Lecture 24: Friday November 2, 2001

today

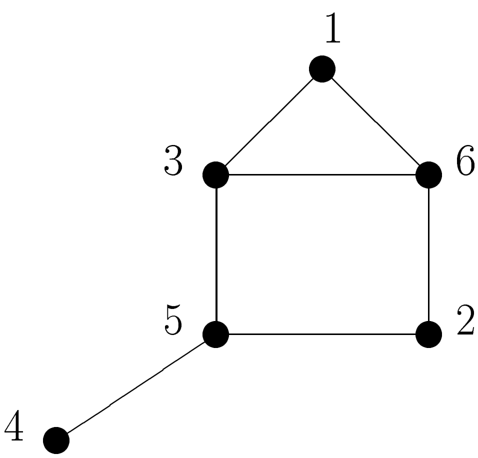
- graphs
  - traversal: breadth first search
    * the alg'm
    * analysis
    * BFS trees
    * BFS and distance
  - traversal: depth first search
    * alg'm
    * analysis
    * DFS trees
    * DFS and biconnected components

announcements

- assignment 6: due Monday November 5
- midterm II: Wednesday November 7
- pre-midterm office hours
  - today 3:30-4:30
  - Monday 3:30-4:30
  - none on Tuesday

breadth first search

- traversal: examine all edges and vertices
- BFS: vertices waiting to be processed in queue
- vertex colours:
  - white: unencountered
  - gray: in queue
  - black: finished processing
- vertex processing
  - for each neighbour: if white, then enqueue
  - processing order depends on order neighbours are examined
- BFS tree
  - root is start vertex
  - parent of node x is predecessor[x]

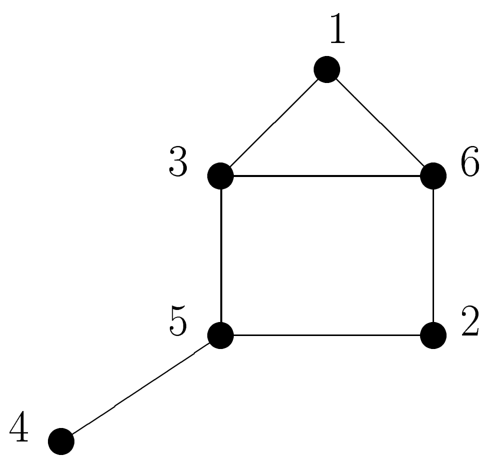



```
1:   3 6
2:   5 6
3:   1 5 6
4:   5
5:   2 3 4
6:   1 2 3
```

	1	2	3	4	5	6
1			*			*
2					*	*
3	*				*	*
4					*	
5		*	*	*		
6	*	*	*			

BFS(G,1)

BFS(G,2)




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

BFS(G,1)

BFS(G,2)

BFS analysis

- $n = |V|$ $m = |E|$
- recall handshaking lemma: $\sum_{v \in V} deg(v) = 2|E|$
- BFS analysis:
  - each vertex enqueued only once — white $\rightarrow$ gray
  - each vertex dequeued only once — gray $\rightarrow$ black
  - adjacency matrix representation
    - space — $\Theta(n^2)$
    - time $\Theta(n + \sum_v n = n + n^2)$ — $\Theta(n^2)$
  - adjacency list representation
    - space $\Theta(n + \sum_v d(v) = n + 2m)$ — $\Theta(n + m)$
    - time $\Theta(n + \sum_v d(v) = n + 2m)$ — $\Theta(n + m)$

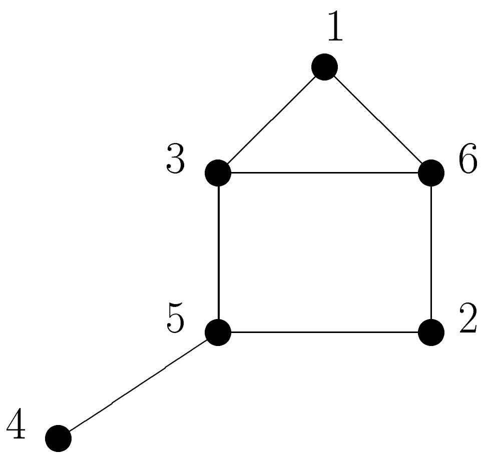
depth first search: recursive traversal

```
DFS(G)                  { G=(V,E)
  for each v in V do
    color[v] <- WHITE  { unknown
    p[v]     <- NIL    { predecessor
  time <- 0
  for each v in V do
    if color[v]=WHITE then DFS-visit(v)

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
```

WARNING: DFS(G) versus BFS(G,s)

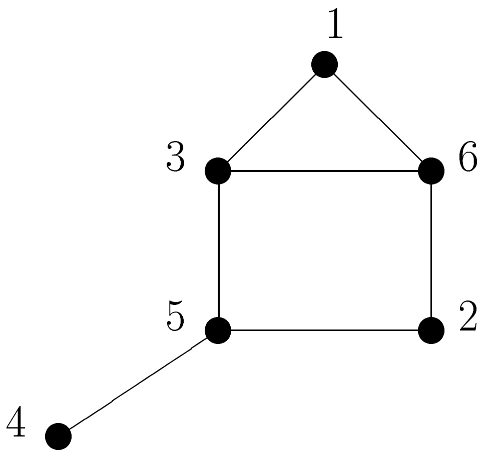
- DFS(G): traverses all components of the graph
- DFS-visit(u): never leaves the component containing u
- BFS(G,s): never leaves the component containing s
- exercise: write BFS(G) which traverses all components




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

	1	2	3	4	5	6
1			*			*
2					*	*
3	*				*	*
4					*	
5		*	*	*		
6	*	*	*			

DFS(G)




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

DFS(G)