CMPS 3120

Algorithm Analysis

Dr. Chengwei Lei
CEECS
California State University, Bakersfield
Many problems require processing all graph vertices (and edges) in systematic fashion

Graph traversal algorithms:

- Depth-first search (DFS)
- Breadth-first search (BFS)
Depth-First Search (DFS)

- Visits graph’s vertices by always moving away from last visited vertex to unvisited one, backtracks if no adjacent unvisited vertex is available.

- Uses a stack
  - a vertex is pushed onto the stack when it’s reached for the first time
  - a vertex is popped off the stack when it becomes a dead end, i.e., when there is no adjacent unvisited vertex

- “Redraws” graph in tree-like fashion (with tree edges and back edges for undirected graph)
Pseudocode of DFS

**ALGORITHM**  
$DFS(G)$

// Implements a depth-first search traversal of a given graph
// Input: Graph $G = (V, E)$
// Output: Graph $G$ with its vertices marked with consecutive integers
// in the order they've been first encountered by the DFS traversal
mark each vertex in $V$ with 0 as a mark of being “unvisited”

$count \leftarrow 0$

for each vertex $v$ in $V$ do
  if $v$ is marked with 0
    $dfs(v)$

$dfs(v)$
  // visits recursively all the unvisited vertices connected to vertex $v$ by a path
  // and numbers them in the order they are encountered
  // via global variable $count$
  $count \leftarrow count + 1$; mark $v$ with $count$

  for each vertex $w$ in $V$ adjacent to $v$ do
    if $w$ is marked with 0
      $dfs(w)$
Example: DFS traversal of undirected graph

DFS traversal stack:  
DFS tree:
Notes on DFS

- DFS can be implemented with graphs represented as:
  - adjacency matrices: $\Theta(V^2)$
  - adjacency lists: $\Theta(|V| + |E|)$

- Yields two distinct ordering of vertices:
  - order in which vertices are first encountered (pushed onto stack)
  - order in which vertices become dead-ends (popped off stack)

- Applications:
  - checking connectivity, finding connected components
  - checking acyclicity
  - finding articulation points and biconnected components
  - searching state-space of problems for solution (AI)
Breadth-first search (BFS)

- Visits graph vertices by moving across to all the neighbors of last visited vertex
- Instead of a stack, BFS uses a queue
- Similar to level-by-level tree traversal
- “Redraws” graph in tree-like fashion (with tree edges and cross edges for undirected graph)
Pseudocode of BFS

ALGORITHM BFS(G)

// Implements a breadth-first search traversal of a given graph
// Input: Graph G = (V, E)
// Output: Graph G with its vertices marked with consecutive integers
// in the order they have been visited by the BFS traversal
// mark each vertex in V with 0 as a mark of being “unvisited”

count ← 0
for each vertex v in V do
    if v is marked with 0
        bfs(v)

bfs(v)

// visits all the unvisited vertices connected to vertex v by a path
// and assigns them the numbers in the order they are visited
// via global variable count

count ← count + 1; mark v with count and initialize a queue with v
while the queue is not empty do
    for each vertex w in V adjacent to the front vertex do
        if w is marked with 0
            count ← count + 1; mark w with count
            add w to the queue
    remove the front vertex from the queue
Example of BFS traversal of undirected graph

BFS traversal queue:

BFS tree:
Notes on BFS

- BFS has same efficiency as DFS and can be implemented with graphs represented as:
  - adjacency matrices: $\Theta(V^2)$
  - adjacency lists: $\Theta(|V| + |E|)$

- Yields single ordering of vertices (order added/deleted from queue is the same)

- Applications: same as DFS, but can also find paths from a vertex to all other vertices with the smallest number of edges