertices():
        if vertex not in results.parent:
            dfs_visit(g, vertex, results)
    return results

def dfs_visit(g, v, results, parent = None):
    results.vertices.append(v)
    results.parent[v] = parent
    for n in g.neighbors(v):
        if n not in results.parent:
            dfs_visit(g, n, results, v)
    results.t += 1
    results.time[v] = results.t
```
## DFS and CLRS Colors

<table>
<thead>
<tr>
<th>Color</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>White</td>
<td>vertex not in parents</td>
</tr>
<tr>
<td>Gray</td>
<td>vertex in parents and vertex not in time</td>
</tr>
<tr>
<td>Black</td>
<td>vertex in time</td>
</tr>
</tbody>
</table>
Application: Porting BFS and DFS to a New Platform
Disclaimers
(Please Don’t Sue Me!)

• You may close your eyes and cover your ears if you find this material offensive

• If you are under 13 and your mommy doesn’t allow you on the Internet: please close your eyes

• Under 18: please don’t use this knowledge to do something inappropriate for your age
Stalking Hotties on Facebook

- Our Platform: Firefox 3.0b4
  - any browser with tabs would do

- Profiles + Friendship = Graph

- Our mission:
  - apply DFS and BFS to the fine art of stalking hot boys/babes on Facebook
Hueihan’s Heuristic

• “Hot boys have hot friends”

• Heuristics are useful in huge graphs, with multiple solutions
  • Goal: avoid visiting most of the graph
  • So we’ll only follow paths of hot* people
Facebook as Graph

• Traversal: go to ‘Friends’ to display all your friends (like g.neighbors)

• BFS: the tabs are a queue - open all friends profiles in new tabs, then close current tab and go to the next one

• DFS: the history is a stack - open the first hot friend profile in the same window; when hitting a dead end, use back button
Topological Sorting

even your Course 15 friends know it
Topological Sorting

- Do a DFS on the graph, record exiting times for the nodes
- Sort the nodes in the inverse order of the exit times (just draw it!)
- A node is never exited before a node it points to is exited

```python
1 def topological_sort(graph):
2     dfs_result = dfs(graph)
3     top = [None for i in dfs_result.vertices]
4     count = len(dfs_result.vertices)
5     for vertex in dfs_result.time:
6         top[count - dfs_result.time[vertex]] = vertex
7     return top
```
Topological Sorting
Topological Sorting

![Diagram of a graph with nodes labeled F, D, N, E, P, W, 'W, T. Connections are indicated by arrows. Below the diagram is a table with rows labeled F, D, N, E, P, W, 'W, T and columns labeled 6, 1, 3, 2, 5, 4, 8, 7, 'W, T, F, P, W, N, E, D.]}
Two-Way BFS

Discussion on Implementation
• Did BFS and DFS C-style, should have done them in the style of high-level languages (same code for both, replace a queue with a stack) - “the code is hard to read”

• The Facebook example requires Internet, so make sure to have it

• a script might help (it was too random)