Lecture 25: Monday November 5, 2001

today

- graph traversal (finish)
  - BFS and DFS trees
  - classifying graph edges with BFS/DFS

next

- DFS application: biconnected components

announcements

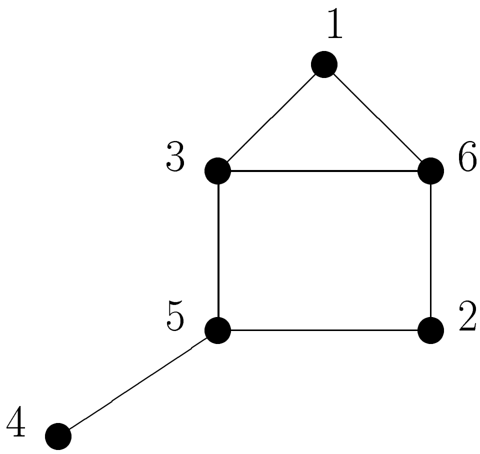
- midterm II: Wednesday November 7
- pre-midterm office hours
  - today 3:30-4:30
  - none on Tuesday

DFS analysis

- similar to BFS
- $n = |V|\ m = |E|$
- recall handshaking lemma: $\sum_{v \in V} deg(v) = 2|E|$
- DFS analysis:
  - adjacency matrix representation
    - space $\qquad \Theta(n^2)$
    - time $\quad \Theta(n + \sum_v n = n + n^2) \qquad \Theta(n^2)$
  - adjacency list representation
    - space $\quad \Theta(n + \sum_v d(v) = n + 2m) \qquad \Theta(n + m)$
    - time $\quad \Theta(n + \sum_v d(v) = n + 2m) \qquad \Theta(n + m)$

classifying graph edges with BFS/DFS

- given a BFS/DFS forest, where are the possible other edges?
- during graph traversal, all edges/vertices examined
- traversal forest:
  - tree root: starting vertex for that component
  - tree edge (parent p, child c): c discovered while processing p
- during graph traversal, each edge encountered twice (from each end)
- wrt traversal tree, categorize graph edges by first encounter
  - tree edges
  - graph edges which are not tree edges are called:
    - back edges — to ancestor
    - forward edges — to descendant
    - cross edges — to non-ancestor, non-descendant




```
1:   3 6
2:   5 6
3:   1 5 6
4:   5
5:   2 3 4
6:   1 2 3
```

BFS(G,6)

DFS-visit(6)

properties of BFS, DFS

- classifying graph edges with BFS
  - each graph edge: node levels differ by ≤ 1 why?
  - so no back edges
  - so no forward edges
  - cross edges
    - level edges within same level
    - downleft edges child of previous node at same level
- classifying graph edges with DFS
  - every non-tree edge is back edge why?
  - so no forward edges
  - so no cross edges
- BFS vertex order level-order of each tree of BFS forest
- DFS vertex order pre-order of each tree of DFS forest

rooted tree traversals: level/pre/in/post-order

- each describes an order for visiting all nodes
- in rooted tree, children of a node:
  - arbitrary number, ordered first to last
  - when drawing: first $\leftrightarrow$ leftmost, last $\leftrightarrow$ rightmost

```
level-order(T)
  visit(root of T)
  visit nodes at level 1 (children of root) left to right
  visit nodes at level 2 (grandchildren " ) left to right
  ...

pre-order(T)
  visit(root of T)
  pre-order(first subtree)
  ...
  pre-order(last subtree)

in-order(T)                { only defined for binary trees
  in-order(left subtree)
  visit(root of T)
  in-order(right subtree)

post-order(T)
  post-order(first subtree)
  ...
  post-order(last subtree)
  visit(root of T)
```

DFS discovery/finish times

- useful in proving correctness for DFS algorithms

```
DFS-visit(u)
  color[u] <- GRAY            { discover u
  dtime[u] <- time  <- 1+time
  for each v in Adj[u] do     { nbrs of u
    if color[v]= WHITE then
      p[v] <- u
      DFS-visit(u)
  color[u] <- BLACK           { finished u
  ftime[u] <- time  <- 1+time
```

- DFS parenthesis theorem: for x,y with disc[x] < disc[y]
  - disc[x] < finish[x] < disc[y] < finish[y],
    and neither is descendant of other in DFS forest
  - disc[x] < disc[y] < finish[y] < finish[x]
    and y is descendant of x in DFS forest

midterm II

- cumulative, but mostly since Midterm I
- heapsort
  - trace, analysis (WC, BC, AC), correctness
- priority queue ADT
- quicksort
  - trace, analysis (WC, BC, AC), correctness
  - improvements, removing tail recursion
- c.b. sort lower bounds
  - decision tree
- dynamic programming
  - memoize, Fibonacci, Catalan, chain MM
  - run time of nested loops
- disjoint sets ADT
  - representation array
  - rooted forest: ordinary/rank union, ordinary/compressed find
  - $\log^* n$, $\alpha(n)$
- graphs
  - definitions, handshaking lemma