Graph Traversing II

1 Graph Traversing

Two main approaches for graph exploration:

- Breadth-First Search (BFS)
  - Search in **breadth**
  - layer by layer
- Depth-First Search
  - Search in **depth**
  - dig deeper, until not possible any more
- DFS is **not** appropriate for shortest paths
  - We may reach the target vertex b via a **very long** path, as we just ”dig deeper”
- Both BFS and DFS
  - Appropriate for graph exploration
  - Can list all reachable vertices from a start vertex a
  - very fast (linear time)

2 A generic search algorithm

Listing 1: Generic Graph Search

```
1 reachable[a]=1                                                    //Initialisation
2 S={a}          //initialisation; set of vertices, from which we continue exploration
3 for i=1 to n                                 //Iterate until no more reachable vertices
4     choose a vertex u ∈ S                                //the crucial choice of the search
5     print u                                  // u is the next vertex in the output list
6     for each vertex v ∈ Adj[u]
7         if reachable[v]==0 then                        //We found a new vertex v to reach
8             reachable[v]=1                                       //"mark" v as reached
9             S = S ∪ {v}                       //add v to the list of reachable vertices
```

- The set S changed dynamically
- BFS and DFS have different ”policy” for the choice at line 4
- BFS prefers vertices ”closer to a”
- DFS prefers vertices that are always ”one step further”

3 The policy of BFS

- The policy of BFS
  - Remove the element that has been **longer in S**
  - a First-In-First-Out (FIFO) policy
- This data structure is called a queue
- In other words
  - Add new vertices at the end of the queue
  - remove vertices from the beginning of the queue
  - first process vertices that are closer to the start vertex

4 The policy of DFS

- The policy of DFS:
  - Remove the element that has been shorter S
  - a Last-In-First-Out (LIFO) policy
- This data structure is called a ”stack”
- In other words:
  - add new vertices at the end (top) of the stack
  - remove vertices from the end(top) of the stack
  - first process vertices that always ”one step further”

5 Longest paths

Computing a Longest path is NP complete

Intuitively:

- vertices are balls
- edges are strings tight on the balls

Shortest path problem:

- Pull firmly two specific balls away from each
- the length of the string between then is their distance in the graph

Longest path problem:

- You need to investigate all (possibly ”strange”) paths between the two balls through the net of strings
- which can be very complex

However:

- If the graph has no cycles, then it is easy
- such graphs are called ”trees”