CSCI 2270 Lecture Notes

4/3/2019

Breadth First Traverse

* given a starting vertex (think of this as the root), visit all neighboring vertices before moving on the next depth level (without repetition)

BFT implementation

struct vertex{

string key;

bool visited;

vector<adjVertex> adj;

}

Now the BFT algorithm:

void breadthFirstTraverse(keyStart)

find keyStart (can think of it as the root)

set vStart as “visited”

create queue (q)

Enqueue vStart onto q

loop until q is empty

n = q.dequeue (remember queues are FIFO)

loop across n’s adjacency list (x)

if !n->adj[x]->visited{

mark visited = true

disp(n->adj[x]->key

enqueue onto q

}

BFS (Breadth First Search)

updates the node struct to include the “distance”

struct vertex{

string key;

bool visited;

int distance;

vector<adjVertex> adj;

}

Now the BFS algorithm:

breadthFirstSearch(keyStart, searchKey)

find keyStart (can think of it as the root)

set vStart as “visited”

set vStart distance = 0;

create queue (q)

Enqueue vStart onto q

loop until q is empty

n = q.dequeue (remember queues are FIFO)

loop across n’s adjacency list (x)

if !n->adj[x]->visited{

n->adj[x].v->distance = n->distance+1;

if n->adj[x].v->key == searchKey

return n->adj[x].v;

else

mark visited = true

enqueue onto q

}

return null;