Depth-First Search & Breadth-First Search

Otto Kuosmanen, Martin Winet

EL2310 Scientific Programming

ottoku@kth.se, winet@kth.se

September 24, 2015

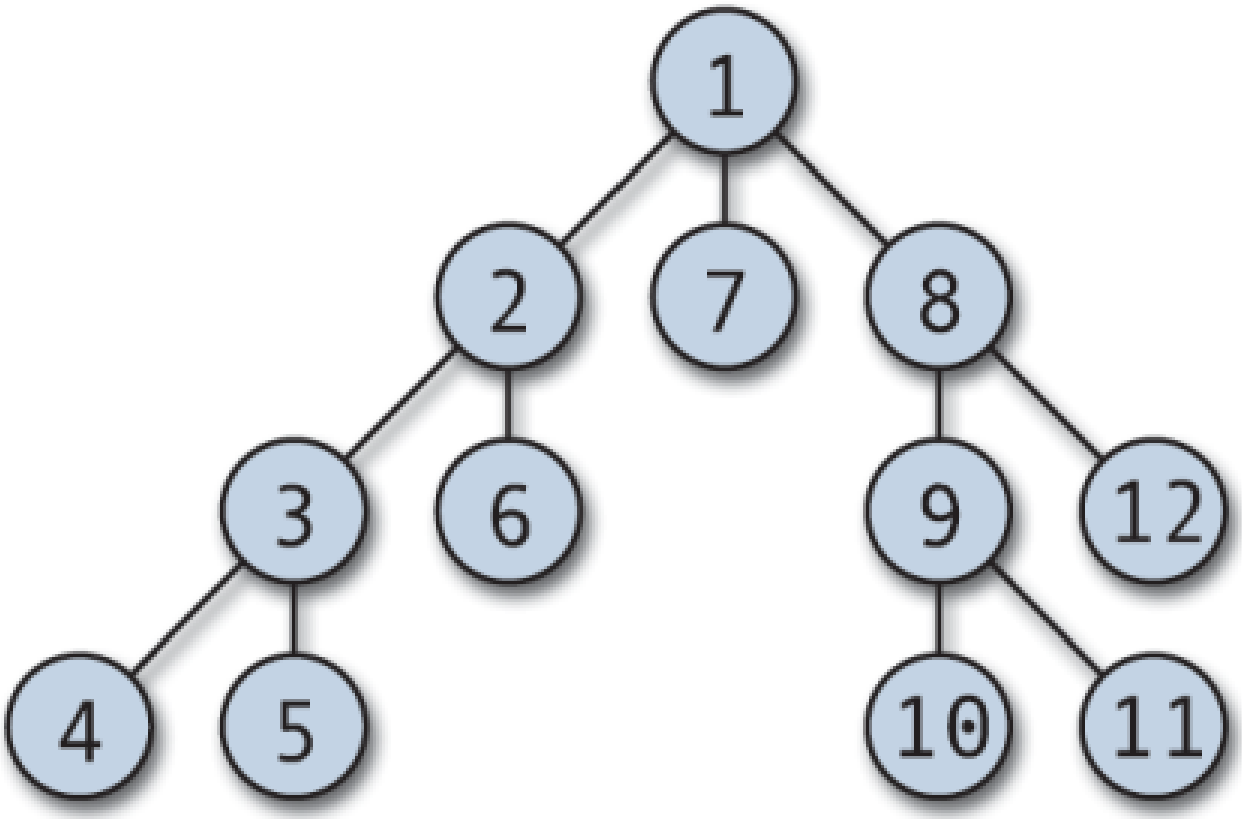
Applications

- Perform searches within graphs
- Perform operations on graphs
- Check connections and accessibilities
- Route planning (find shortest path)
- ...

Prerequisites

- Data available in a tree structure (graph)
- Tree structure expressed as:
  - Adjacency list (blackboard)
  - Adjacency matrix (blackboard)

DFS Principle



Starting from the root vertex explores as far as possible along each branch before backtracking.

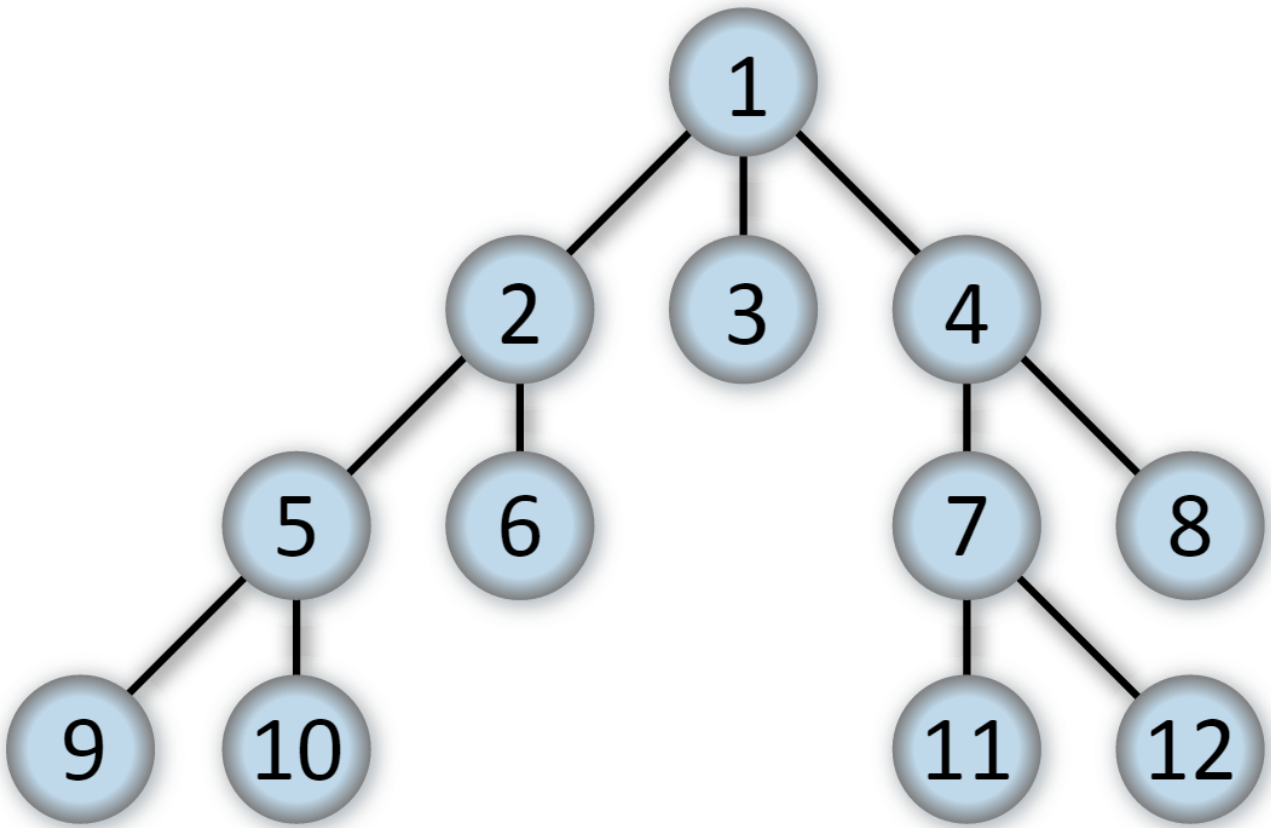
DFS Sample Code

```
int main()
{
    DFS(0);                          // call initial vertex
}

void DFS(int i)
{
    int j;
    printf("%d\n",i+1);                  // print current vertex
    visited[i]=1;                    // mark vertex as visited

    for(j=0;j<n;j++)                 // loop through the neighbours of the
          current vertex
       if(!visited[j]&&G[i][j]==1)
            DFS(j);                  // jump to the first not visited neighbour
}
```

BFS Principle



Starting from the root vertex explores neighbour vertices first before moving to the next level of neighbours.

BFS Sample Code

```
int main()
{
    while(front<12)
    {
        u = queue[front];
        printf("%d\n",u+1);              //print current vertex
        front = front+1;                 //remove first
        nnei = 0;
        for(i=0;i<12;i++)
        {
            if(G[u][i]==1)               //find neighbours of u
            {
                neighbours[nnei] = i;    //save the index in G
                nnei++;
            }
        }

        for(i=0;i<nnei;i++)              //visit each neighbour
        {
            n = neighbours[i];
            if(visited[n] == 0)          //if not visited
            {
                visited[n] = 1;
                rear = rear +1;
                queue[rear] = n;         //add to the end
            }
        }

    }
}
```

Differences between the algorithms

- Memory use
  - DFS: $O(V)$ (stack)
  - BFS: $O(V)$ (queue)
- Runtime
  - DFS: $O(V + E)$ adjacency list, $O(V^2)$ adjacency matrix
  - BFS: $O(V + E)$ adjacency list, $O(V^2)$ adjacency matrix
- Optimality?

What else?

- Iterative deepening depth-first, depth-limited search
- Dijkstra's algorithm (BFS as a special case)

?