Østfold University College

Graph Search Algorithms

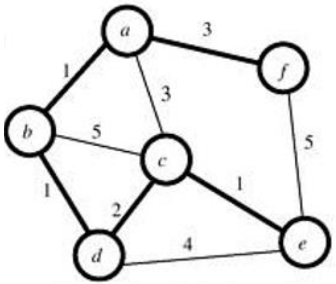
Lecture 8

Contents

- Brute-Force Algorithm
- Breadth-First Search (BFS) Algorithms
- Depth-First Search (DFS) Algorithms (Backtracking Algorithm)
- Parallel BFS and DFS

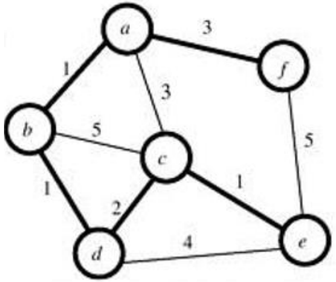
Brute-Force Algorithm

Minimum Spanning Tree

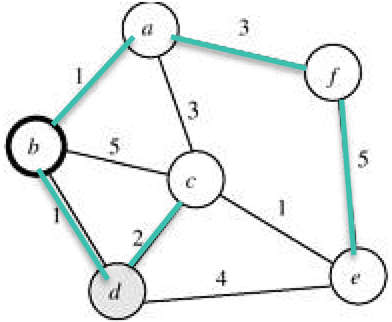


Prim’s algorithm

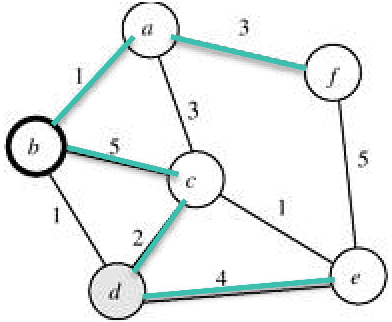
Minimum Spanning Tree



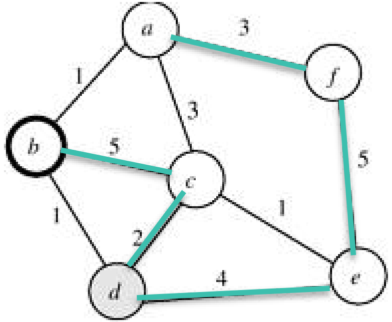
Prim's algorithm
Cost = 8



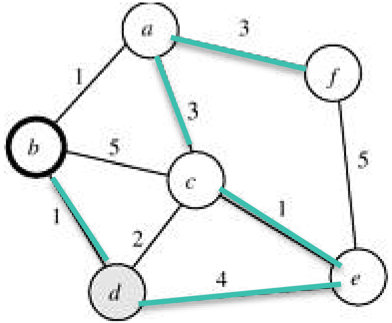
Cost = 13



Cost = 20



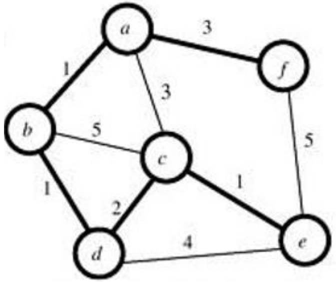
Cost = 19



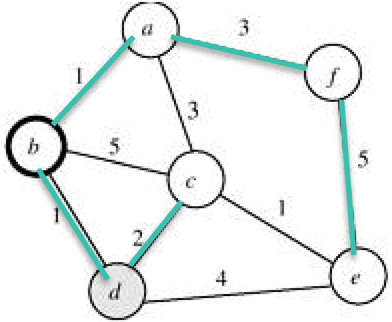
Cost = 14

...

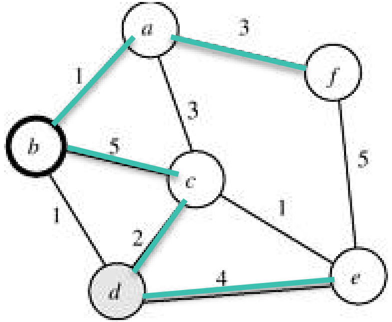
Minimum Spanning Tree



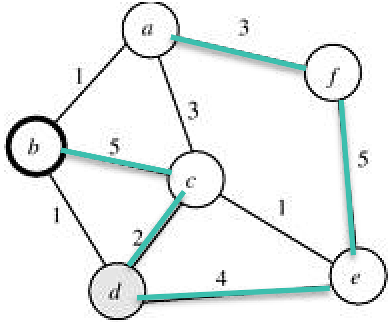
Prim’s algorithm
Cost = 8



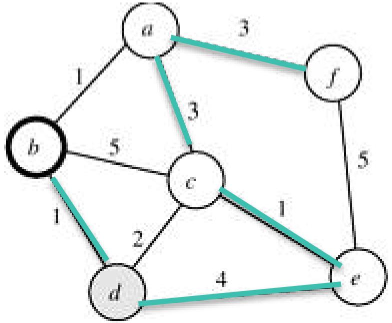
Cost = 12



Cost = 15



Cost = 19

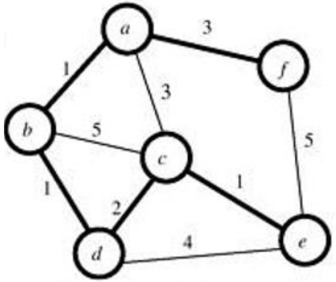


Cost = 14

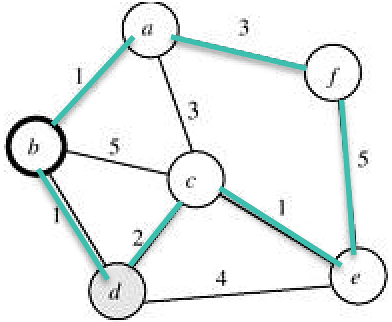
...

- Find all possible path which goes through all vertices in the graph
- Select the one with minimum cost

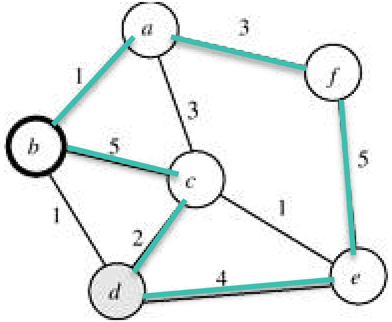
Minimum Spanning Tree



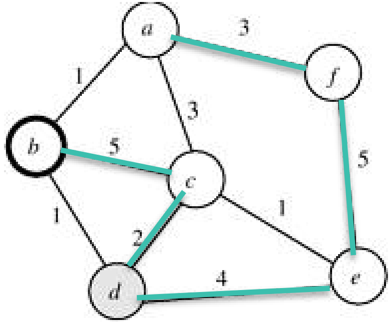
Prim's algorithm
Cost = 8



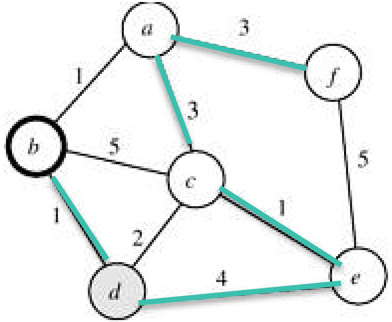
Cost = 13



Cost = 20



Cost = 19

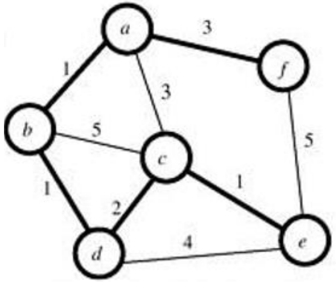


Cost = 14

...

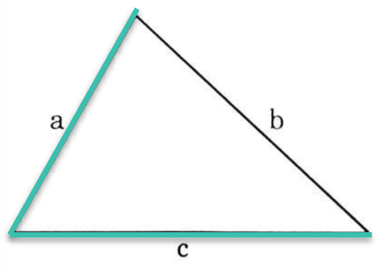
So many cases!!!

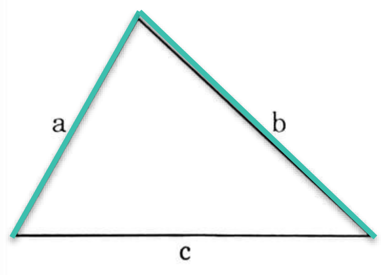
Minimum Spanning Tree

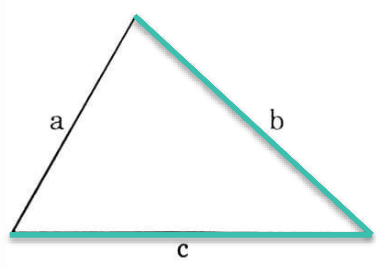


Prim's algorithm

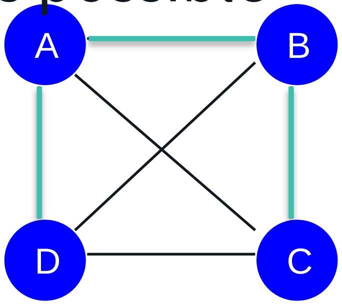
But if the number of vertices is limited, then listing all cases is possible

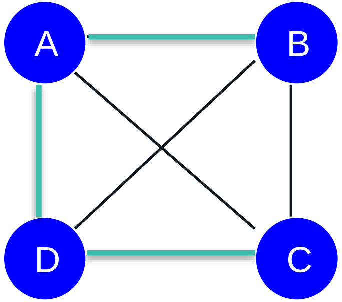


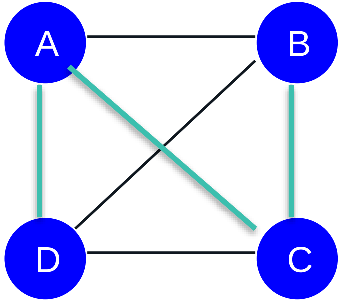


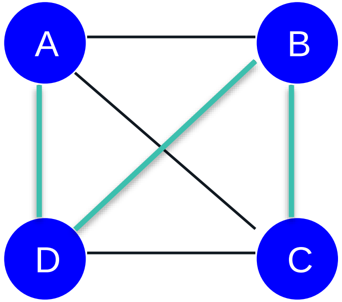


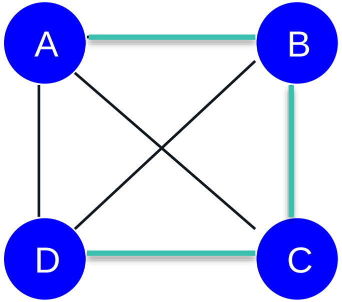
But if the number of vertices is limited, then listing all cases is possible

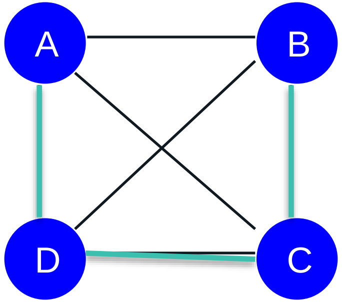












Østfold University College

Another example of Brute-Force solution

- There is a lock of 4-digit PIN. Each digit is chosen from 0-9
- You forgot the PIN and have to find the PIN
- The Brute-Force will try all possible combinations one by one: 0000, 0001, 0002,0003….
- The number of test cases: **10000**

4-digit PIN lock

6-digit PIN lock

Brute-Force Algorithm

- Exhaustive Search in entire searching space
- Search for an element with a special property until either a match is met, or the list is exhausted without finding a match
- Method
  - Generate a list of all protentional solutions to a problem
  - Evaluate each solution
  - When search ends, announce the solution

Brute-Force Algorithm - Pros

- Straightforward, usually directly based on the problem statement
- Simple
- Widely applicable (searching, sorting,…)

Østfold University College

Brute-Force Algorithm - Cons

- Exhaustive-search algorithms run in a reasonable amount of time only on very small instances, sometimes are unacceptable slow
- Usually not efficient
- Not as constructive as some other techniques
  - MST: Prim's algorithm
  - Dijkstra algorithm…
- But, in many cases, exhaustive search or its variation is the only known way to get exact solution

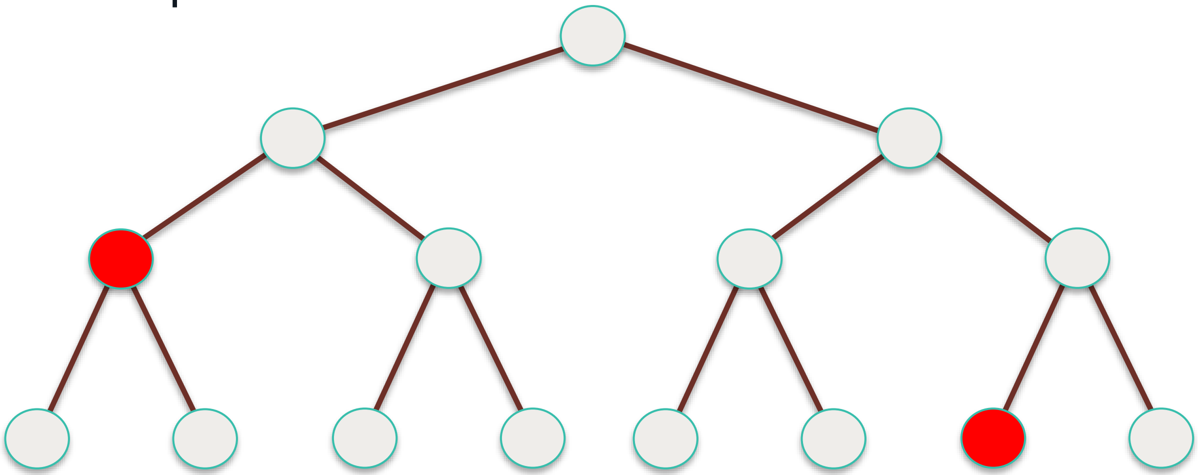
Breadth-First Search (BSF) Algorithms

Searching Algorithm

- Search algorithms can be used to solve discrete optimization problems (DOPs)
  - A class of computationally expensive problems with significant theoretical and practical interest
- Search algorithms systematically search the space of possible solutions subject to constraints

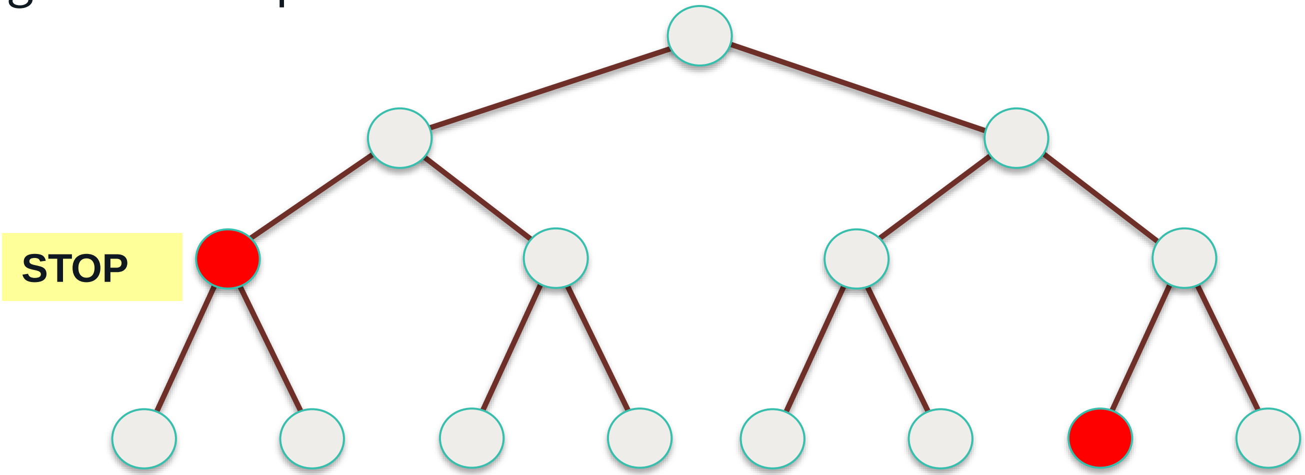
Searching Algorithm

- Look for a given node
  - Stop when node found, even if not all nodes were visited
  - Not guarantee optimization



Searching Algorithm

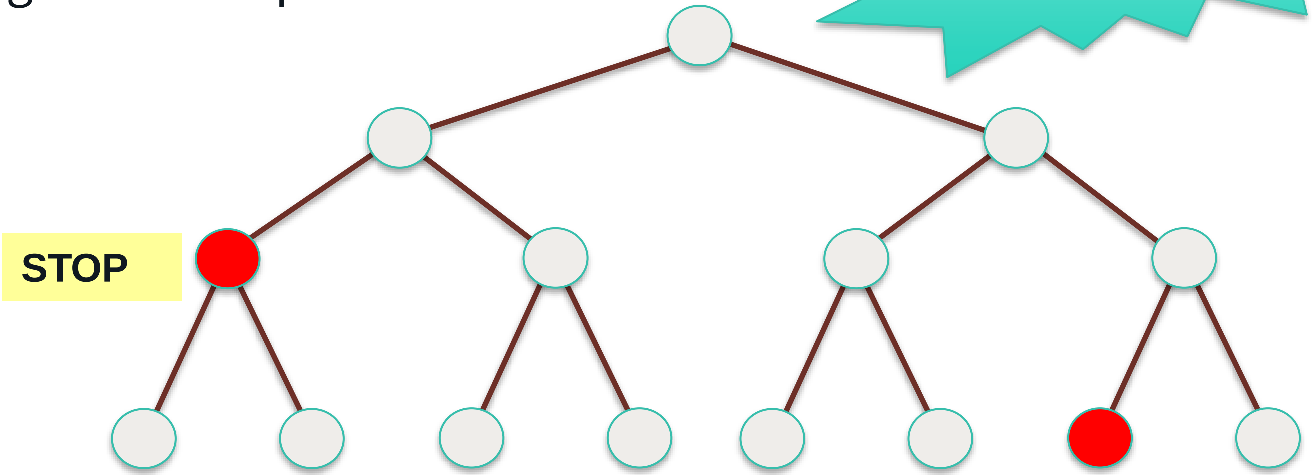
- Look for a given node
  - Stop when node found, even if not all nodes were visited
  - Not guarantee optimization



Searching Algorithm

- Look for a given node
  - Stop when node found, even if not all nodes
  - Not guarantee optimization

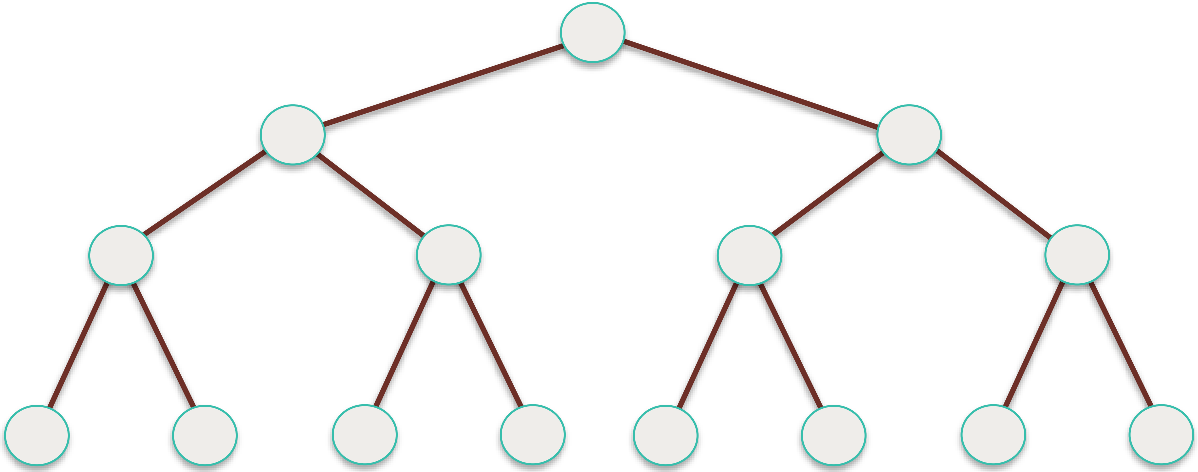
If solutions found while searching, STOP!



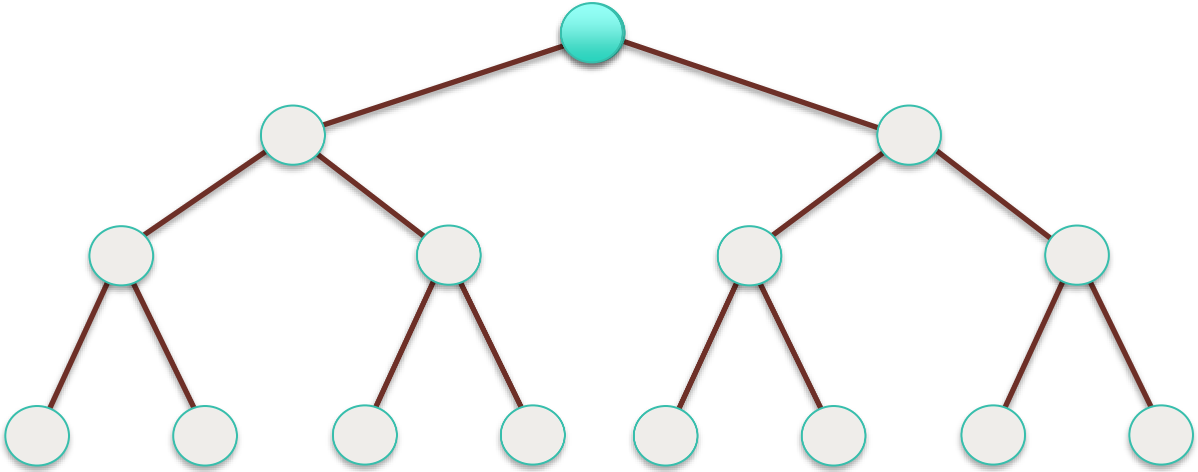
What it is and why

- Look for a given node
  - Stop when node found, even if not all nodes were visited
  - Not guarantee optimization

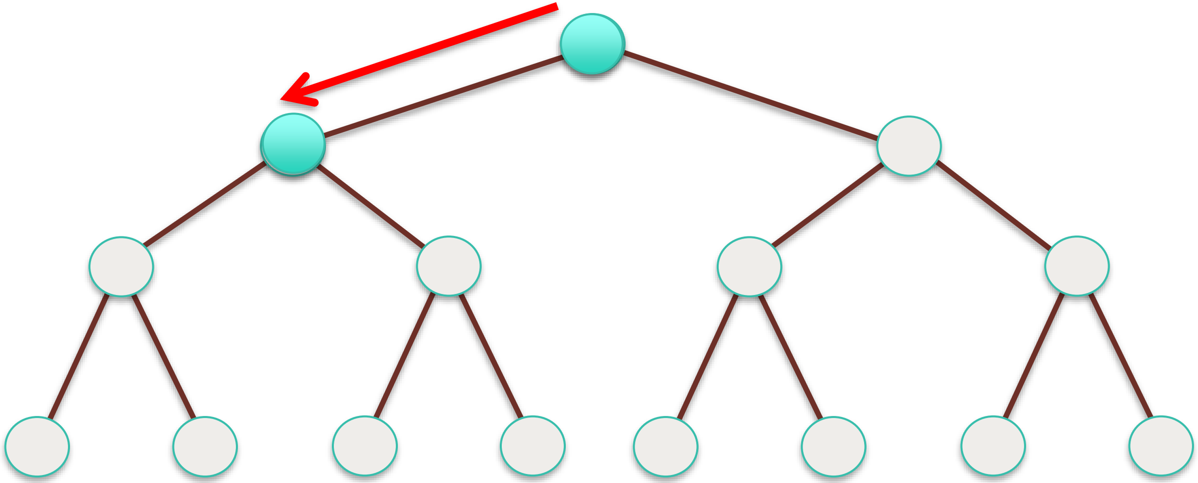
Breadth-First Search (BFS): Tree



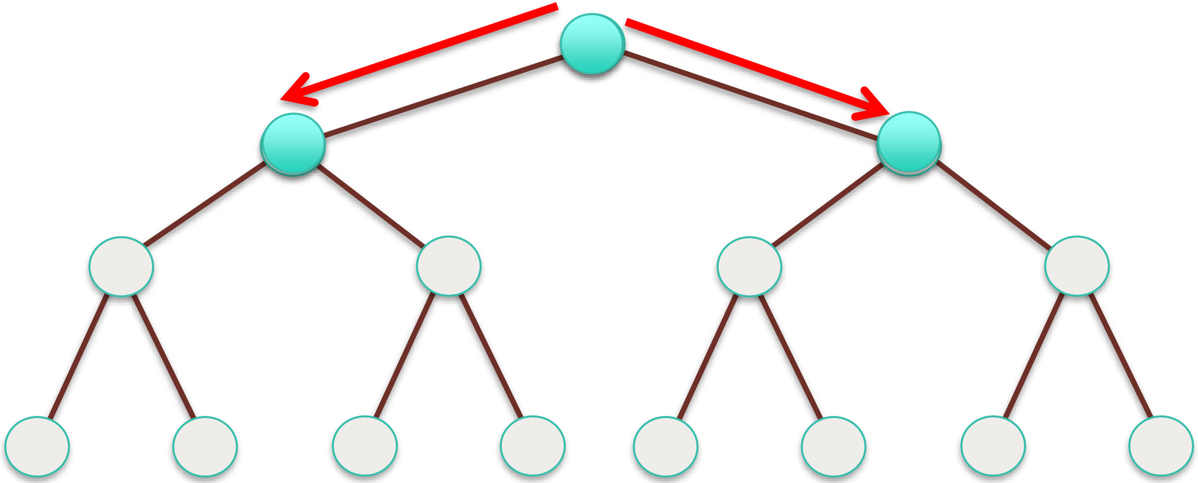
Breadth-First Search (BFS): Tree



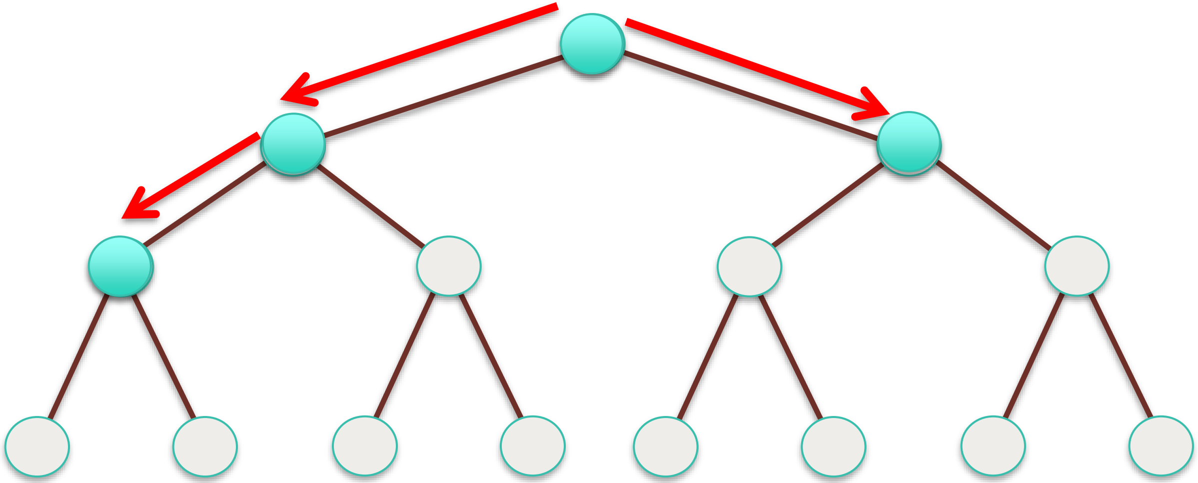
Breadth-First Search (BFS): Tree



Breadth-First Search (BFS): Tree

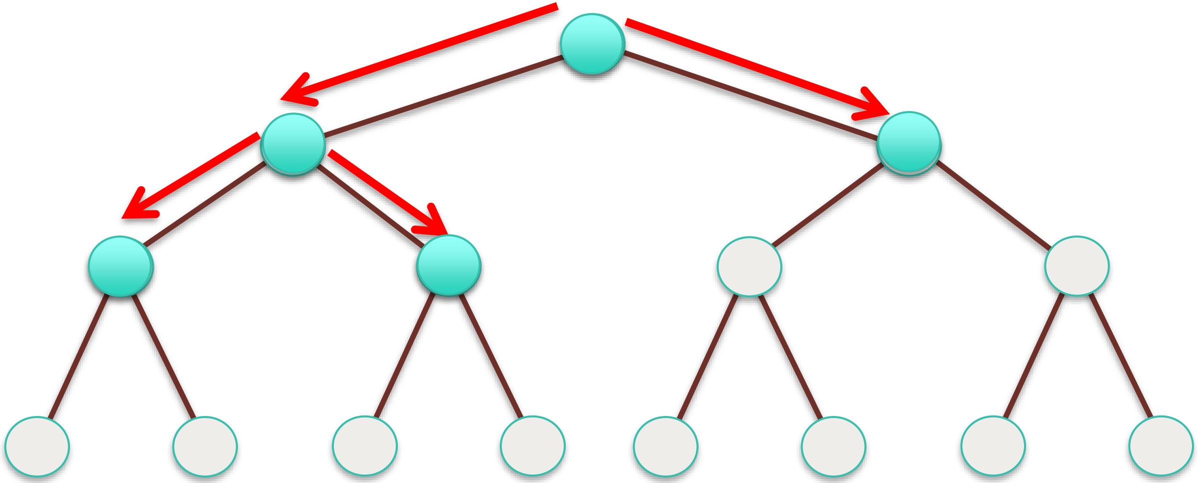


Breadth-First Search (BFS): Tree

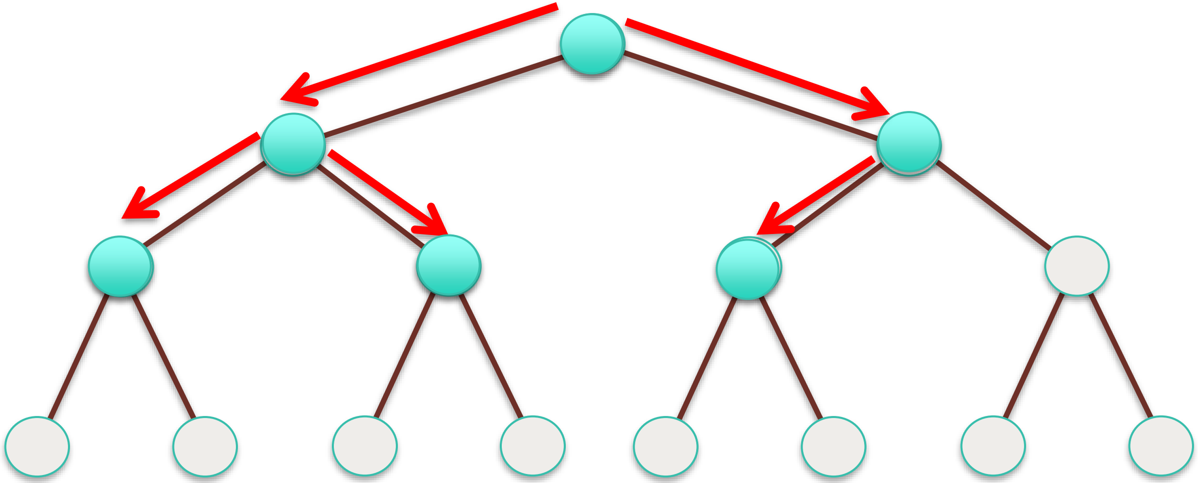


Østfold University College

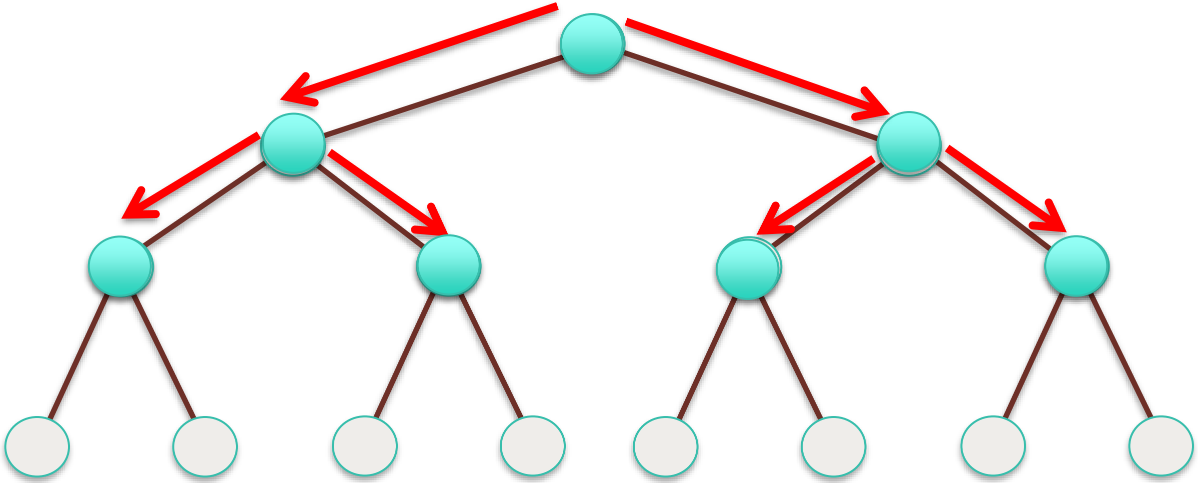
Breadth-First Search (BFS): Tree



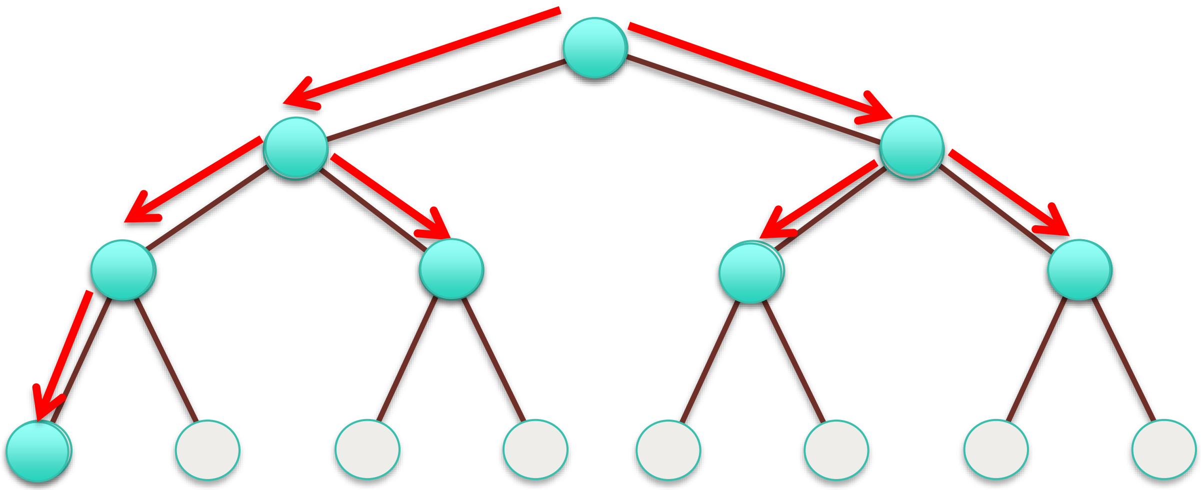
Breadth-First Search (BFS): Tree



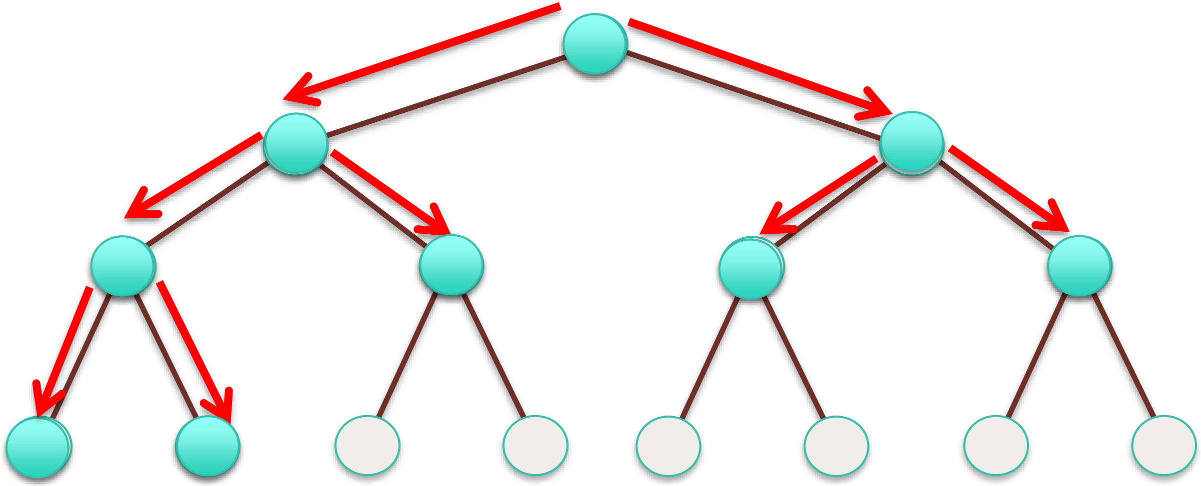
Breadth-First Search (BFS): Tree



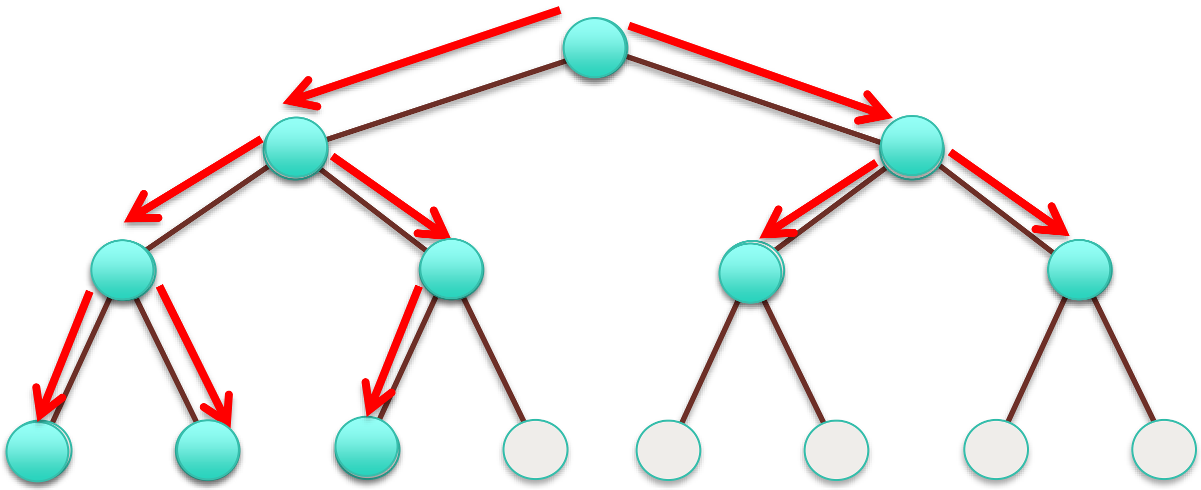
Breadth-First Search (BFS): Tree



Breadth-First Search (BFS): Tree

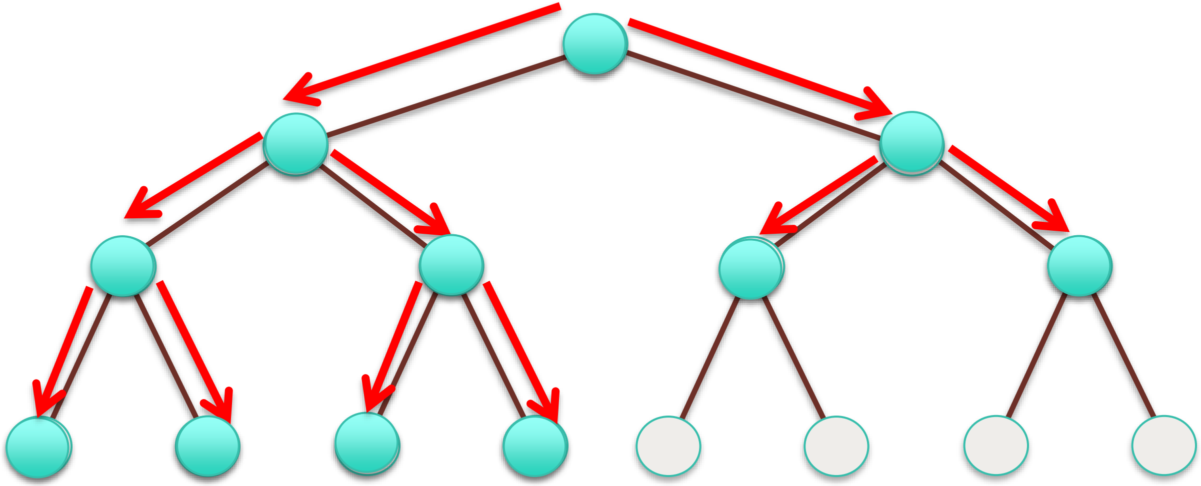


Breadth-First Search (BFS): Tree

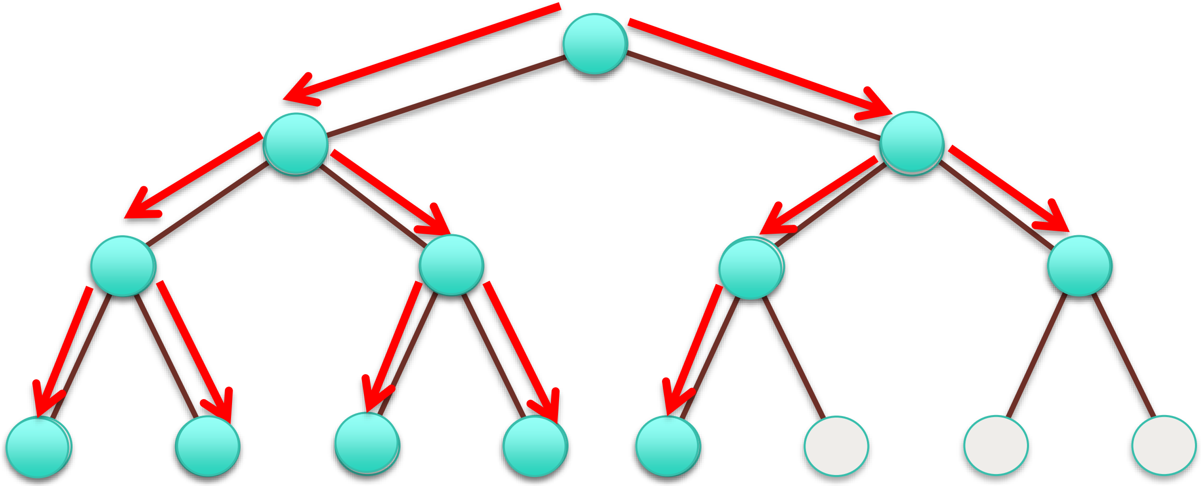


Østfold University College

Breadth-First Search (BFS): Tree

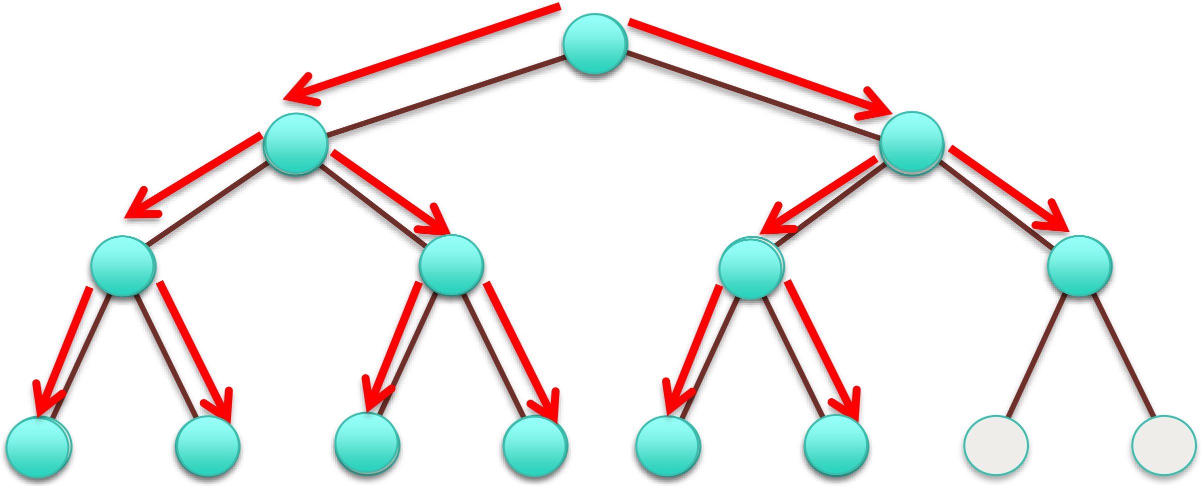


Breadth-First Search (BFS): Tree

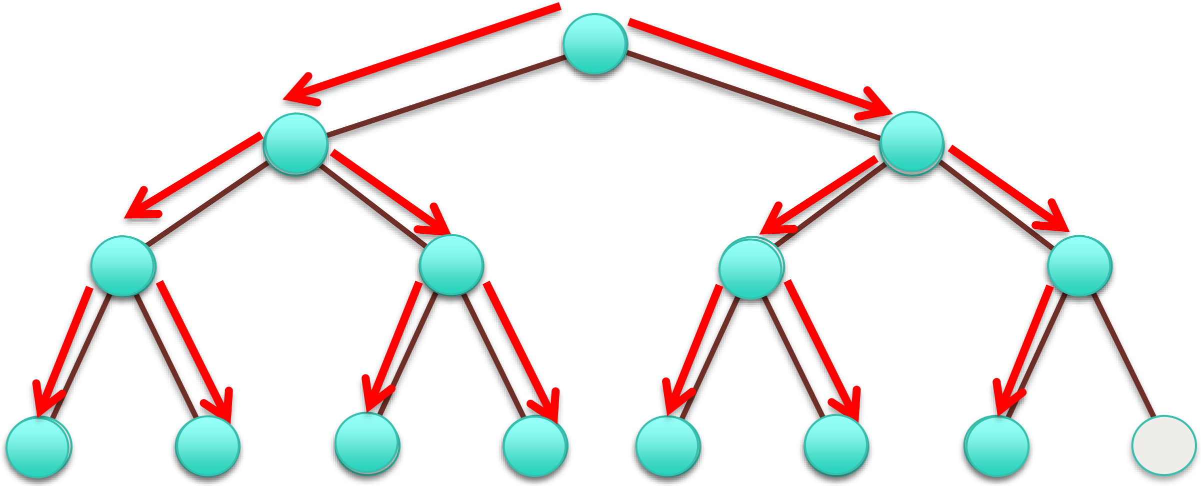


Østfold University College

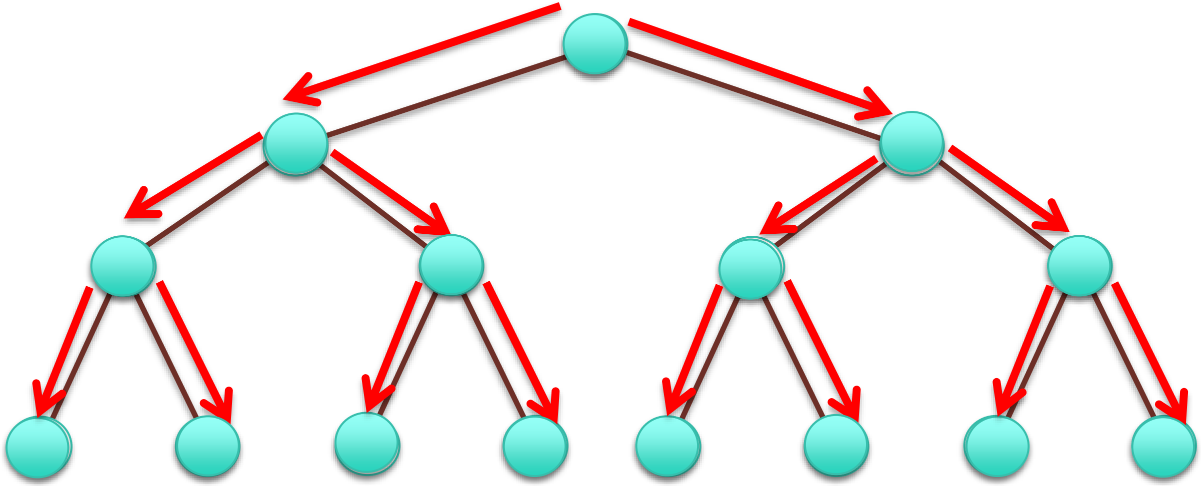
Breadth-First Search (BFS): Tree



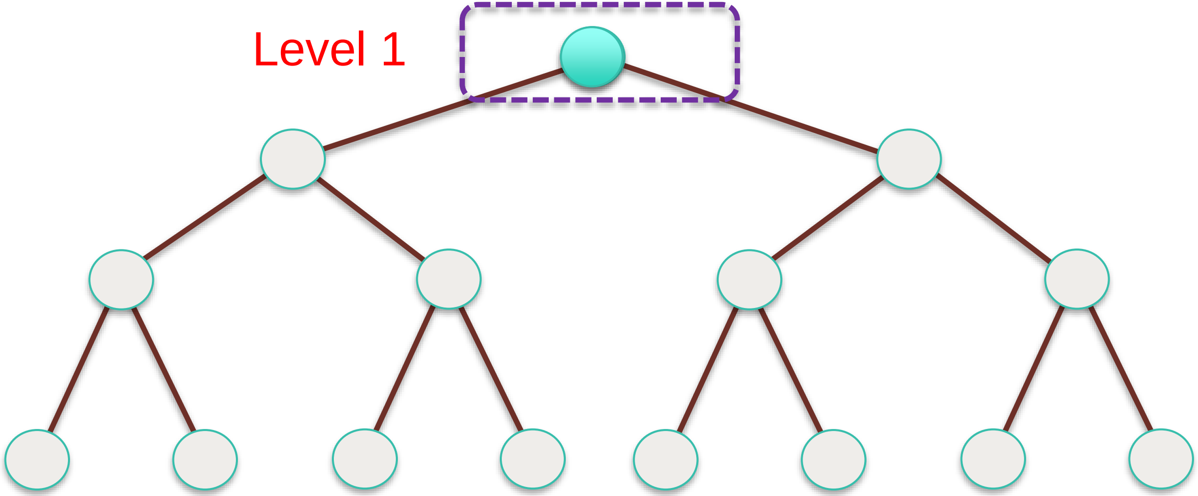
Breadth-First Search (BFS): Tree



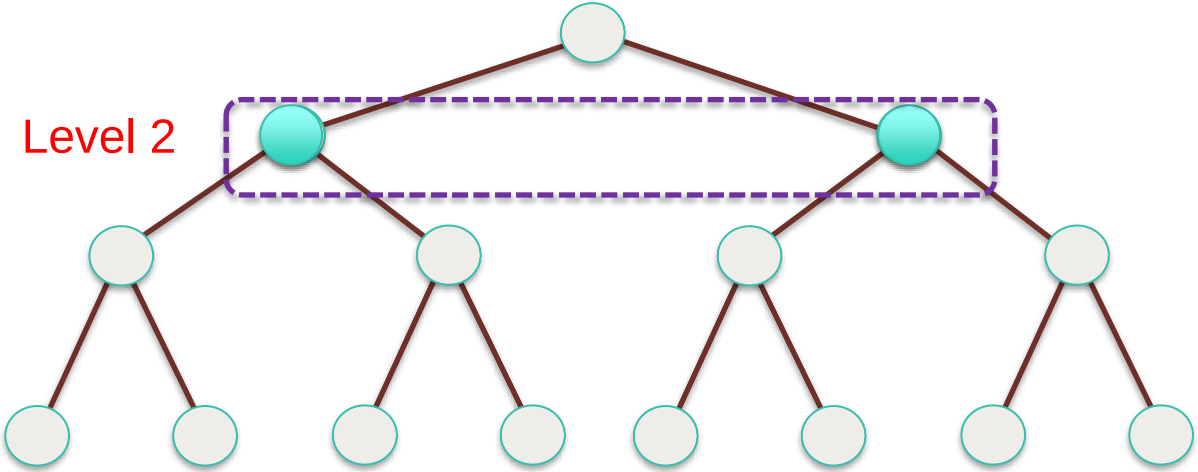
Breadth-First Search (BFS): Tree



Breadth-First Search (BFS): Tree

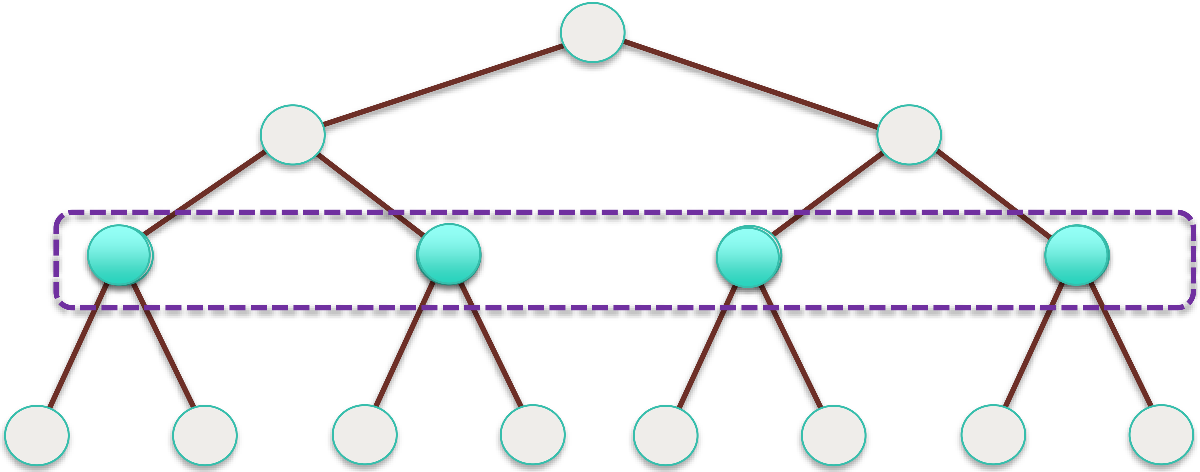


Breadth-First Search (BFS): Tree



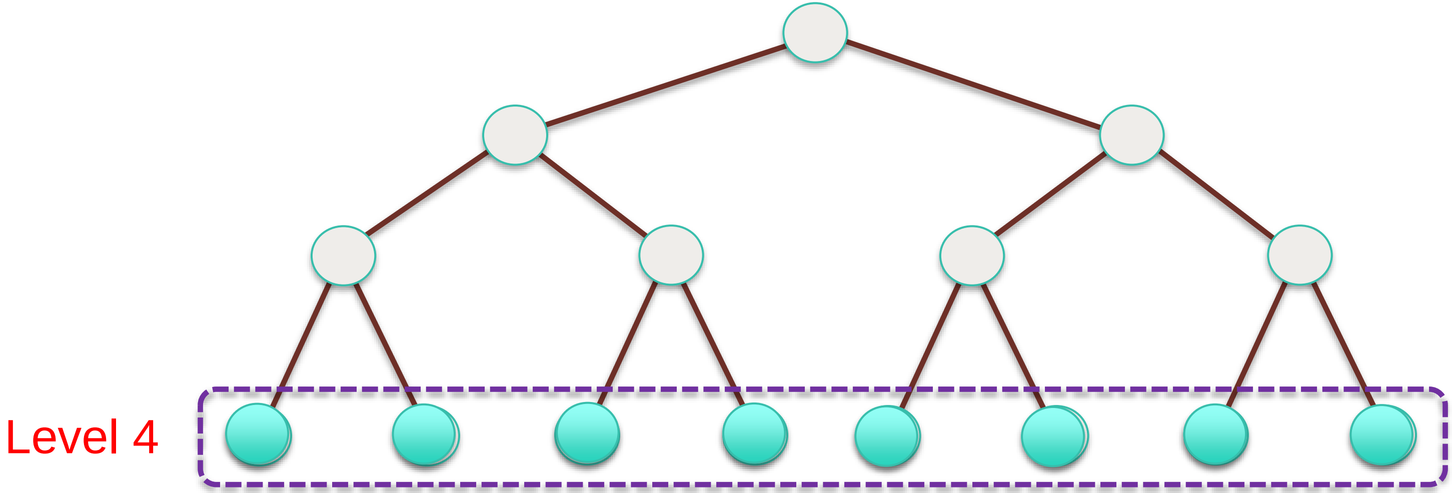
Breadth-First Search (BFS): Tree

Level 3



Østfold University College

Breadth-First Search (BFS): Tree



Breadth- First Search (BFS)

- Used to search in undirected graph
- Start with a node and then visit all its adjacent nodes in the same level
- Move to the adjacent successor node in the next level
- Level-by-level search

Breadth- First Search (BFS) Steps

- Step 1: Start with the root node (any node in the graph), mark it visited
- Step 2: If that node has no node with same level, go to the next level
- Step 3: Visit all adjacent nodes and marked them visited
- Step 4: Go to the next level and visit all unvisited adjacent nodes
- Step 5: Continue until all nodes visited or solution found

Østfold University College

Notes

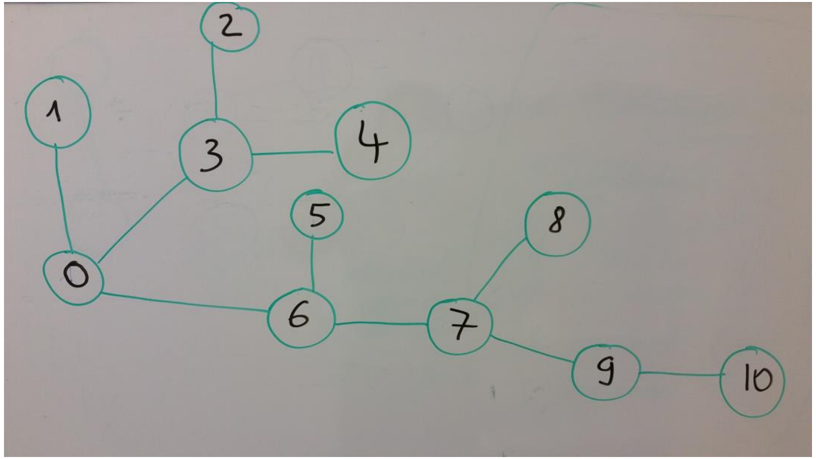
- Breadth- First Search (BFS) is different from Best-First Search (BFS) algorithm in the textbook
- Best-First Search is an algorithm that traverses a graph to reach a target in the shortest possible path.
- Best-First Search follows an evaluation function to determine which node is the most appropriate to traverse next
- But we don't look at Best-First Search (BFS)

BSF Pseudo-code (serial)

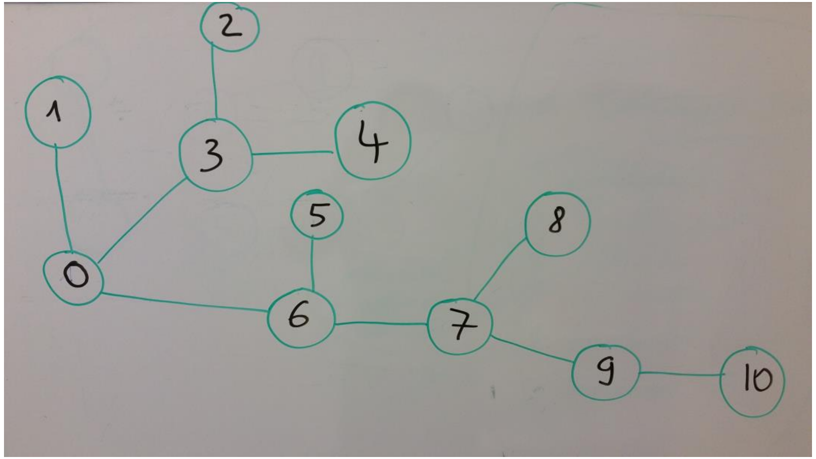
```
Input: G(V,E,w)
Output: List of visited nodes in G in ordered

Queue Q = { };
for a node u,
        set visited[u] = false;
        insert Q, u;
while ((Q ⊄∅) || ( u is not the target node)) // Q is not empty or u is not what we are searching for
        delete u from Q
        if (visited[u] = false)
              for each adjacent v of u then
                     insert Q, v;
       endif
endwhile
```

Sample ‘Implementation’



Sample 'Implementation'

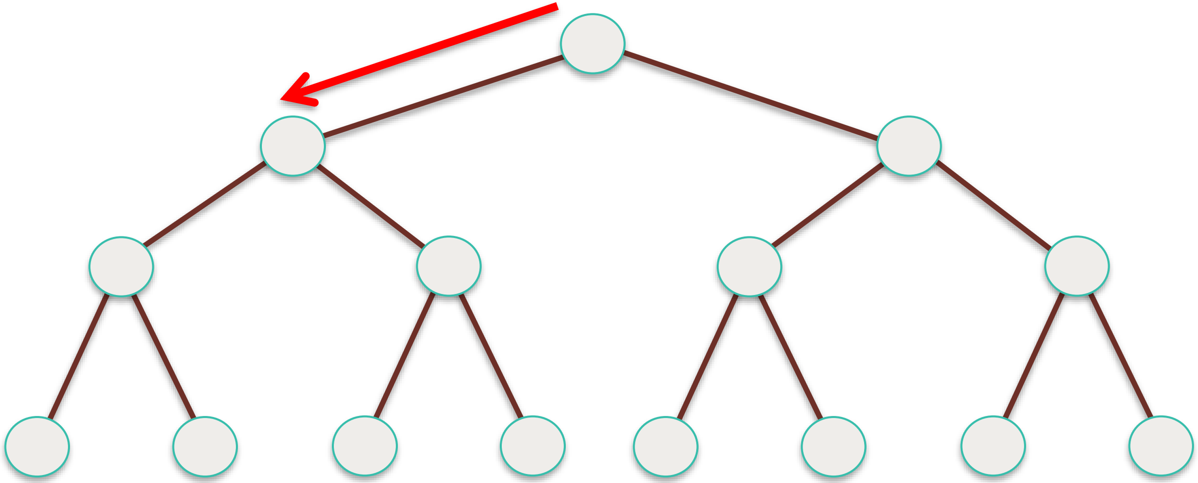


Data structure first: adjacent matrix or adjacent list ?
Start vertex: 0 (but can be any other vertex)
Data Structure: Queue (First-in-First-Out)
Q = ?
Step 1: Q = {0}
Step 2: Q = {1,3,6}
Step 3: Q = {3,6}
Step 4: Q = {6,2,4}
Step 5: Q = {6,2,4}
Step 6: Q = {2,4,5,7}
Step 7: Q= {4,5,7}
Step 9: Q= {5,7}
Step 10: Q= {7}
Step 11: Q= {8,9}
Step 12: Q= {9}
Step 13: Q= {10}
Step 14: Q= {∅} **STOP**

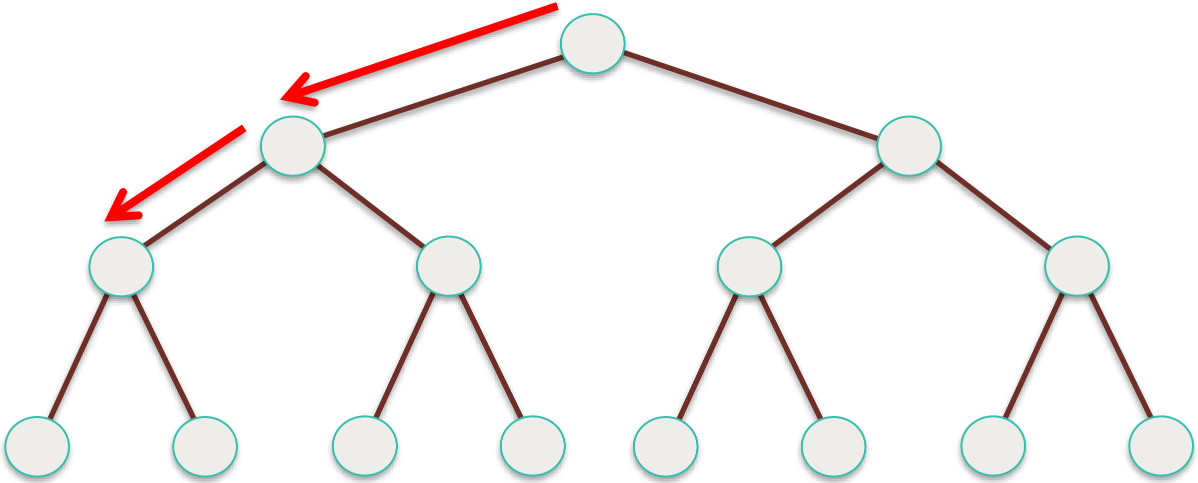
Depth-First Search (DSF) Algorithms

Østfold University College

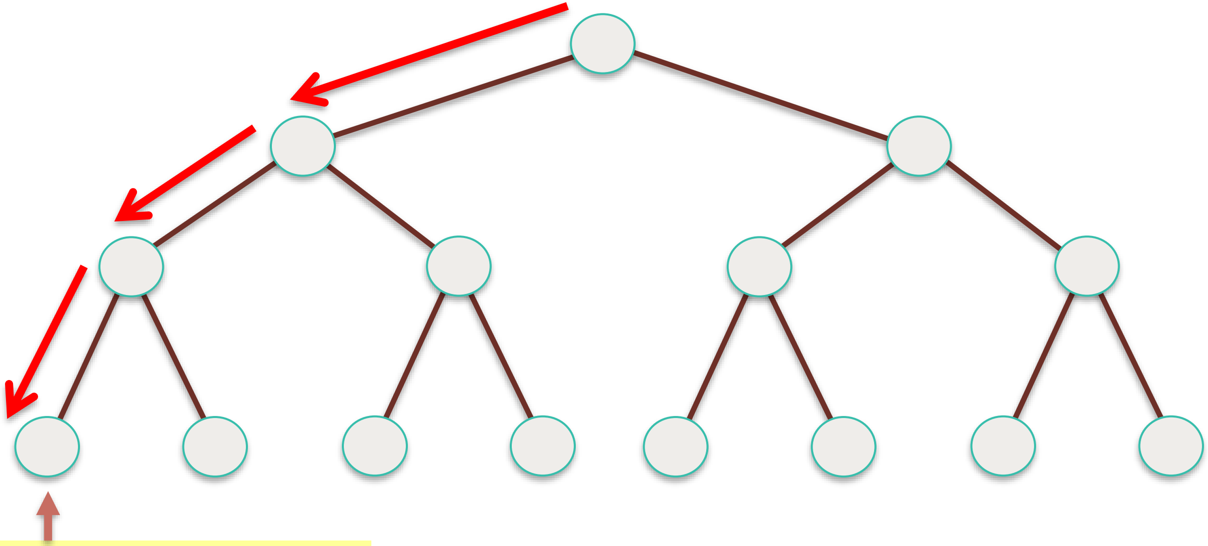
Depth-First Search (DFS): Tree



Depth-First Search (DFS): Tree

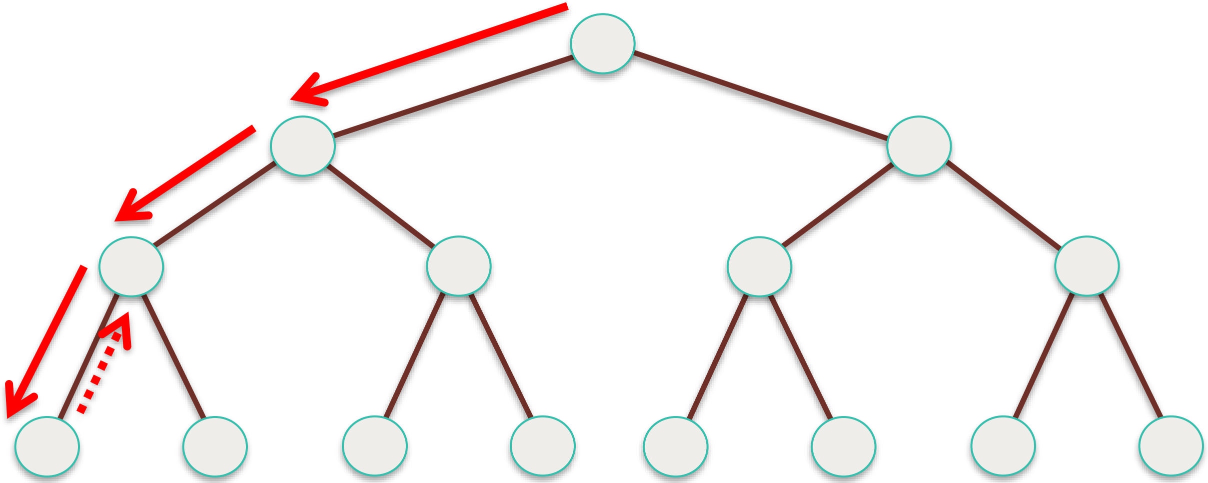


Depth-First Search (DFS): Tree

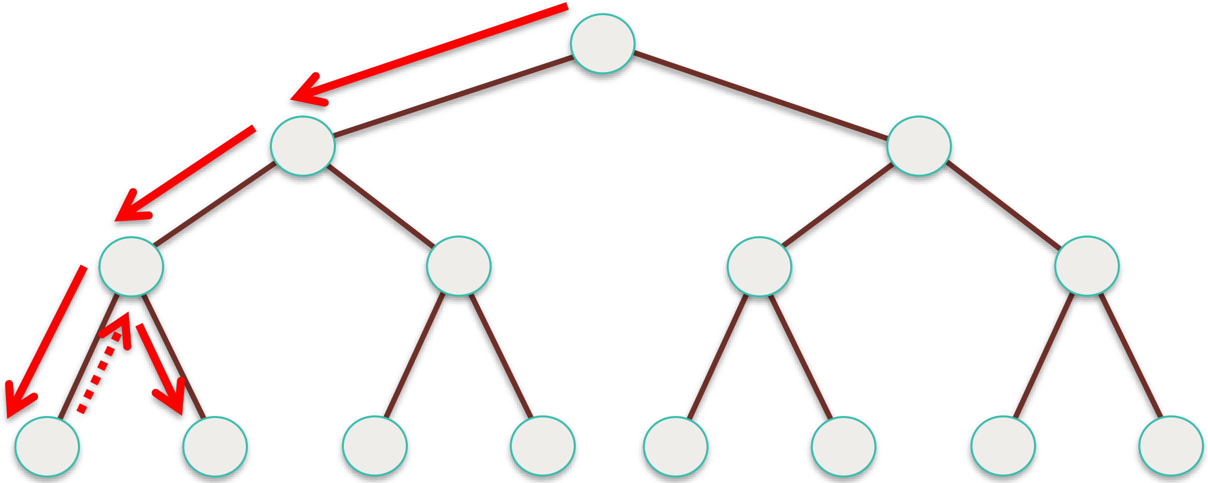


No successor, backtracking!

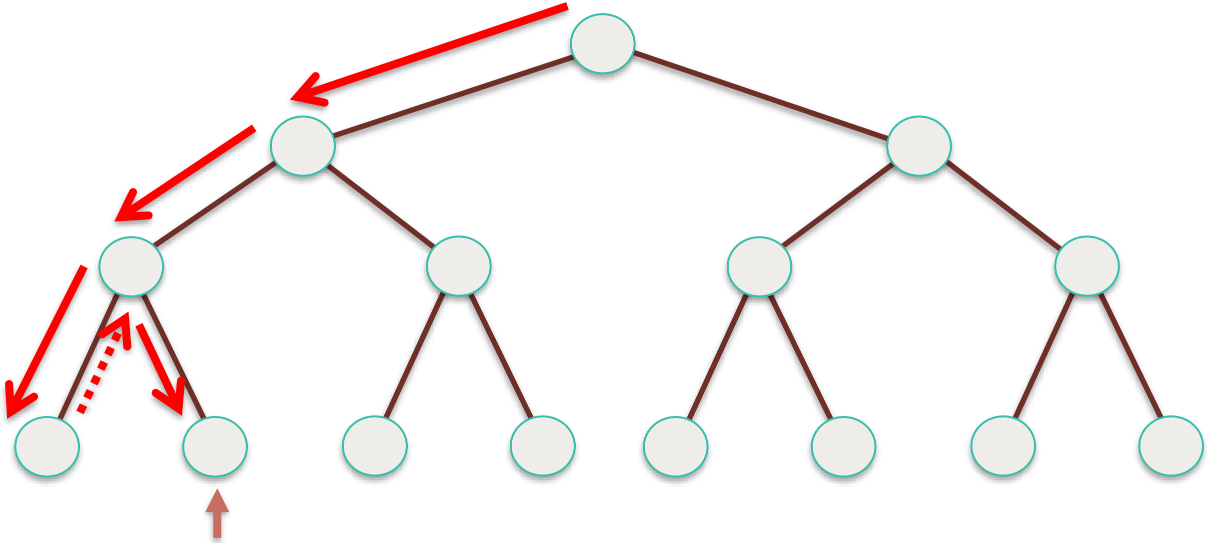
Depth-First Search (DFS): Tree



Depth-First Search (DFS): Tree

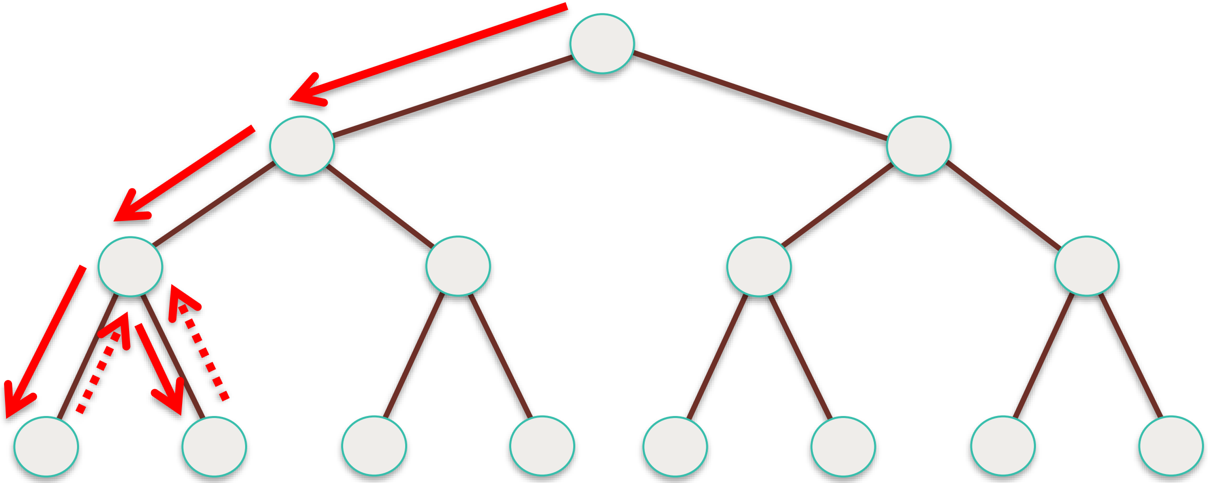


Depth-First Search (DFS): Tree

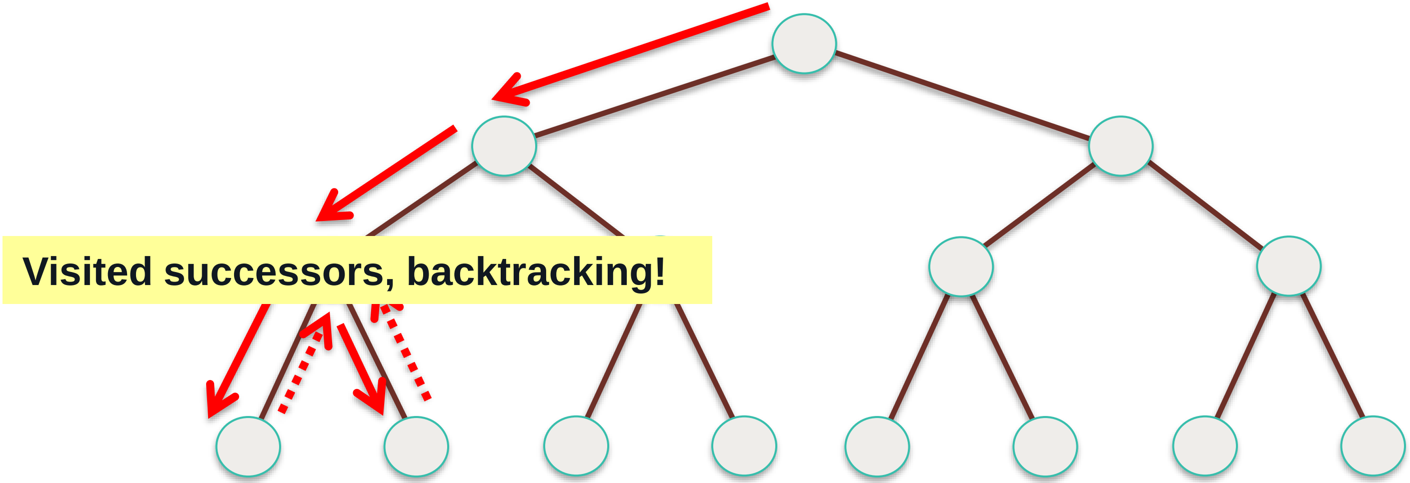


No successor, backtracking!

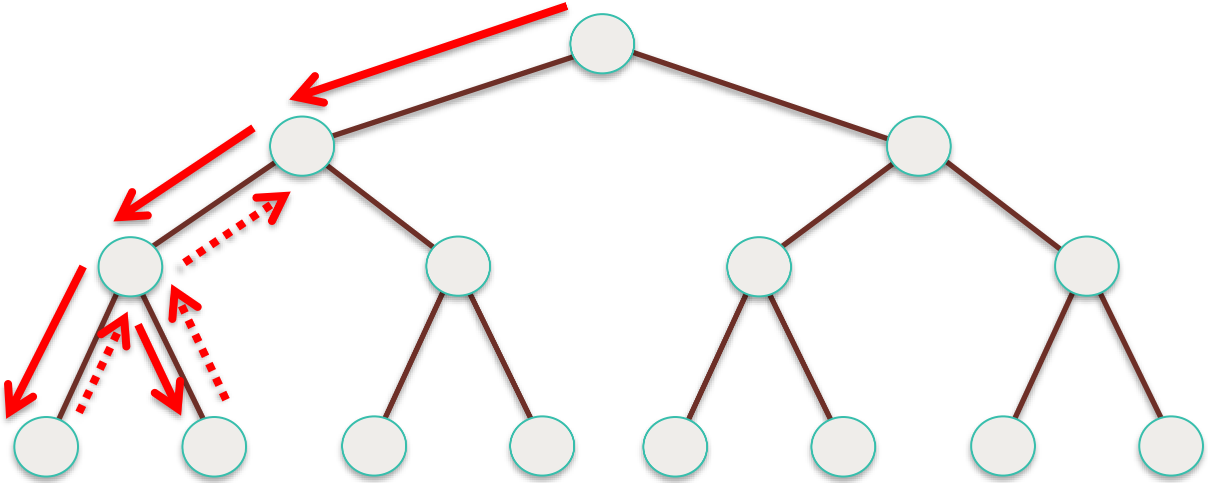
Depth-First Search (DFS): Tree



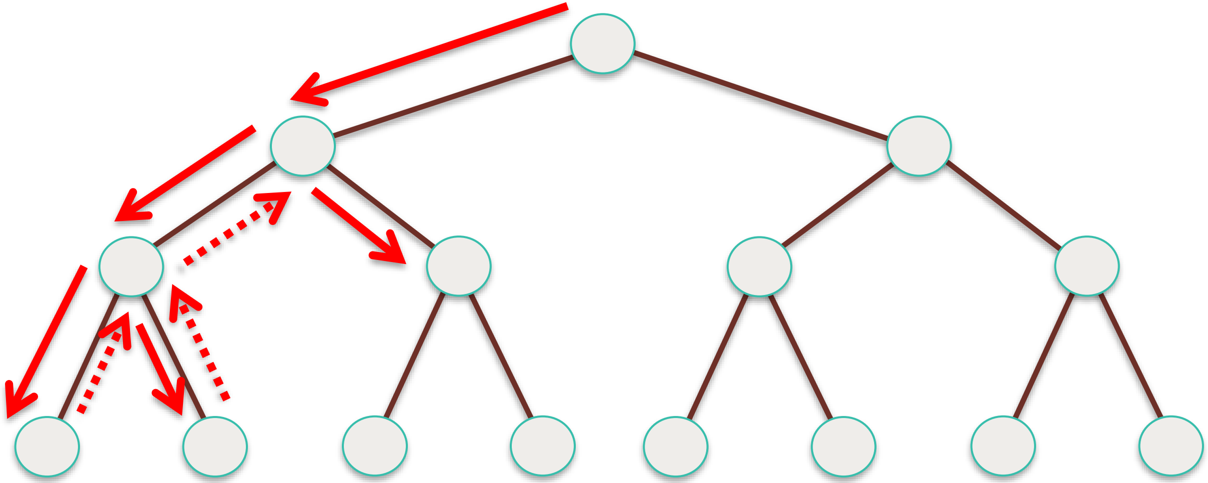
Depth-First Search (DFS): Tree



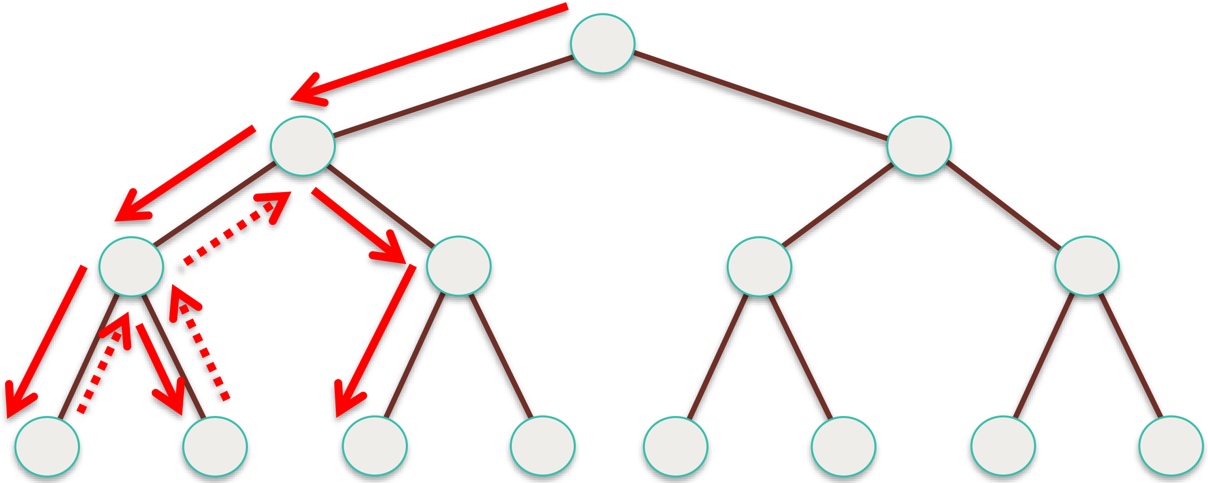
Depth-First Search (DFS): Tree



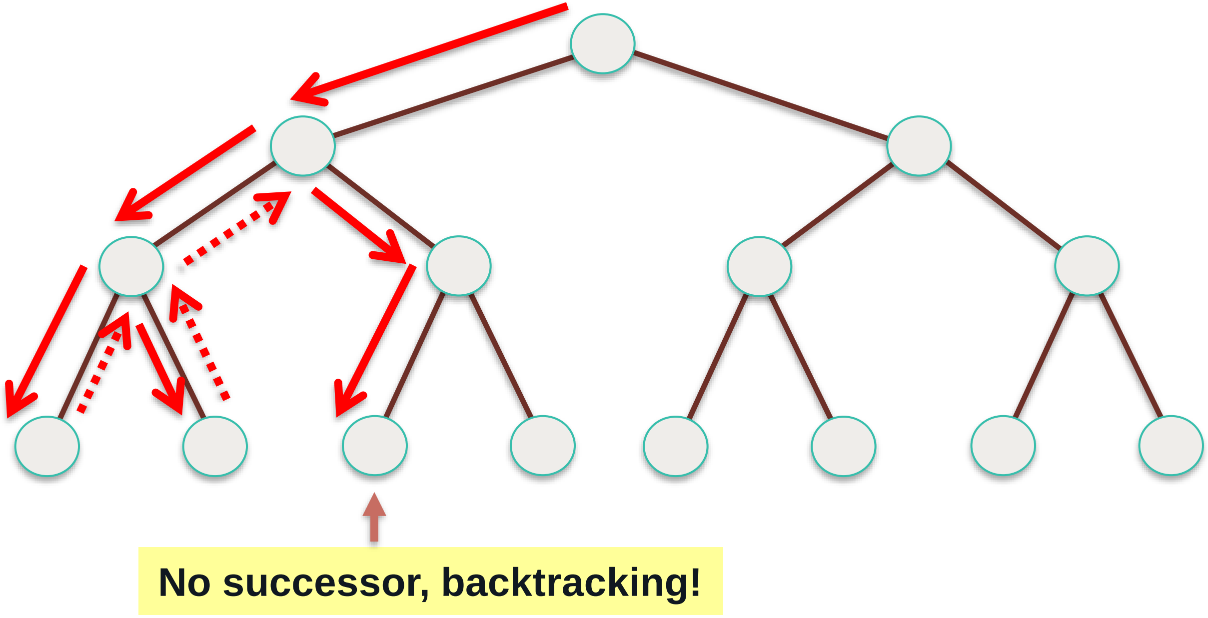
Depth-First Search (DFS): Tree



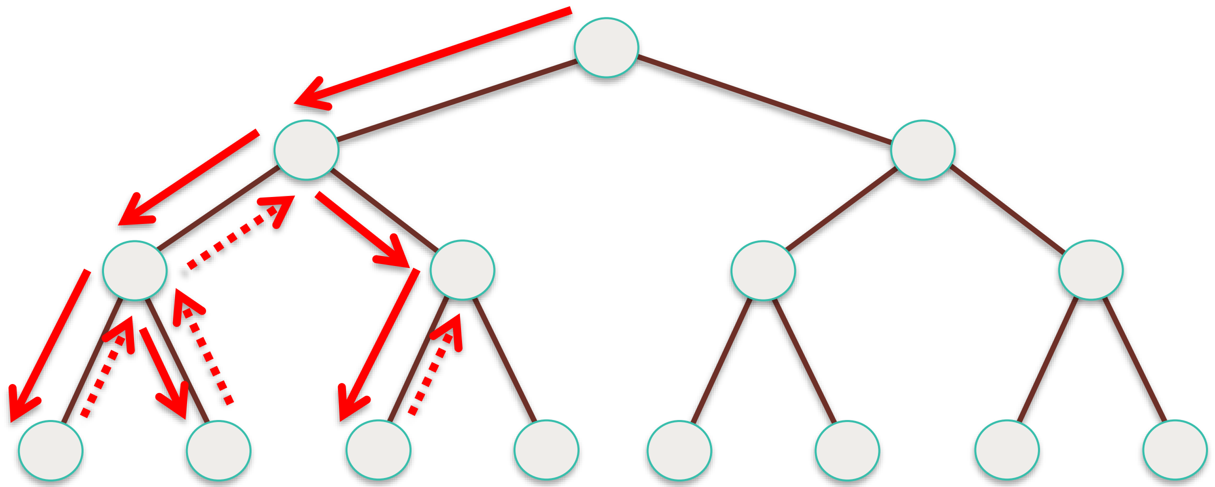
Depth-First Search (DFS): Tree



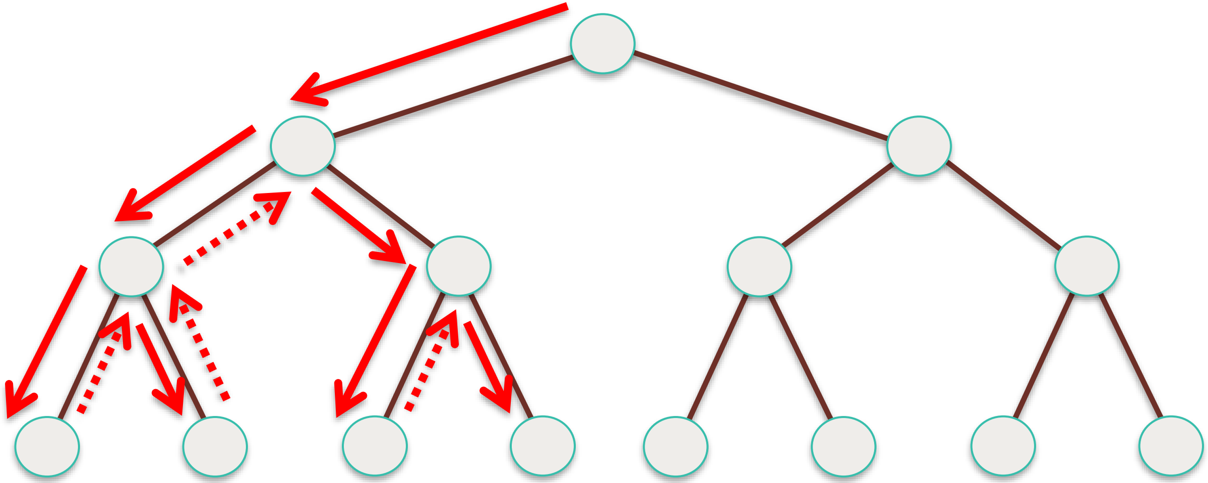
Depth-First Search (DFS): Tree



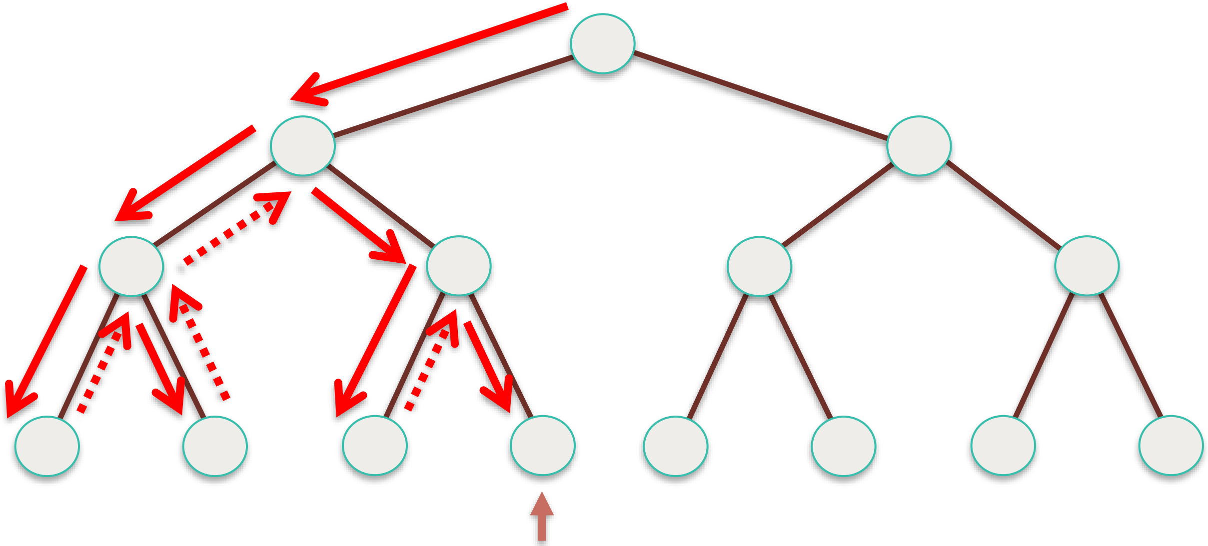
Depth-First Search (DFS): Tree



Depth-First Search (DFS): Tree

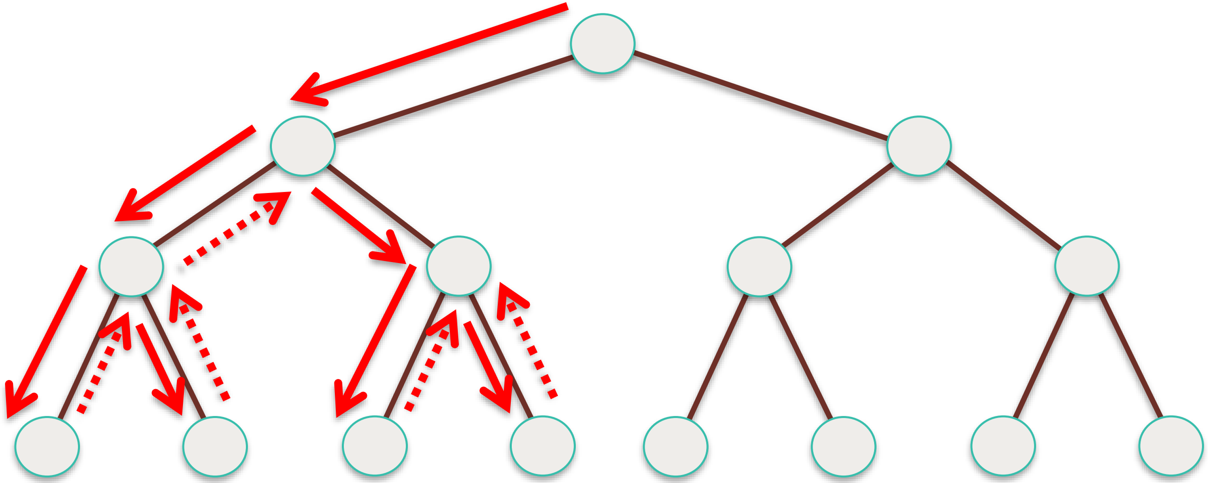


Depth-First Search (DFS): Tree

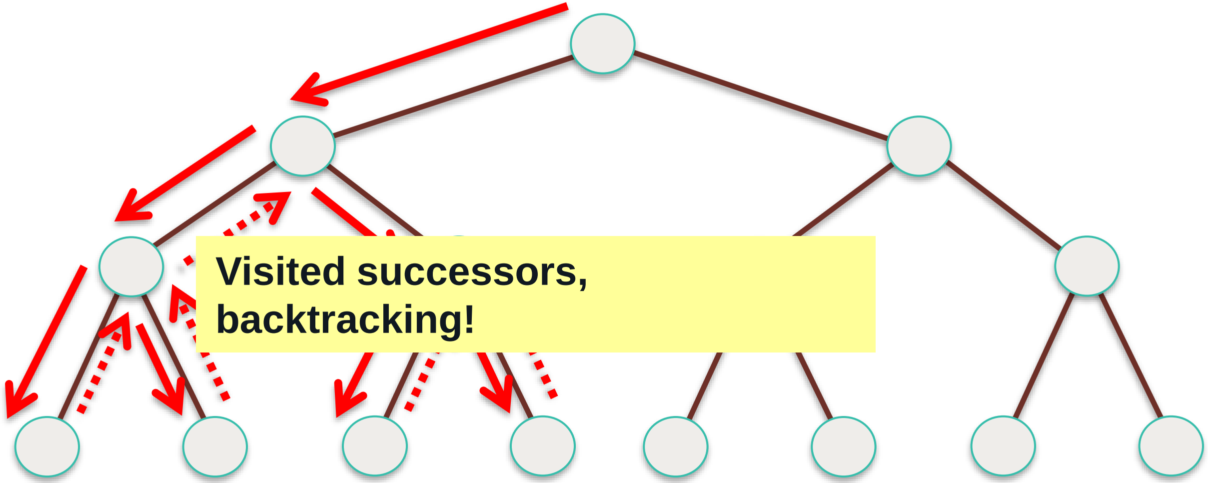


No successor, backtracking!

Depth-First Search (DFS): Tree

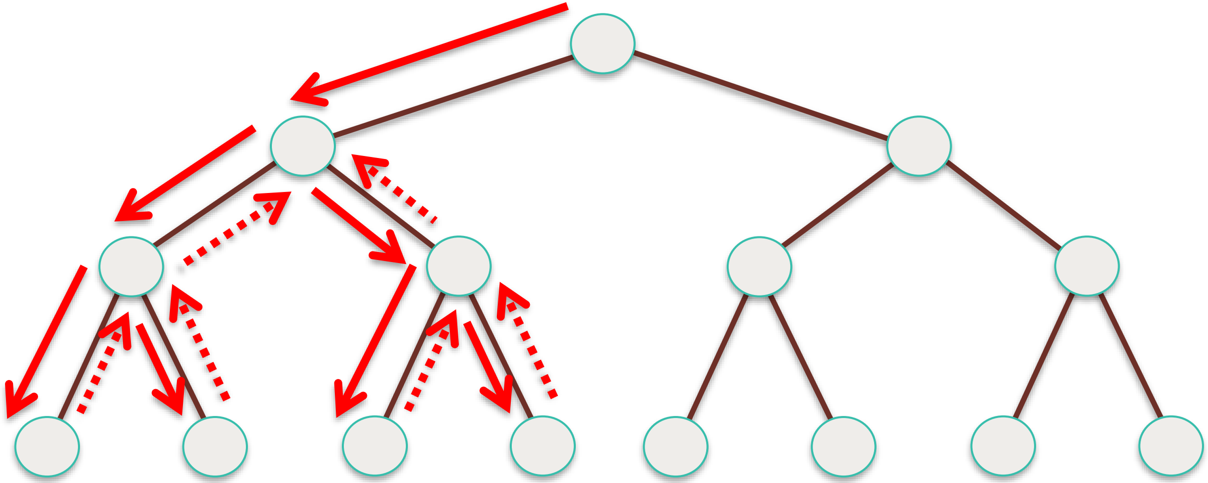


Depth-First Search (DFS): Tree

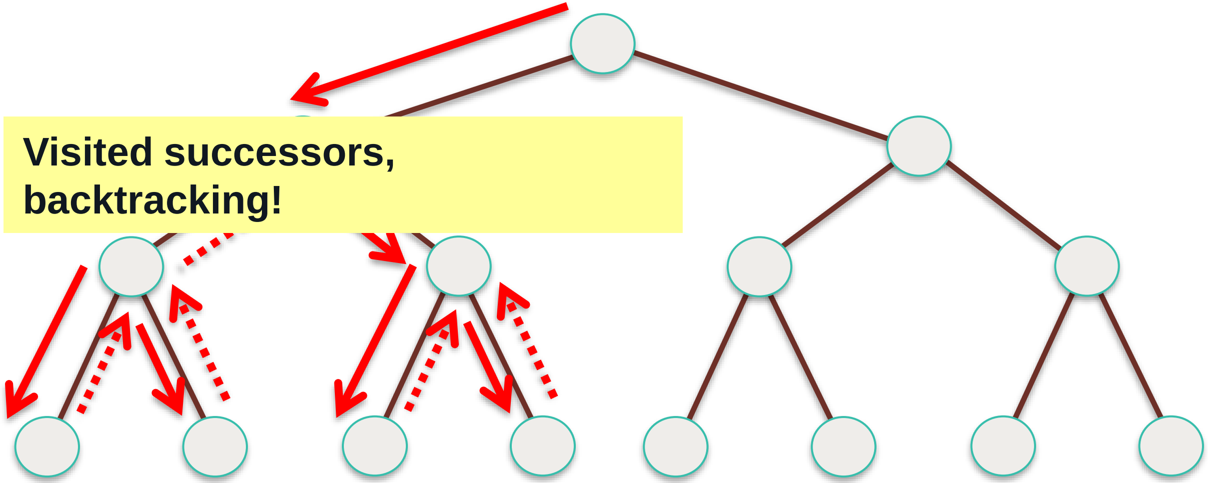


Østfold University College

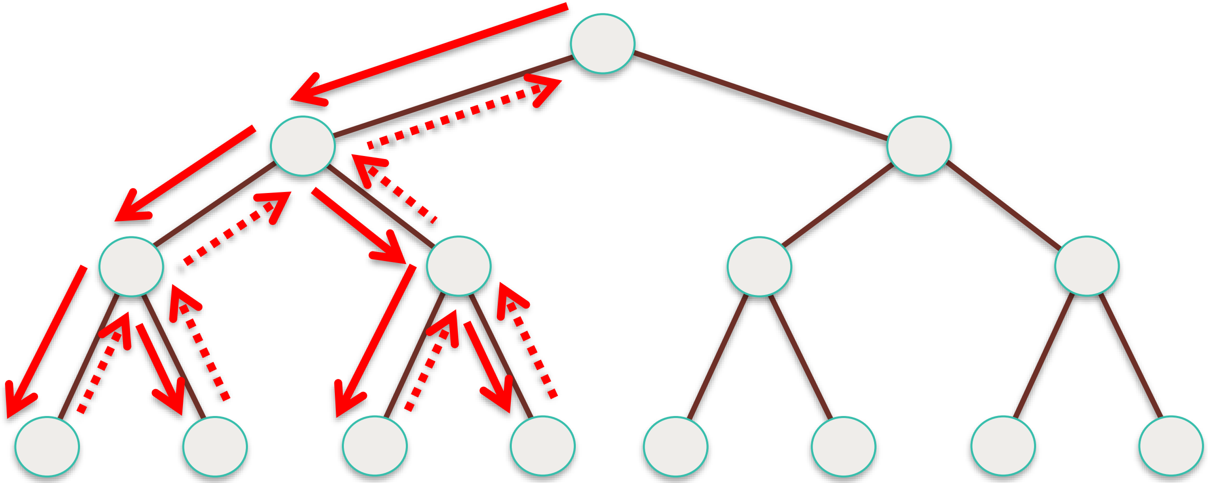
Depth-First Search (DFS): Tree



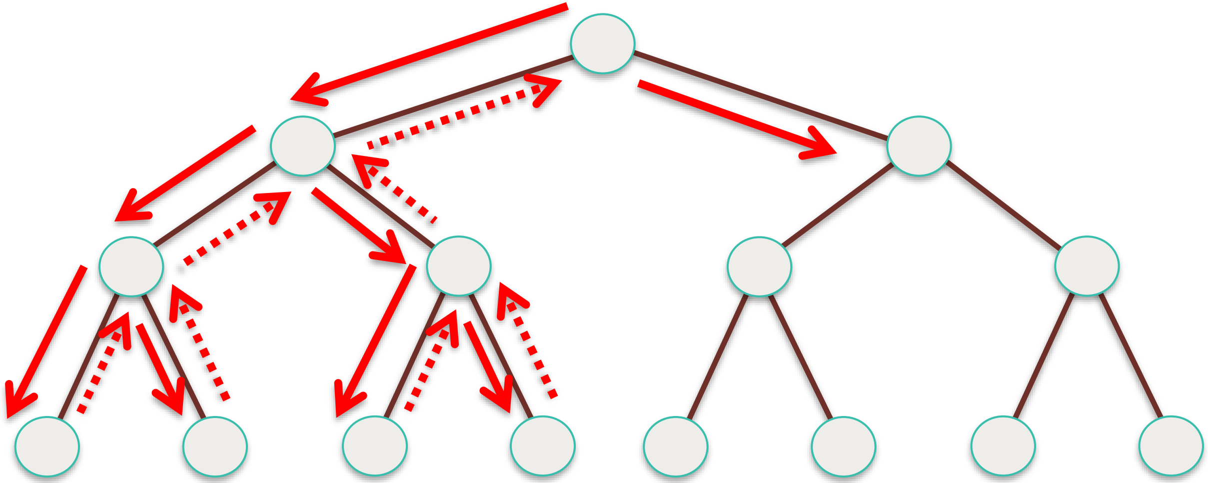
Depth-First Search (DFS): Tree



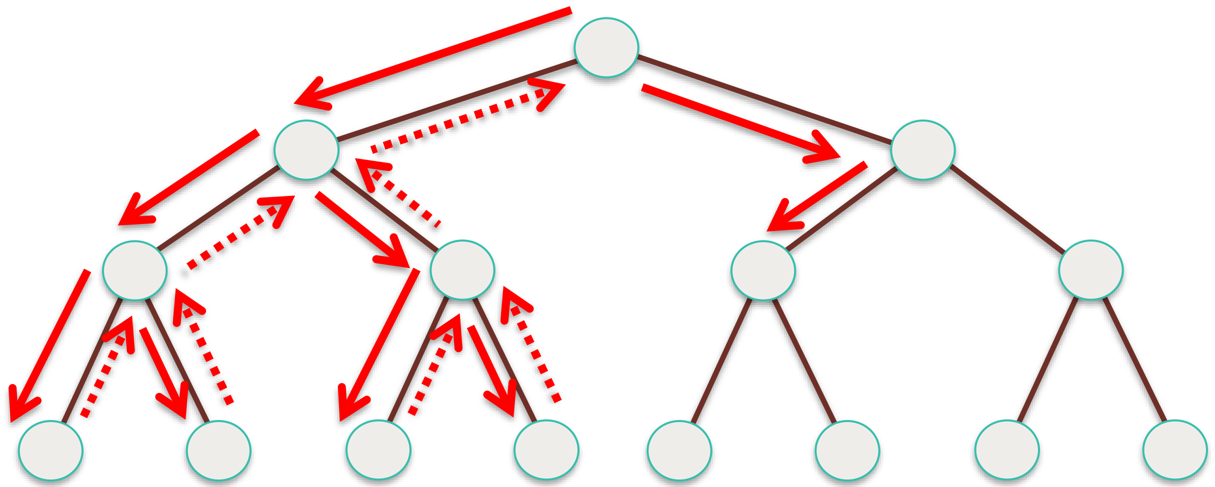
Depth-First Search (DFS): Tree



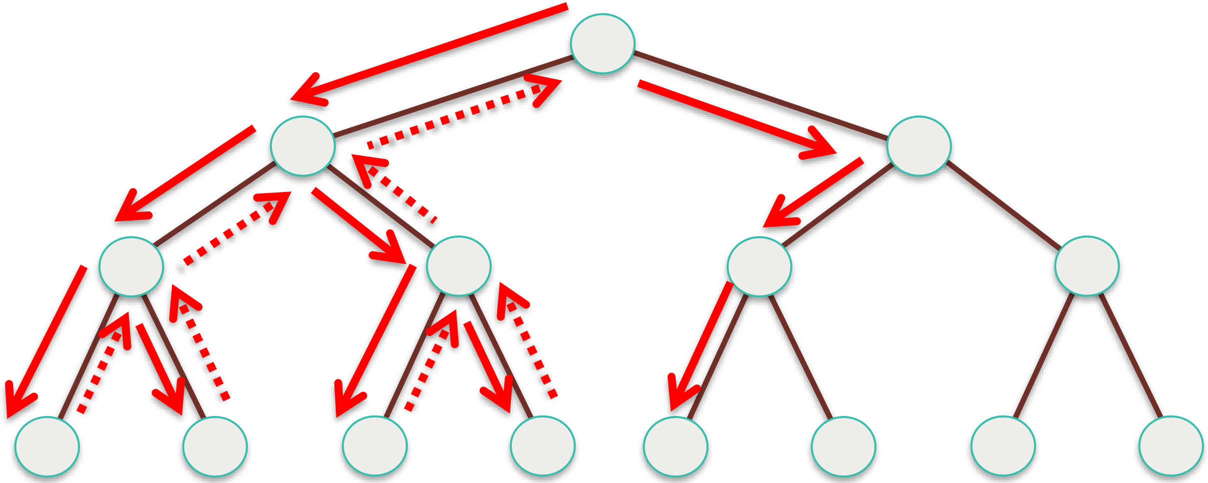
Depth-First Search (DFS): Tree



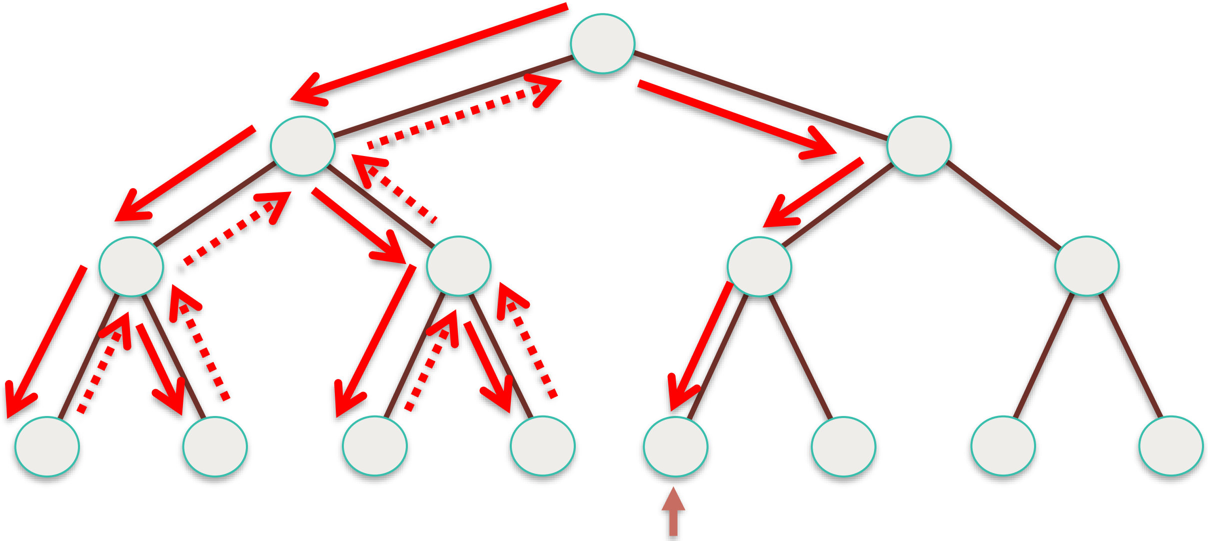
Depth-First Search (DFS): Tree



Depth-First Search (DFS): Tree

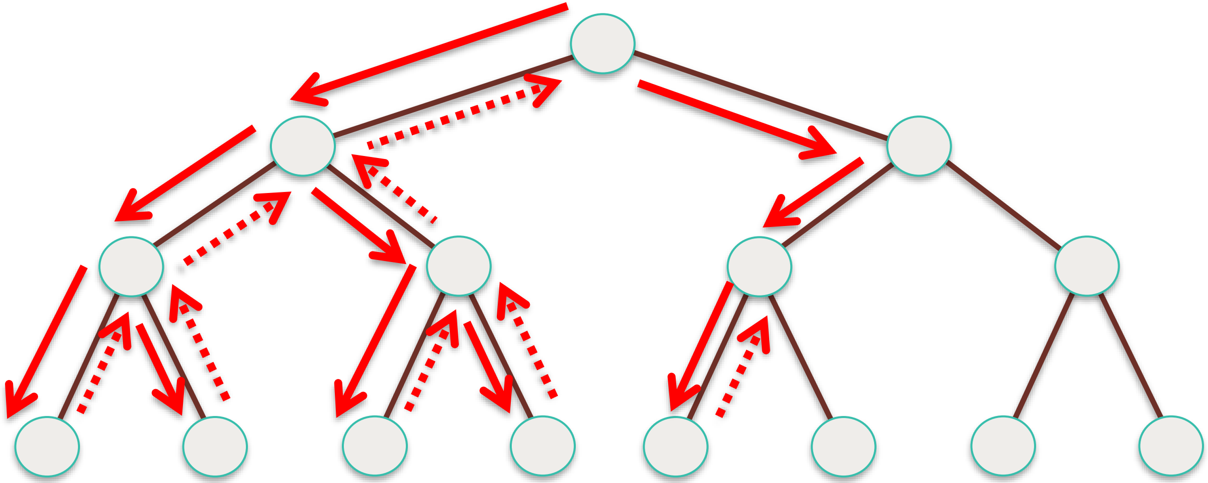


Depth-First Search (DFS): Tree

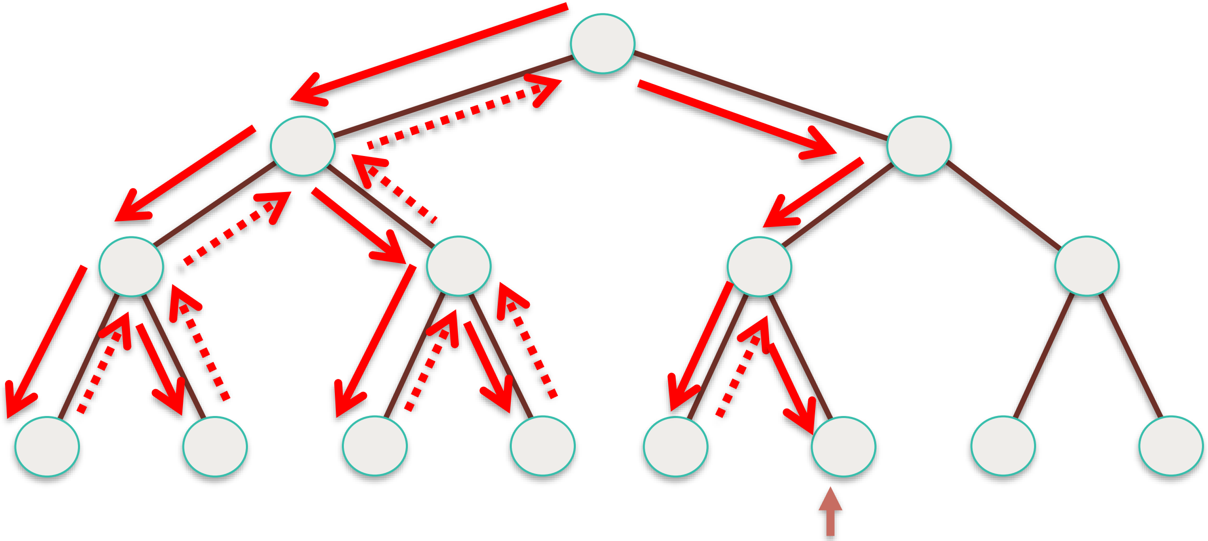


No successor, backtracking!

Depth-First Search (DFS): Tree

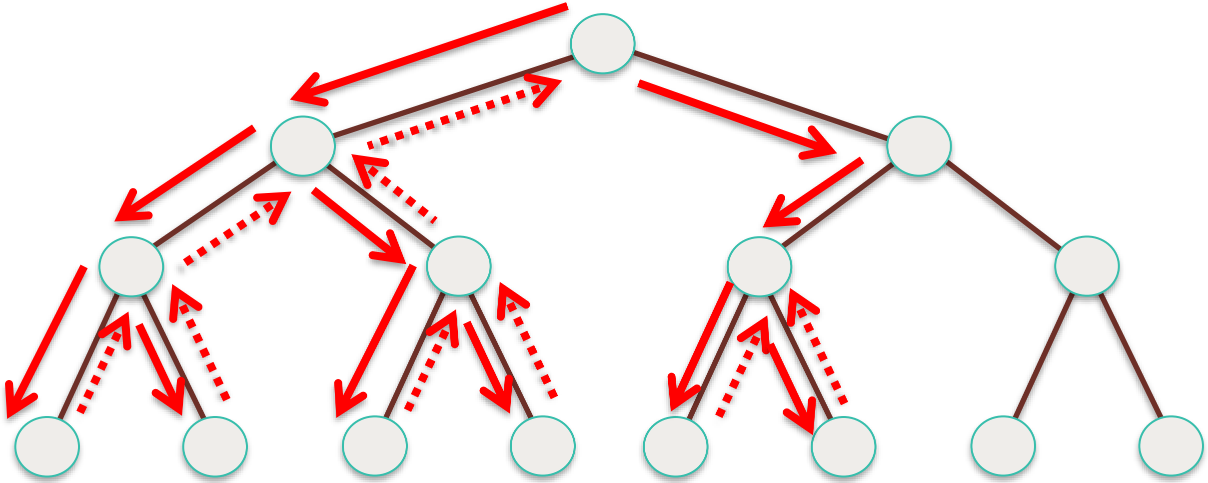


Depth-First Search (DFS): Tree



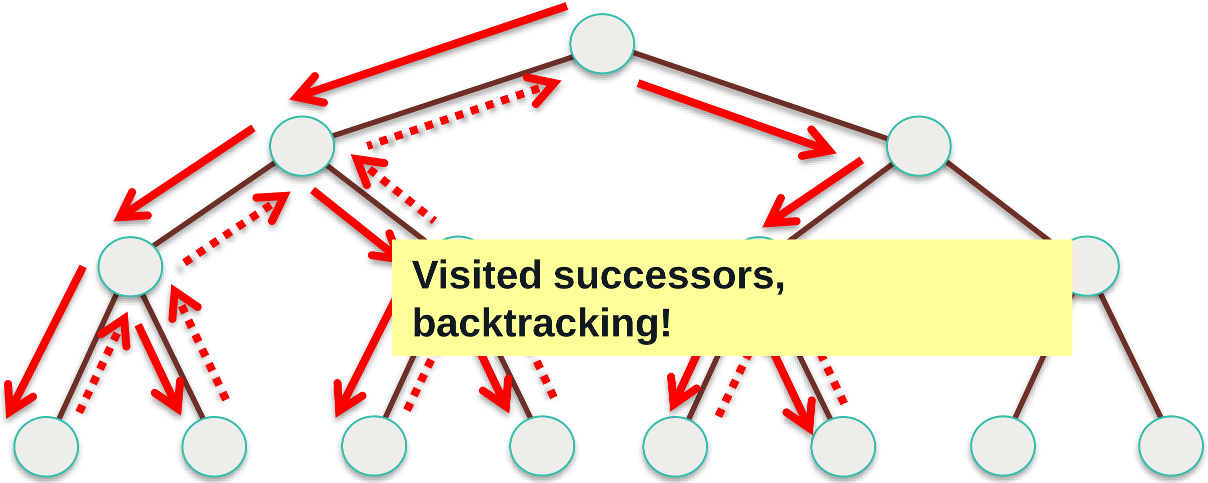
No successor, backtracking!

Depth-First Search (DFS): Tree

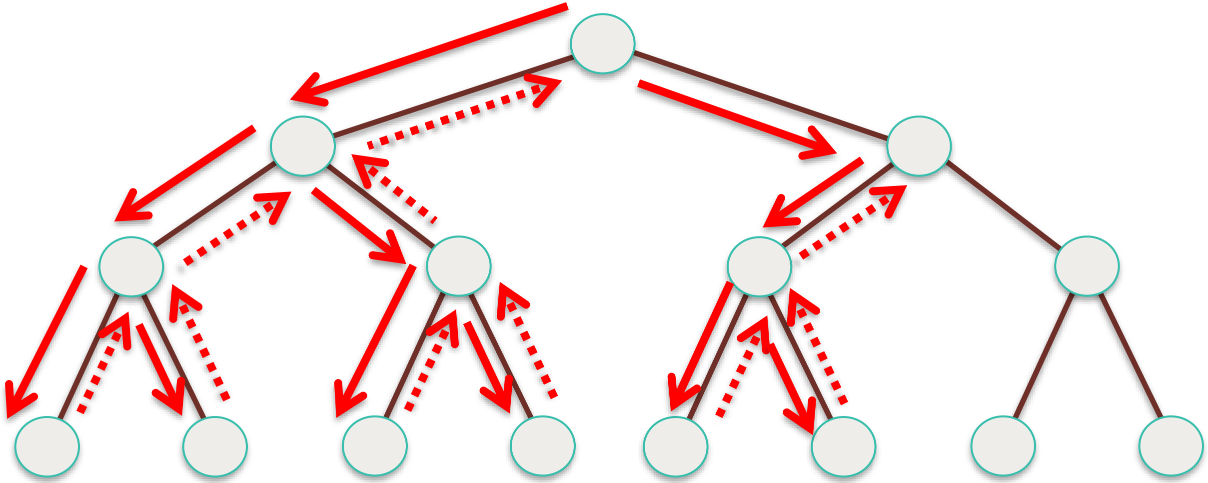


Østfold University College

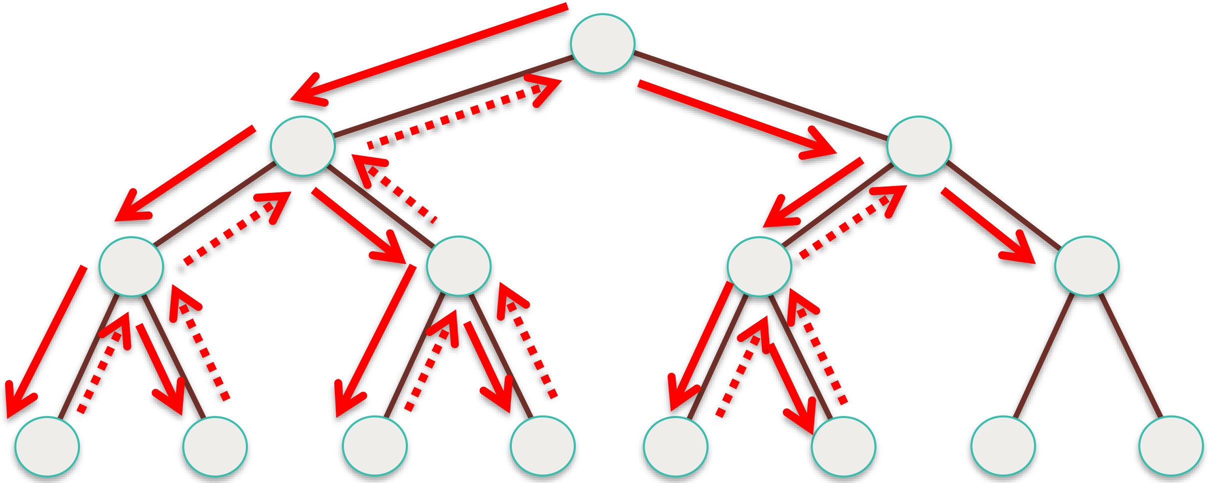
Depth-First Search (DFS): Tree



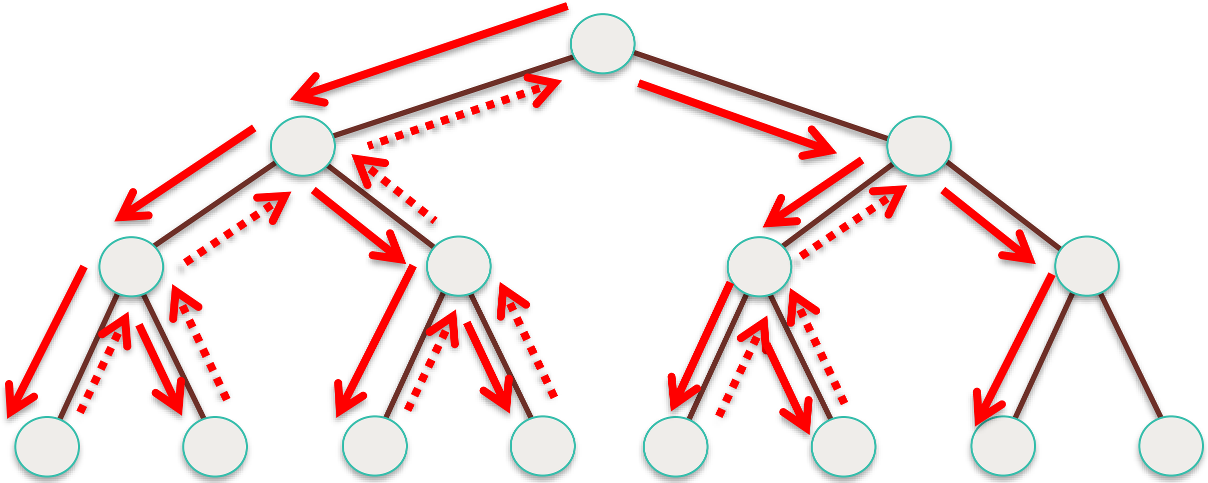
Depth-First Search (DFS): Tree



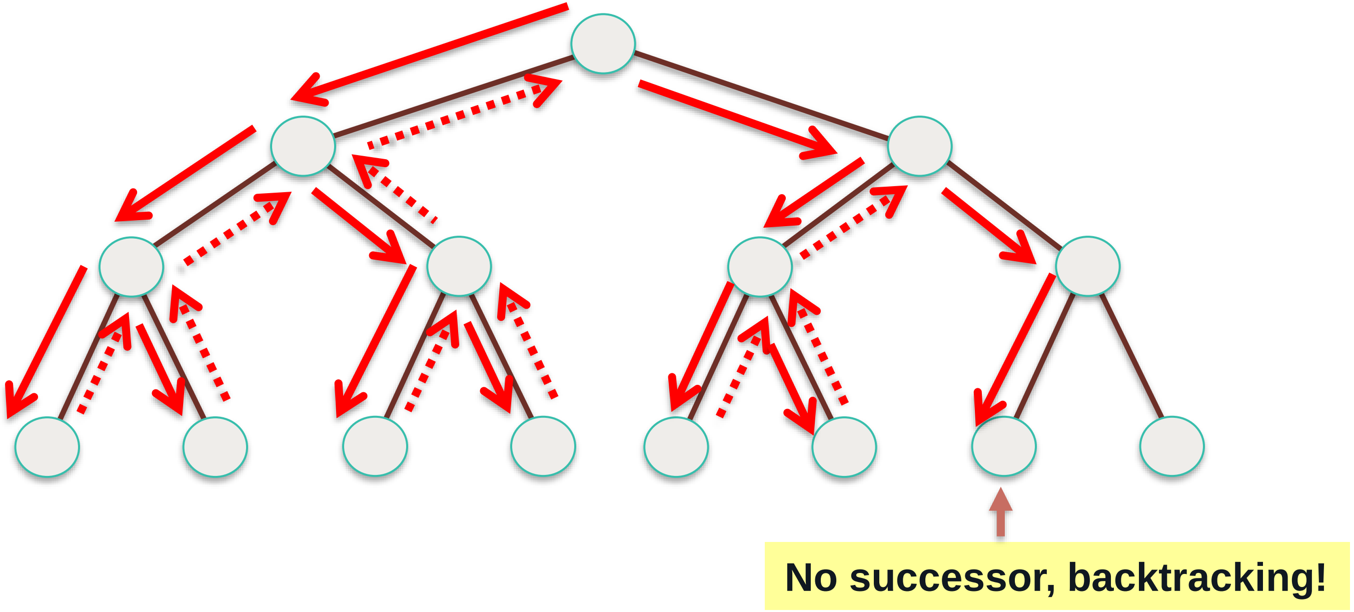
Depth-First Search (DFS): Tree



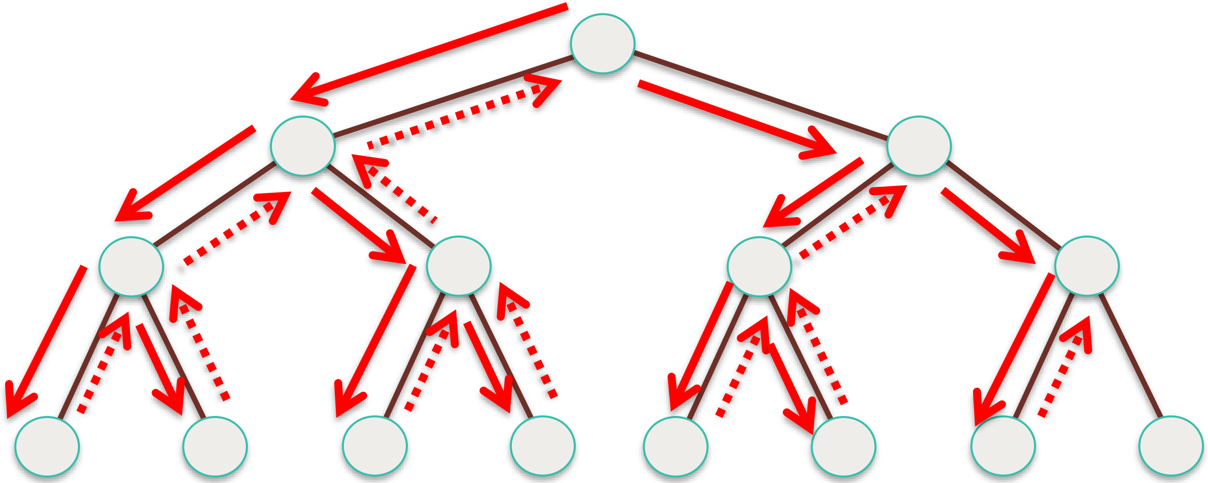
Depth-First Search (DFS): Tree



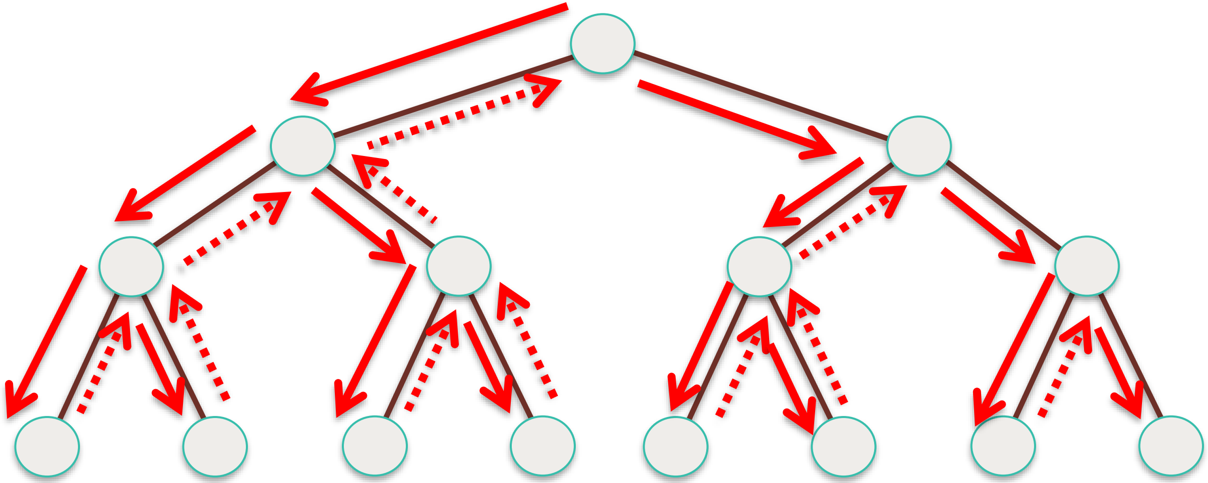
Depth-First Search (DFS): Tree



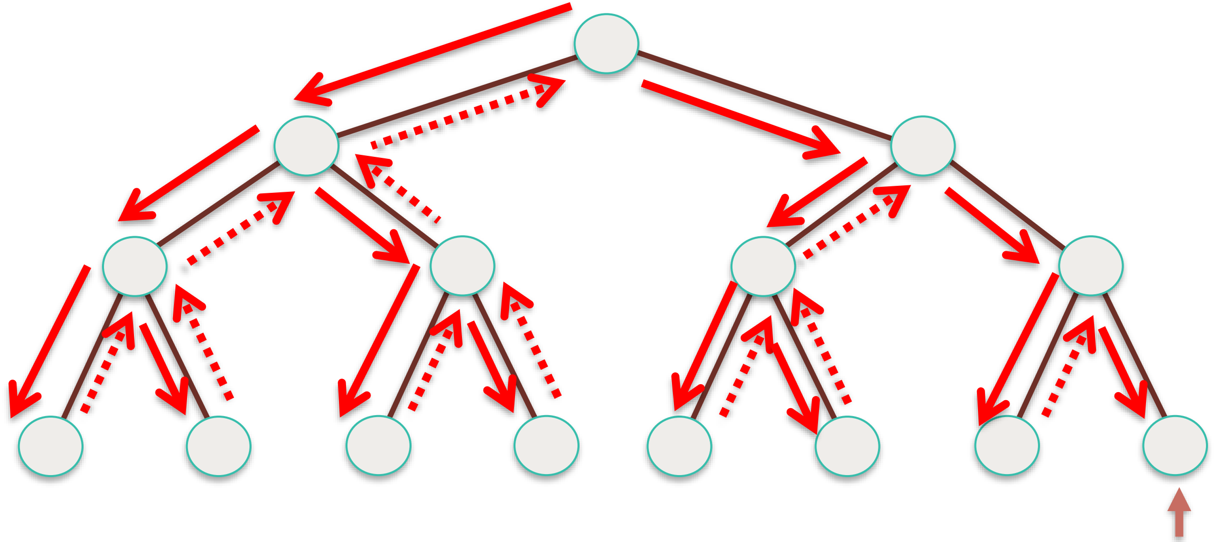
Depth-First Search (DFS): Tree



Depth-First Search (DFS): Tree

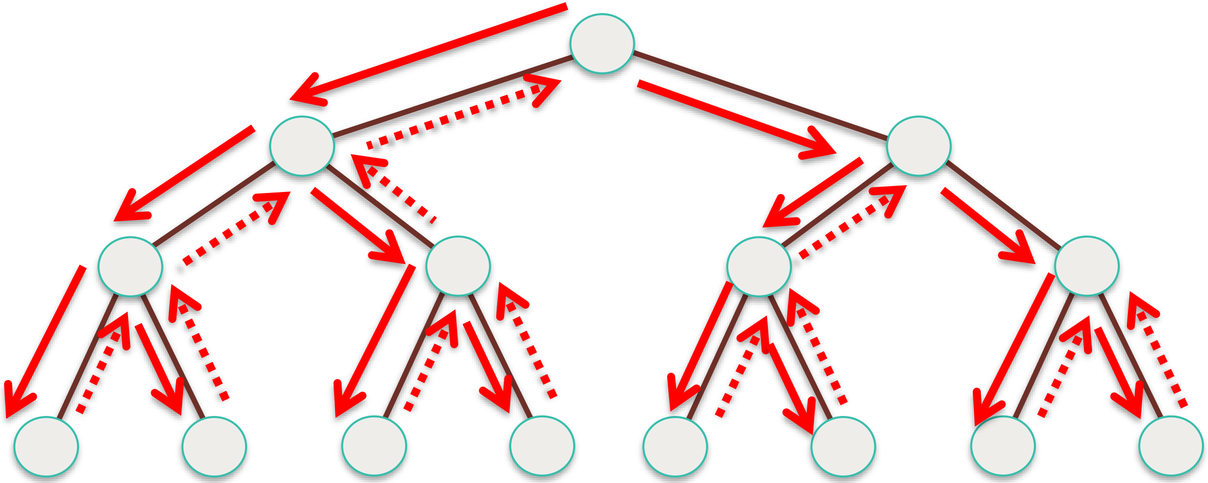


Depth-First Search (DFS): Tree

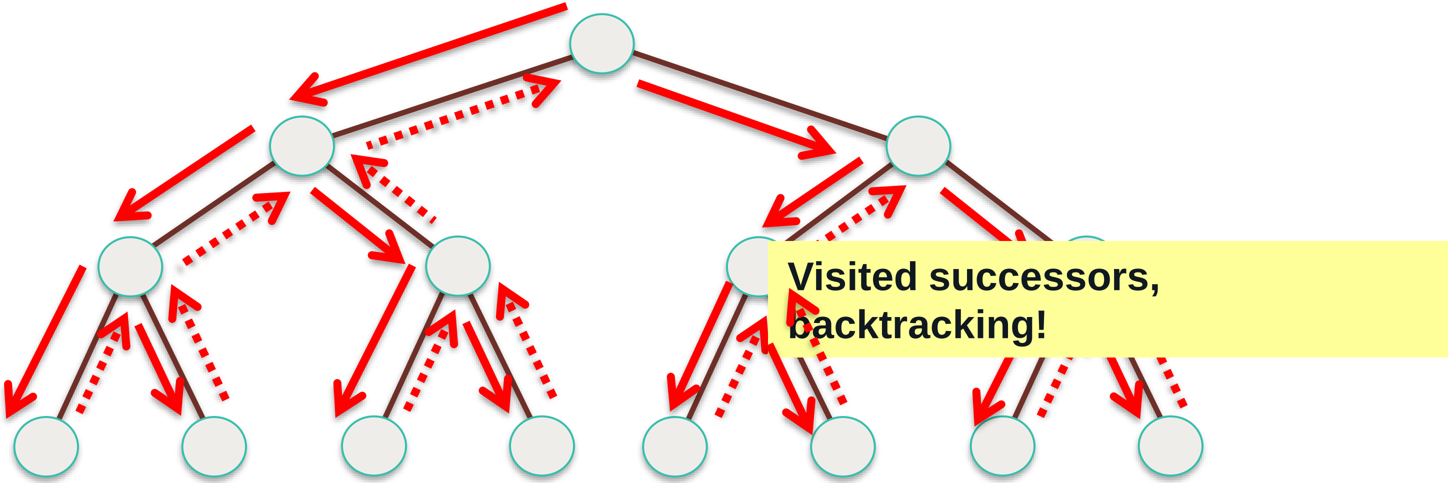


No successor, backtracking

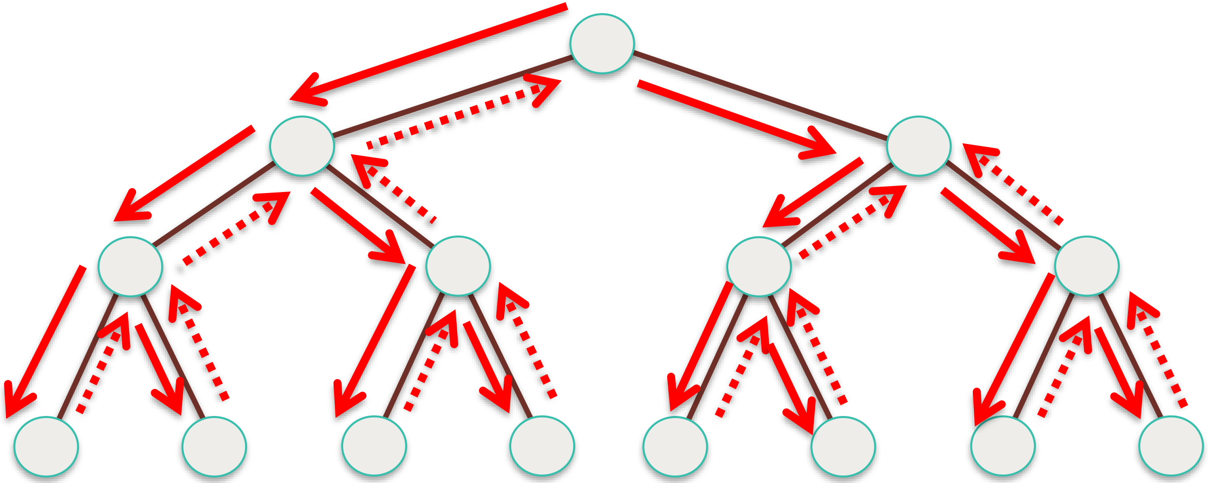
Depth-First Search (DFS): Tree



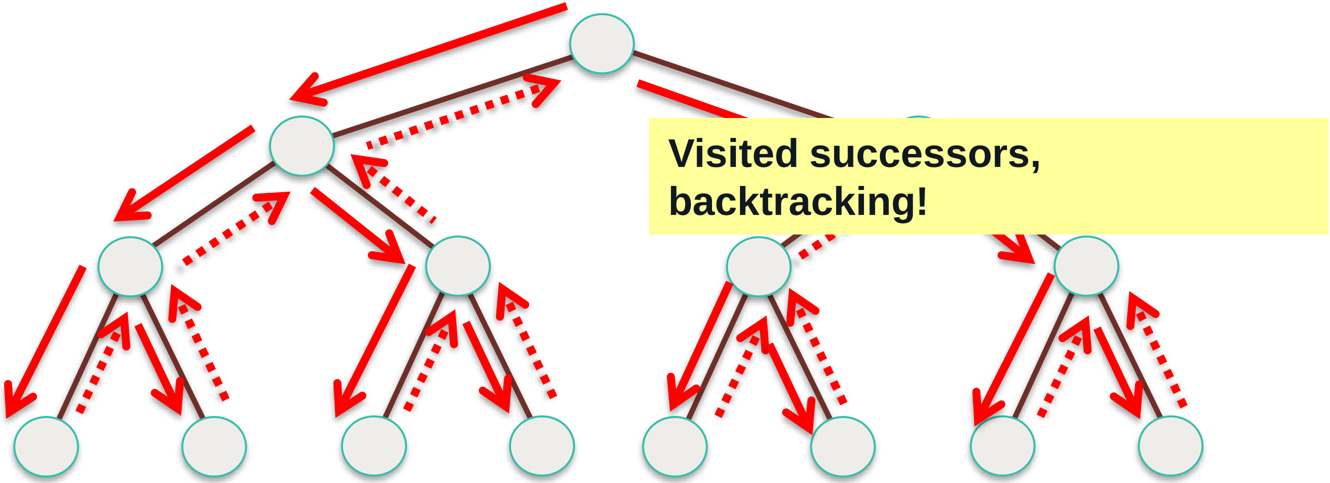
Depth-First Search (DFS): Tree



Depth-First Search (DFS): Tree

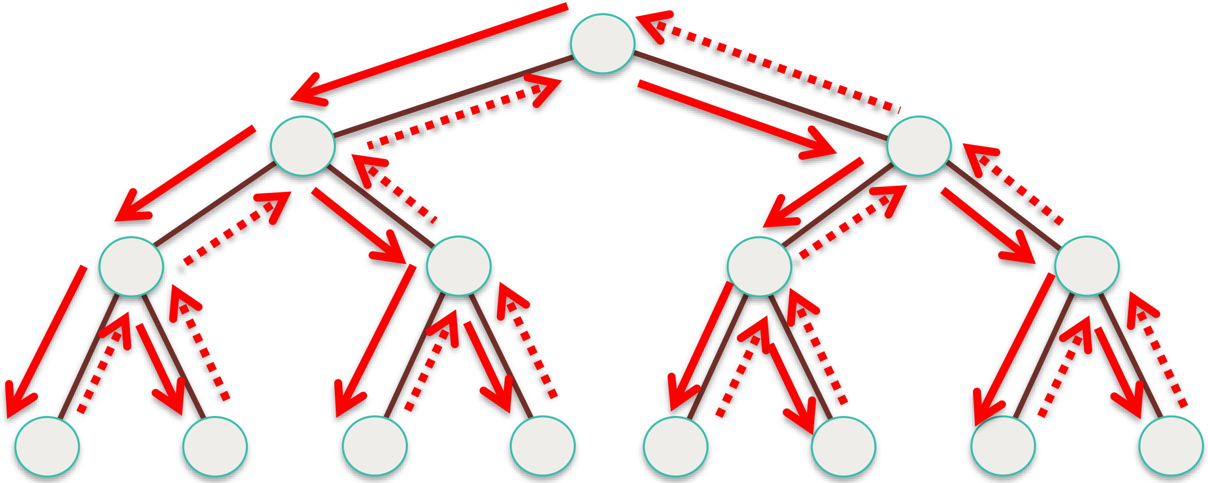


Depth-First Search (DFS): Tree

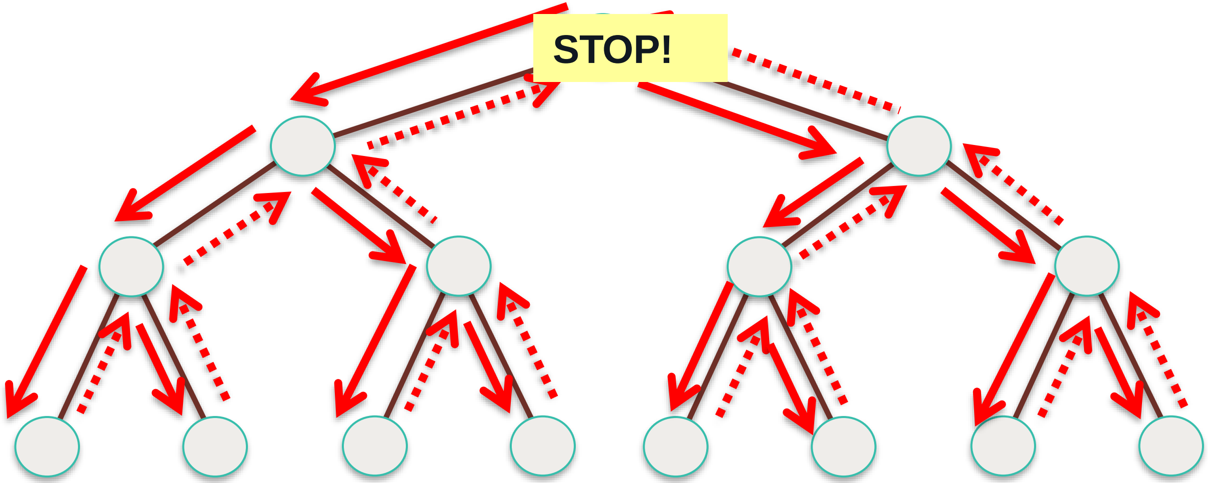


Østfold University College

Depth-First Search (DFS): Tree

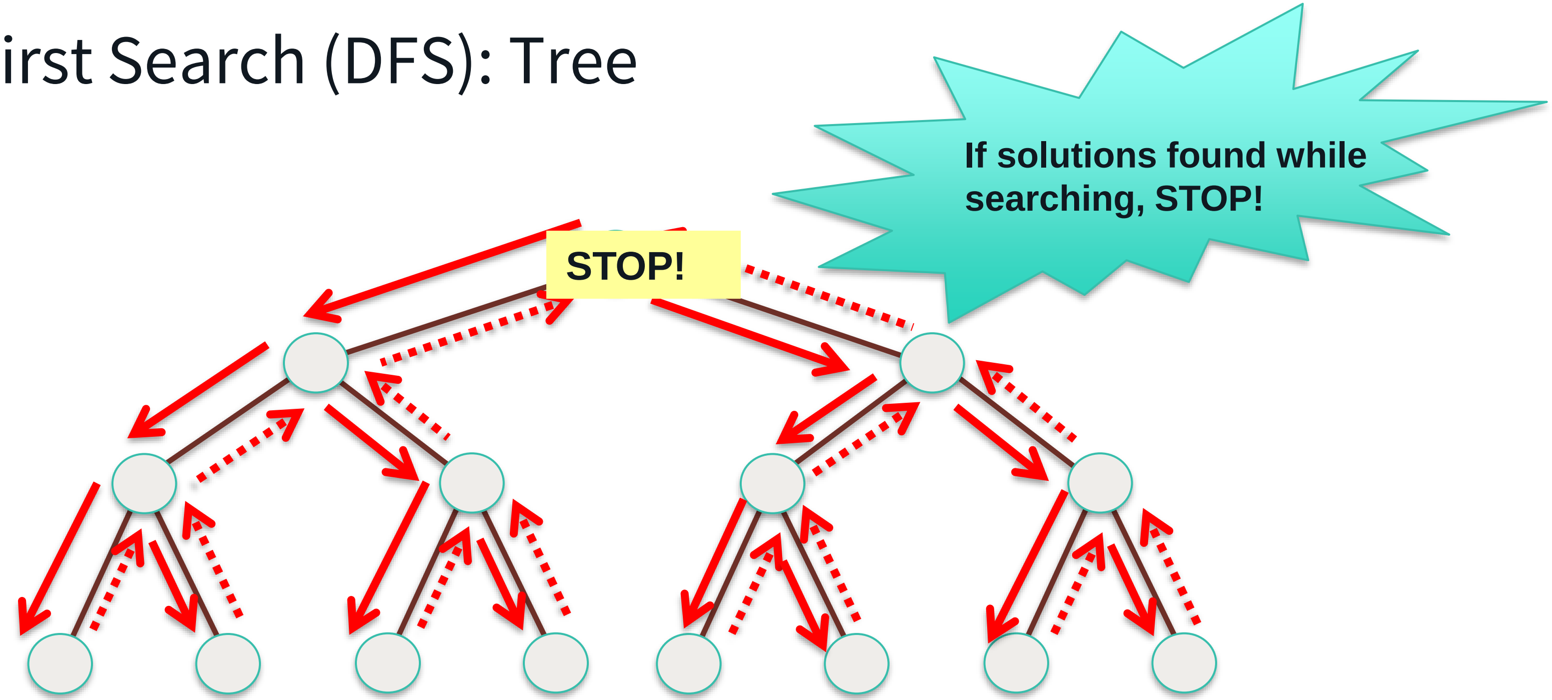


Depth-First Search (DFS): Tree



Østfold University College

Depth-First Search (DFS): Tree



Depth-First Search (DFS)

- Expand the initial node & generate its successors
- In each subsequent step, DFS expands one of the most recently generated nodes.
  - If this node has no successors (or cannot lead to any solutions),
    - Backtracks
    - Expands an alternate child
- Successors of a node are often ordered based on their likelihood of reaching a solution

Depth-First Search (DFS) – Simple Backtracking

- Simple backtracking performs DFS until it finds the first feasible solution and terminates
- No optimal solution guaranteed
- Simple backtracking: uses no heuristic information to order the successors of an expanded node
- Ordered backtracking: uses heuristics to order the successors of an expanded node

Applications of DSF

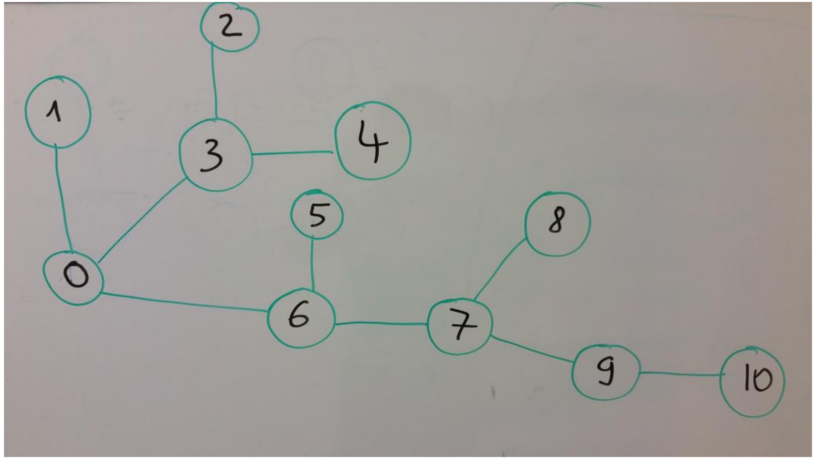
- Detecting cycle in a graph
- Path Finding: Find a path between two given vertices u and v.
  - Call DFS(G, u) with u as the start vertex
  - Keep track of the path between the start vertex and the current vertex
  - As soon as destination vertex z is encountered, return the path
- Finding Strongly Connected Components of a graph
  - A directed graph is called strongly connected if there is a path from each vertex in the graph to every other vertex

https://www.geeksforgeeks.org/applications-of-depth-first-search/

Depth-First Search (DFS): Pseudocode

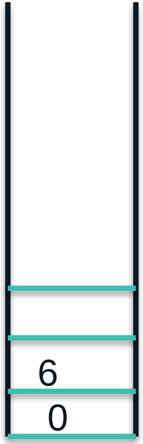
```
DFS-recursive(G, s):
     mark s as visited
     for all neighbours w of s in Graph G:
          if w is not visited:
               DFS-recursive(G, w)
```

Sample 'Implementation'

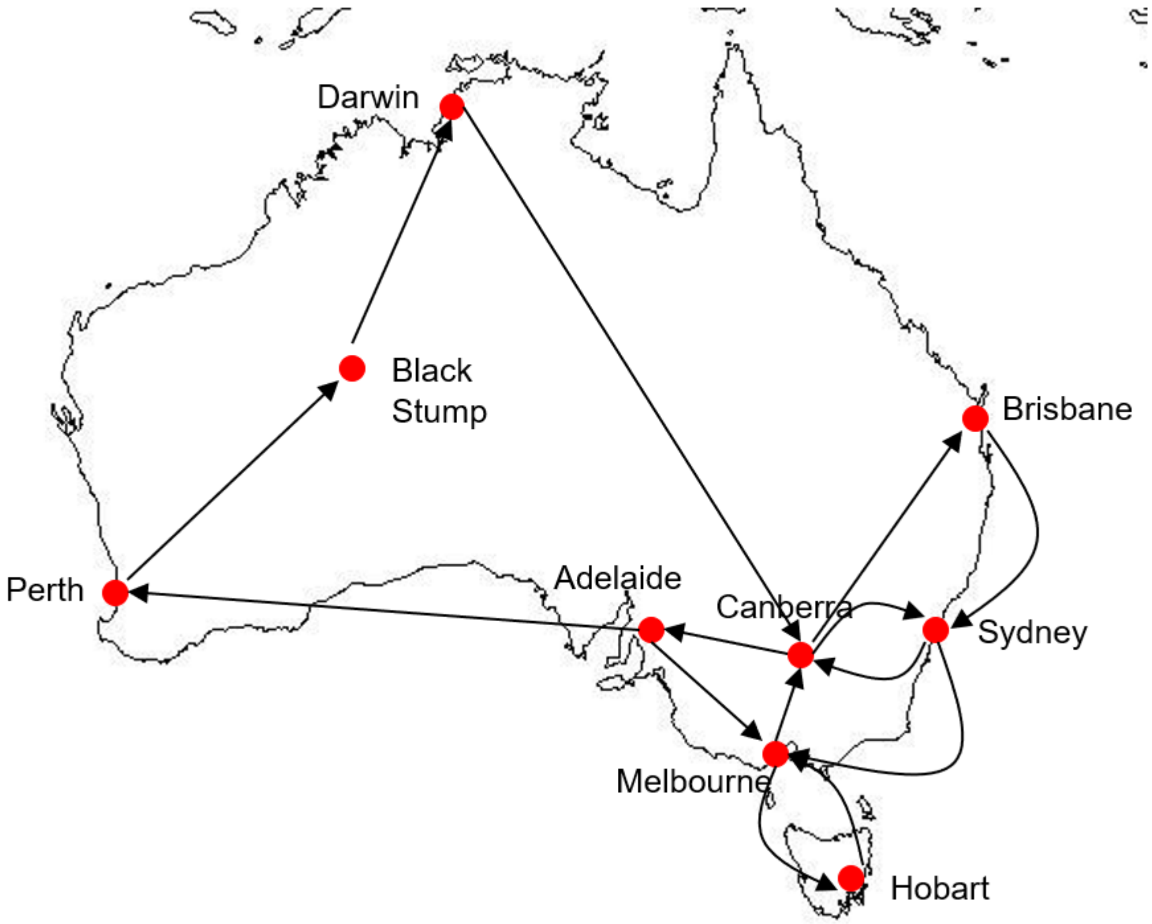


Data structure first: adjacent matrix or adjacent list ?
Start vertex: 0 (but can be any other vertex)
Use Stack (Last-in-First-Out)

DFS: 0, 6, 7, 9, 10, 8, 5, 3, 4, 2,1



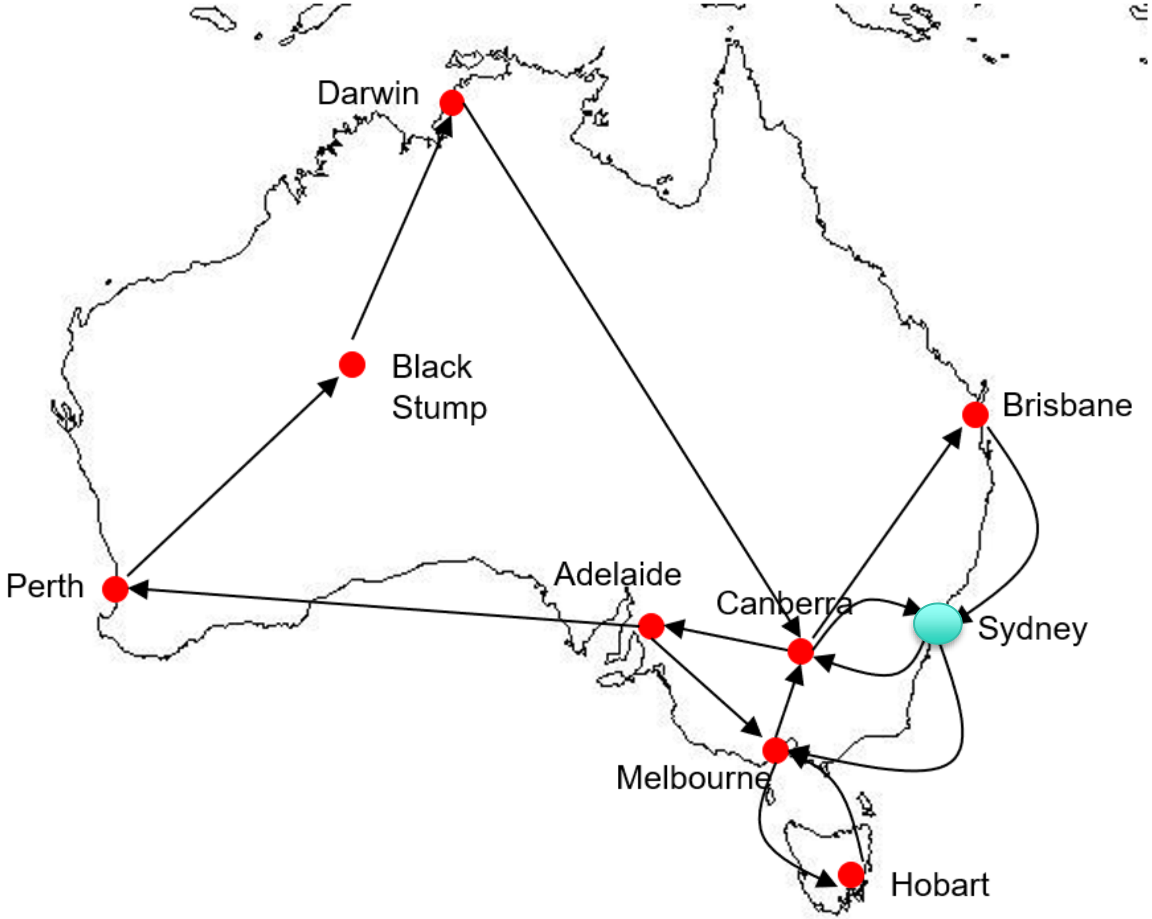
BSF

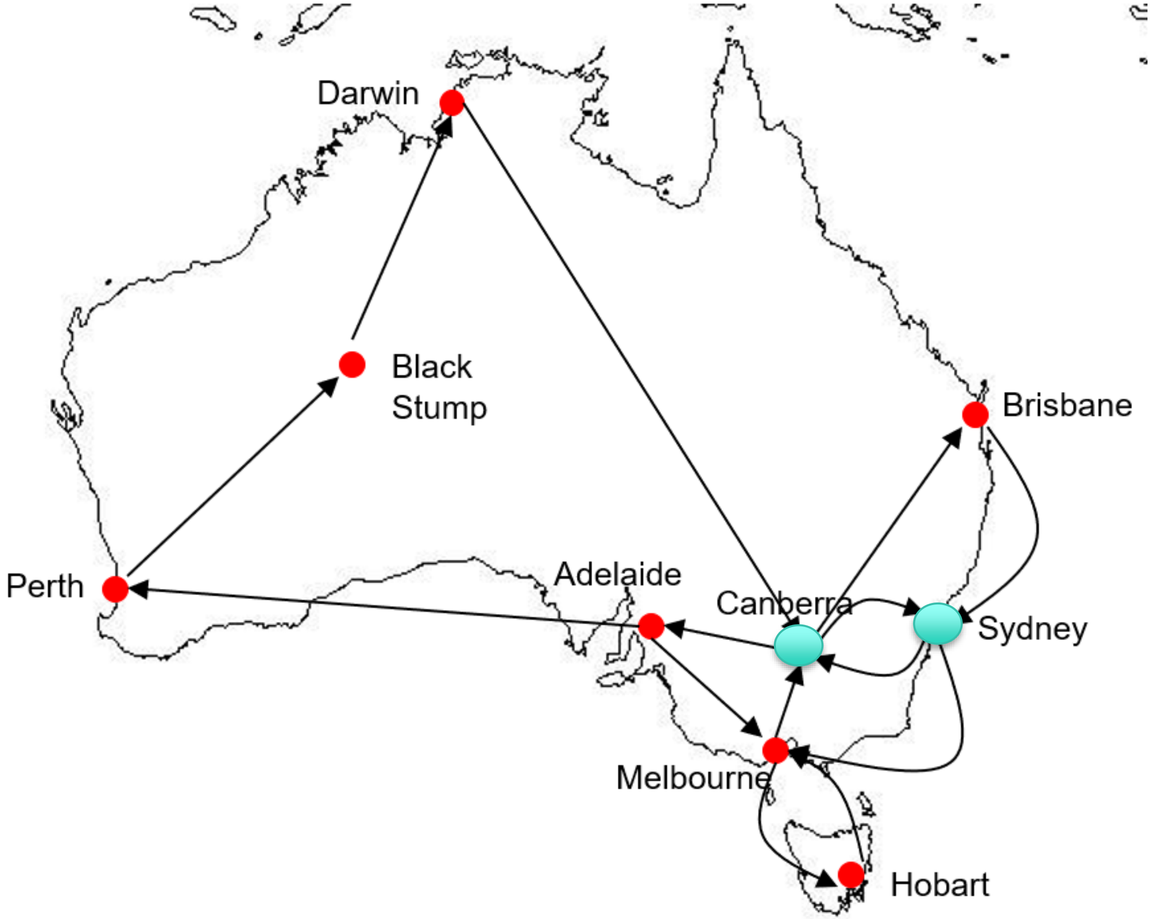


Darwin
Black
Stump
Brisbane
Perth
Adelaide
Canberra
Sydney
Melbourne
Hobart

- BSF

Sydney

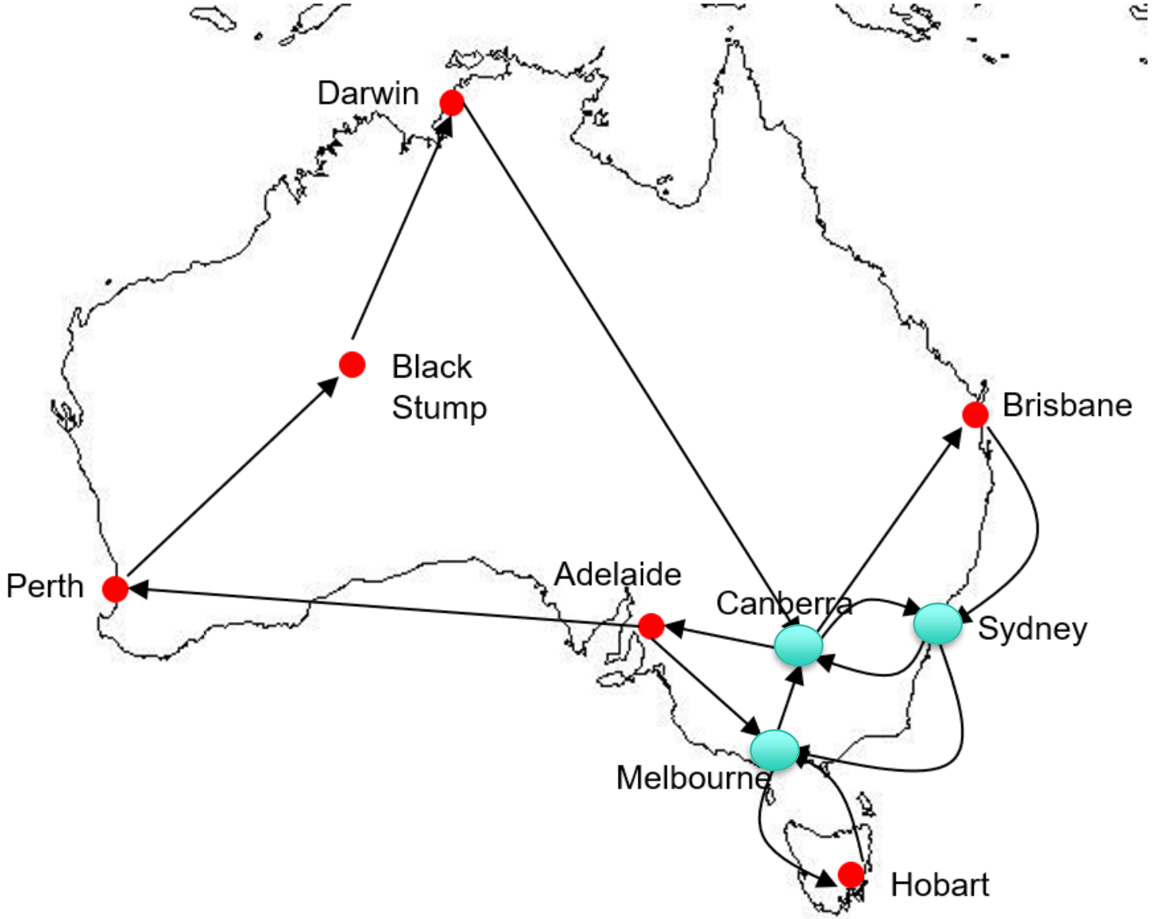

Darwin
Black Stump
Brisbane
Perth
Adelaide
Canberra
Sydney
Melbourne
Hobart



- BSF

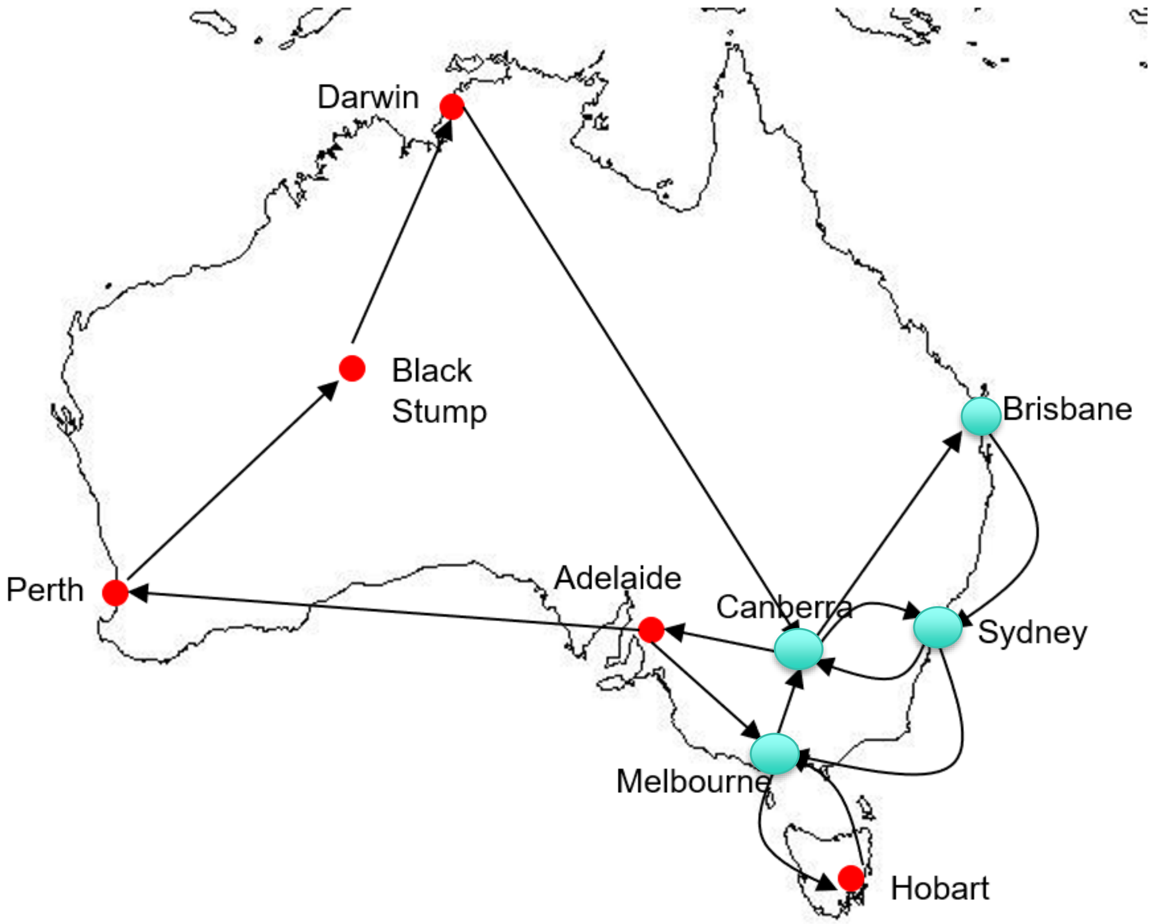
Sydney

Canberra



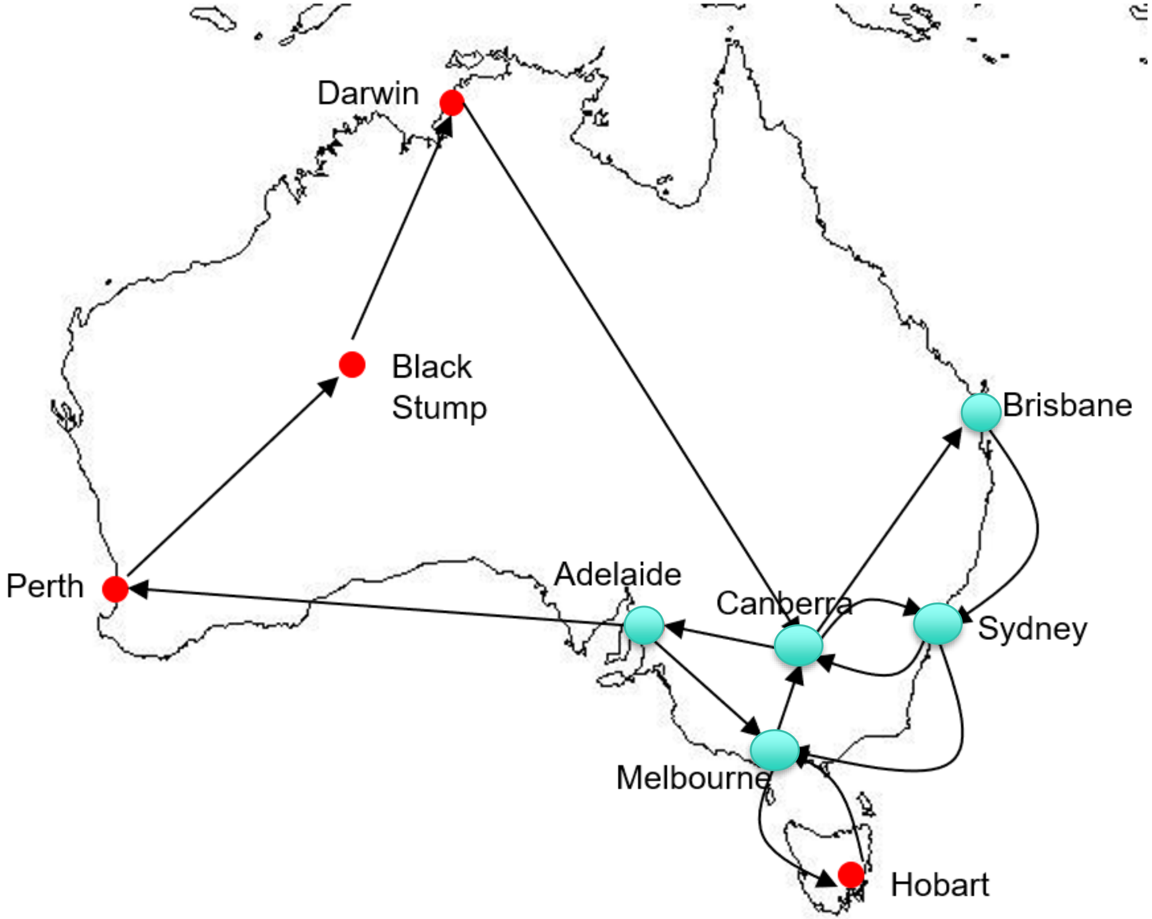
- BSF

Sydney
Canberra
Melbourne

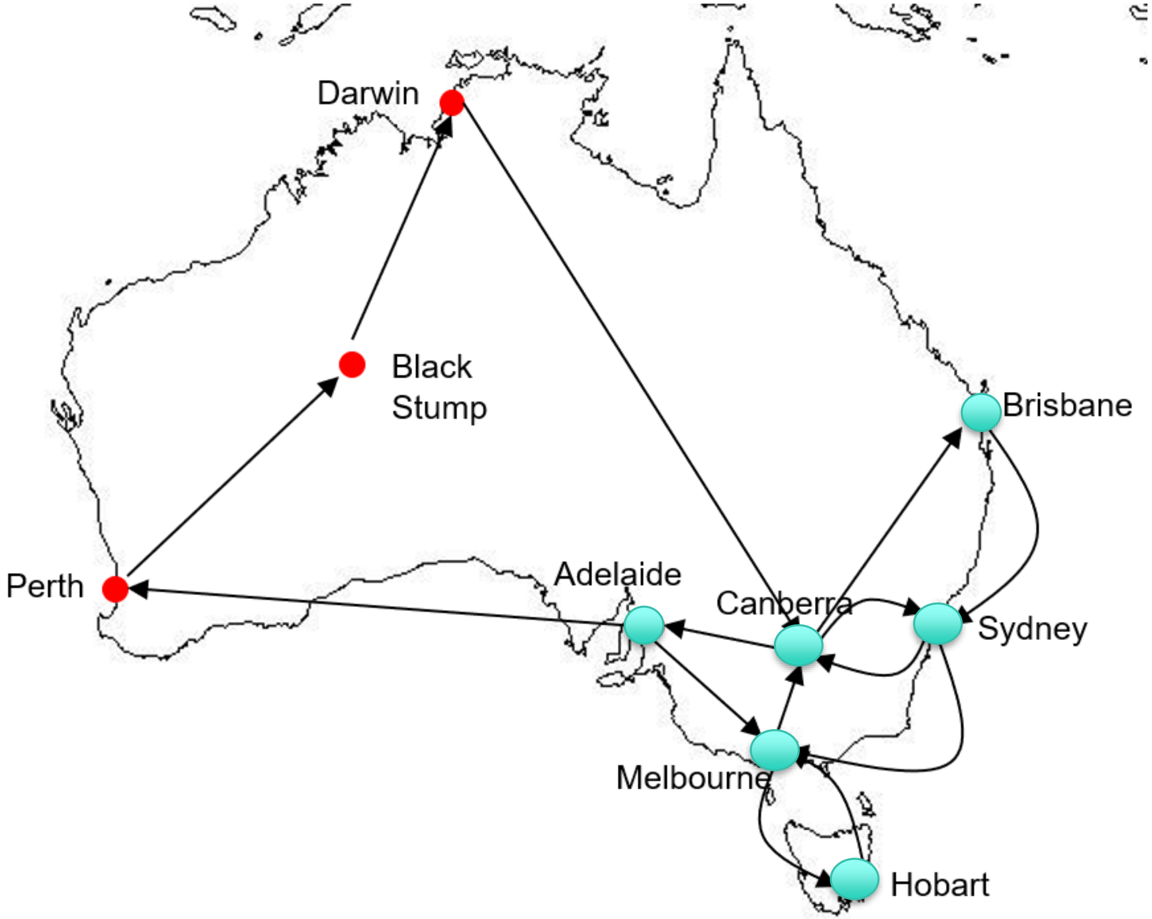


- BSF

Sydney
Canberra
Melbourne
Brisbane

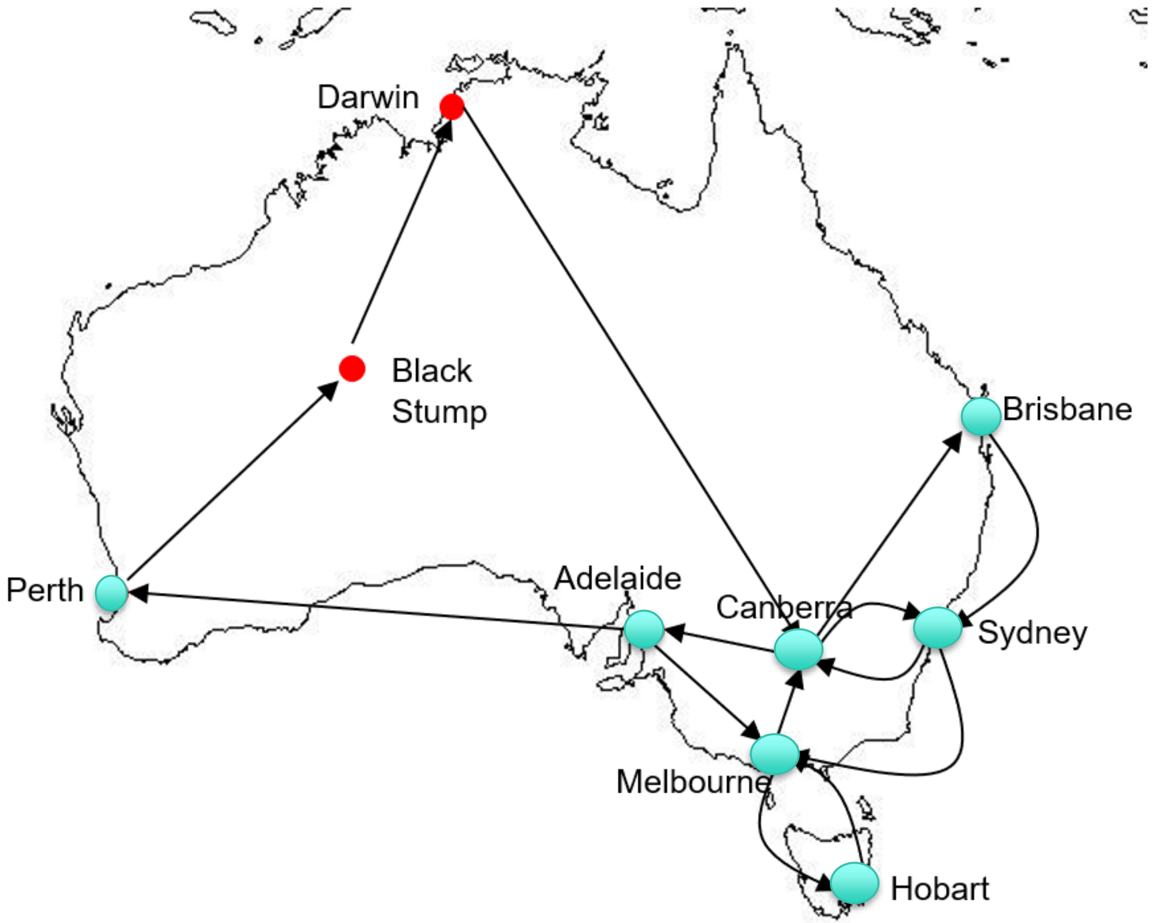


- BSF
Sydney
Canberra
Melbourne
Brisbane
Adelaide



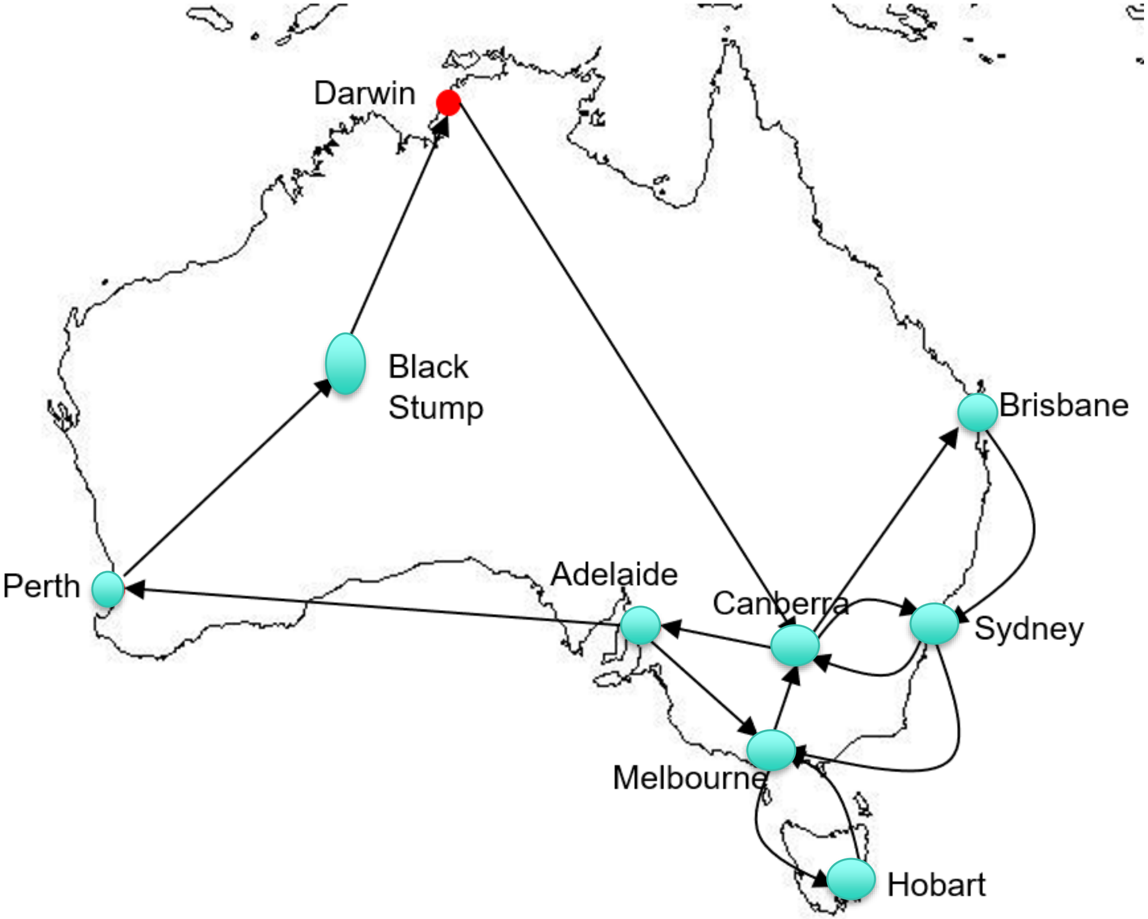
- BSF

Sydney
Canberra
Melbourne
Brisbane
Adelaide
Hobart



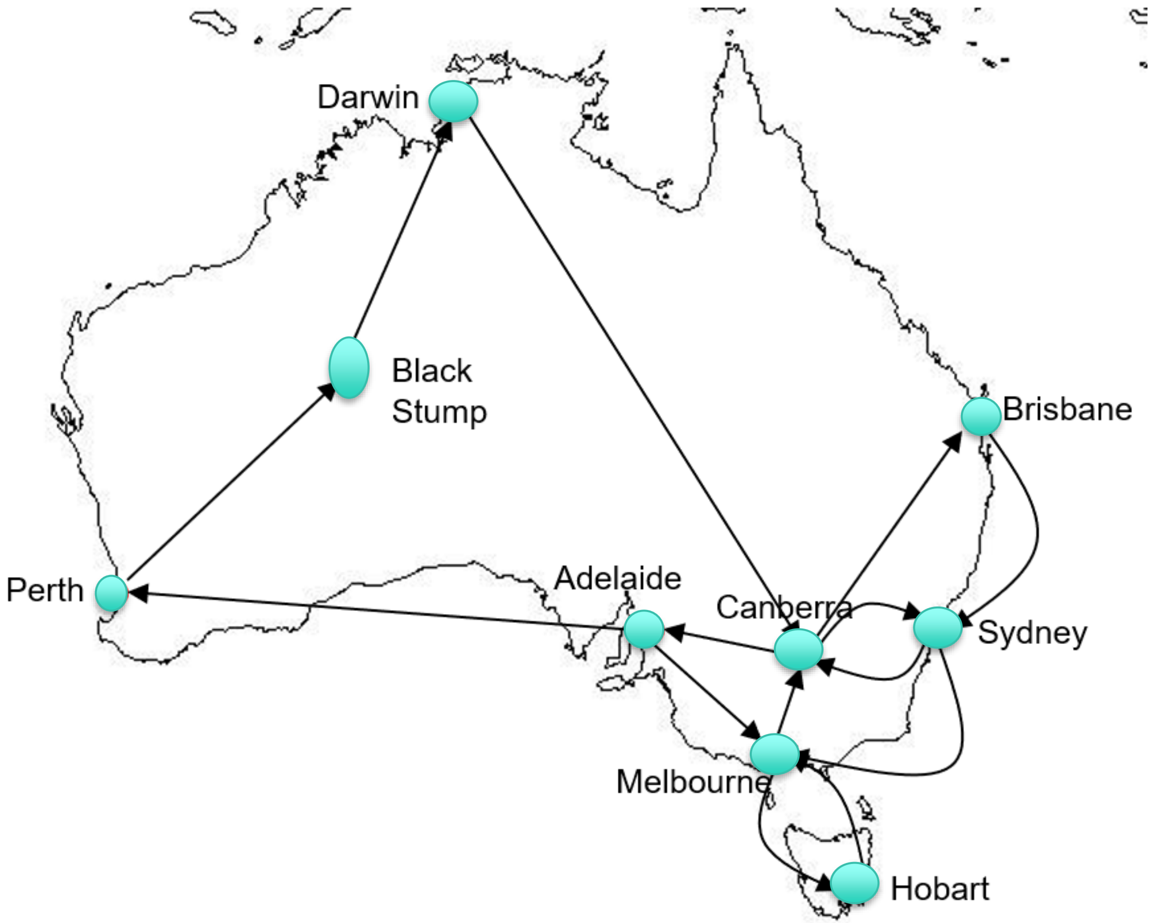
- BSF

Sydney
Canberra
Melbourne
Brisbane
Adelaide
Hobart
Perth



- BSF

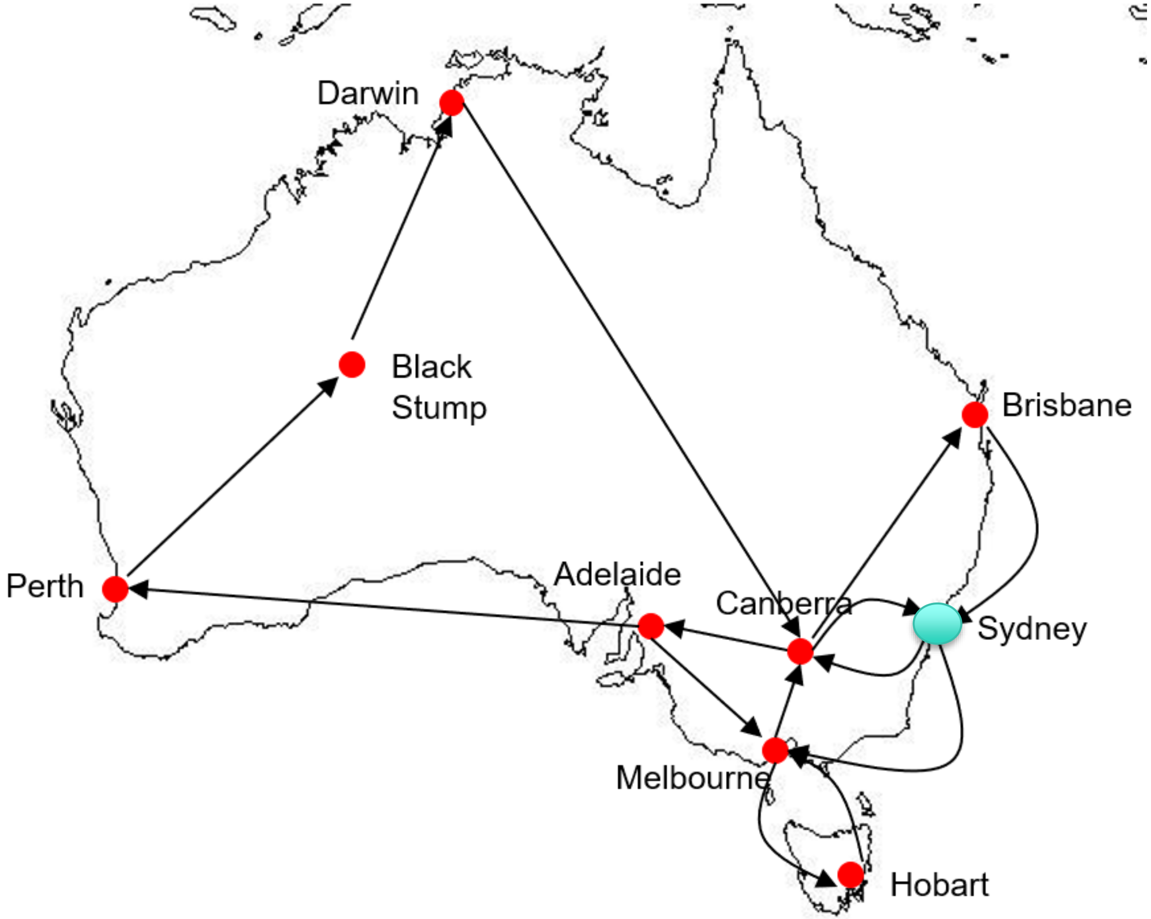
Sydney
Canberra
Melbourne
Brisbane
Adelaide
Hobart
Perth
Black Stump

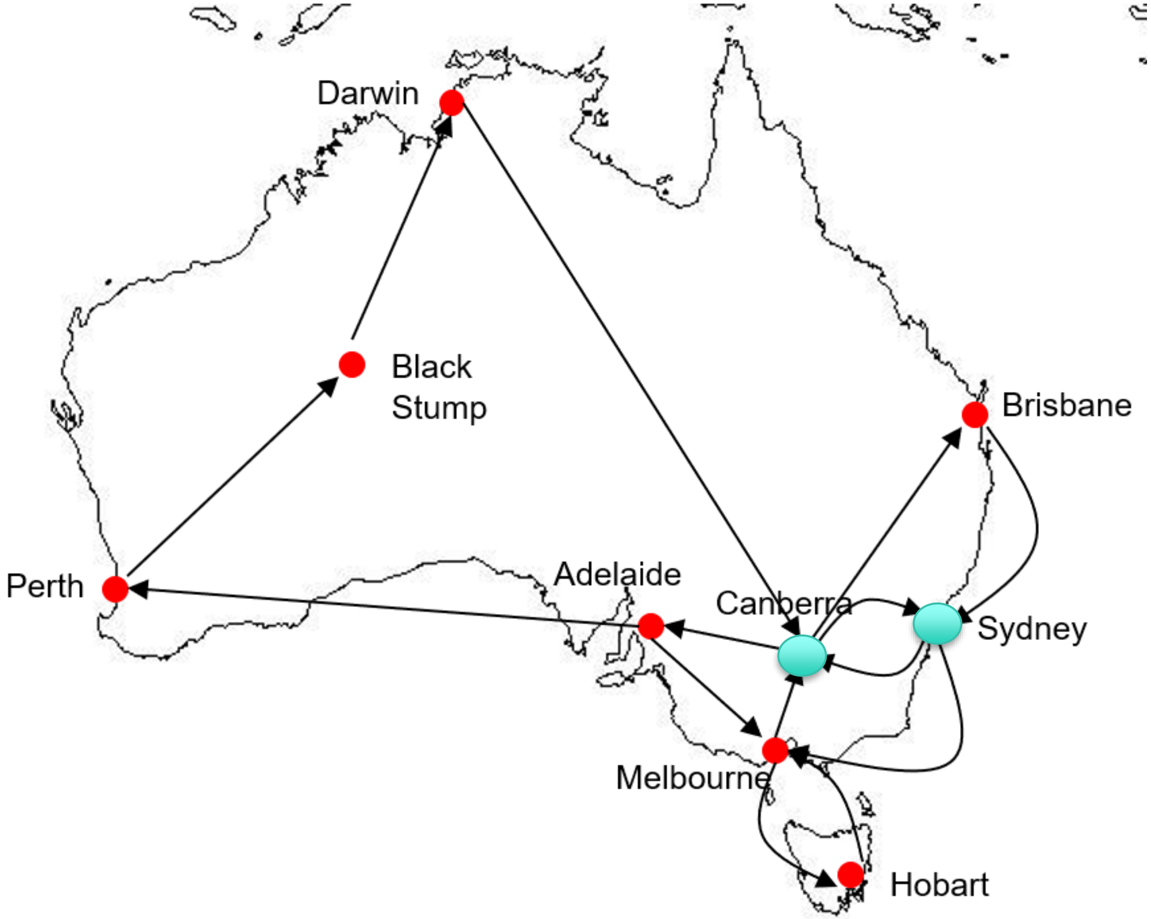


- BSF

Sydney
Canberra
Melbourne
Brisbane
Adelaide
Hobart
Perth
Black Stump
Darwin

- DSF
Sydney

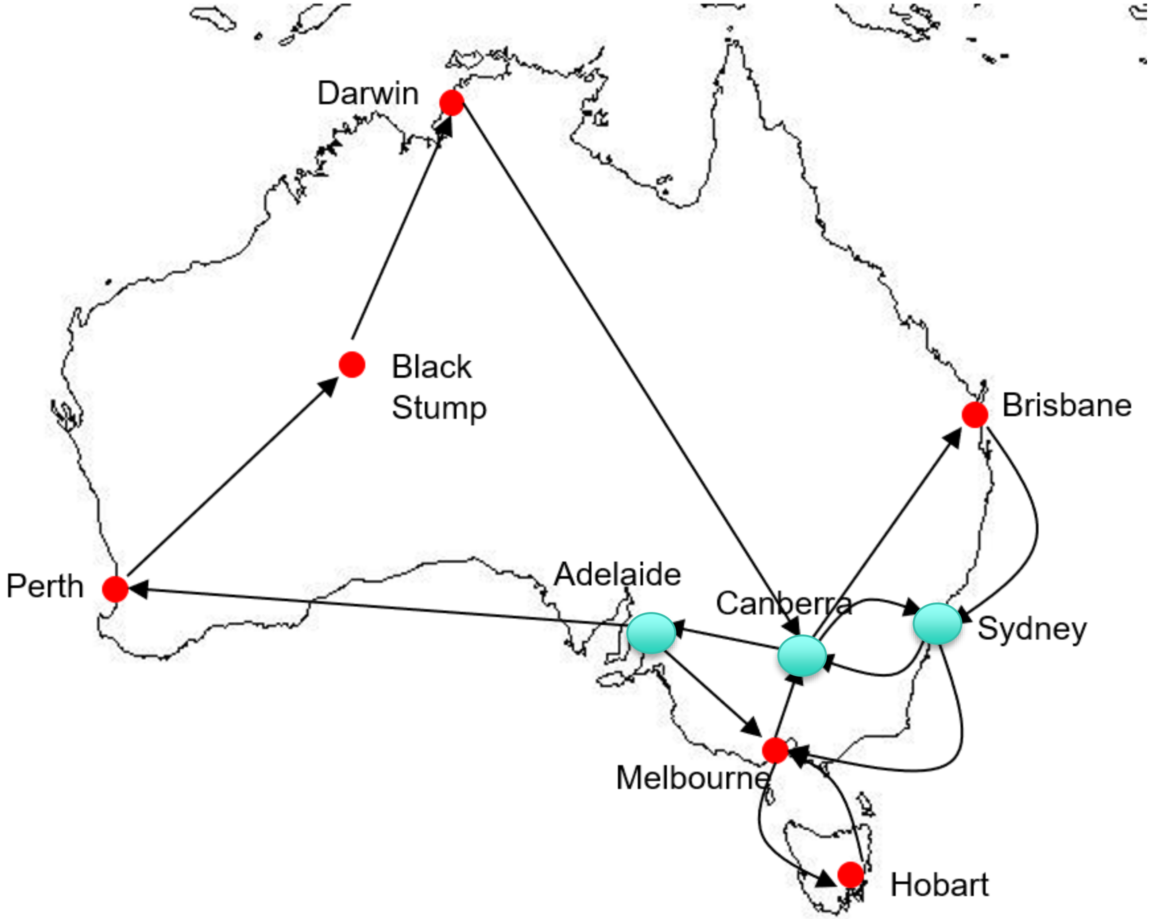

Darwin
Black
Stump
Brisbane
Perth
Adelaide
Canberra
Sydney
Melbourne
Hobart



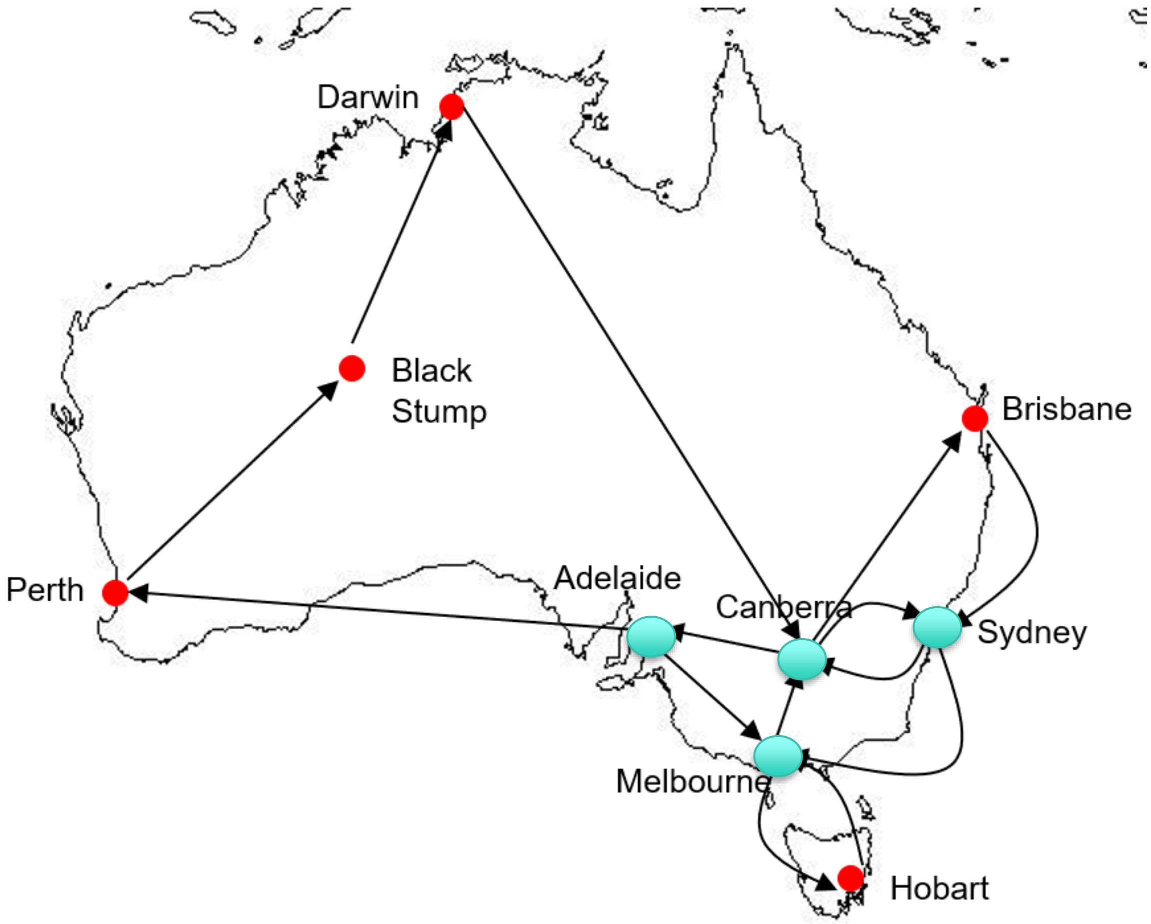
- DSF

Sydney

Canberra

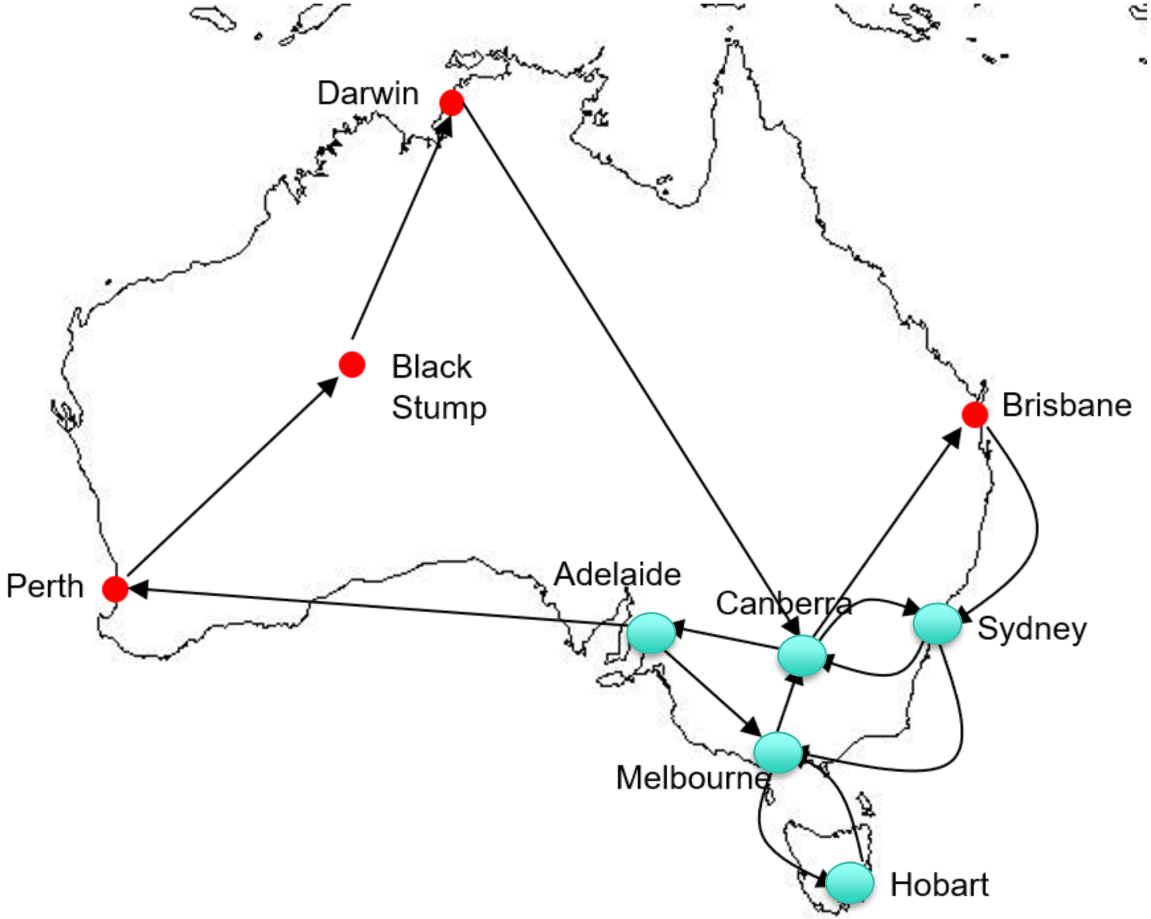


- DSF
Sydney
Canberra
Adelaide



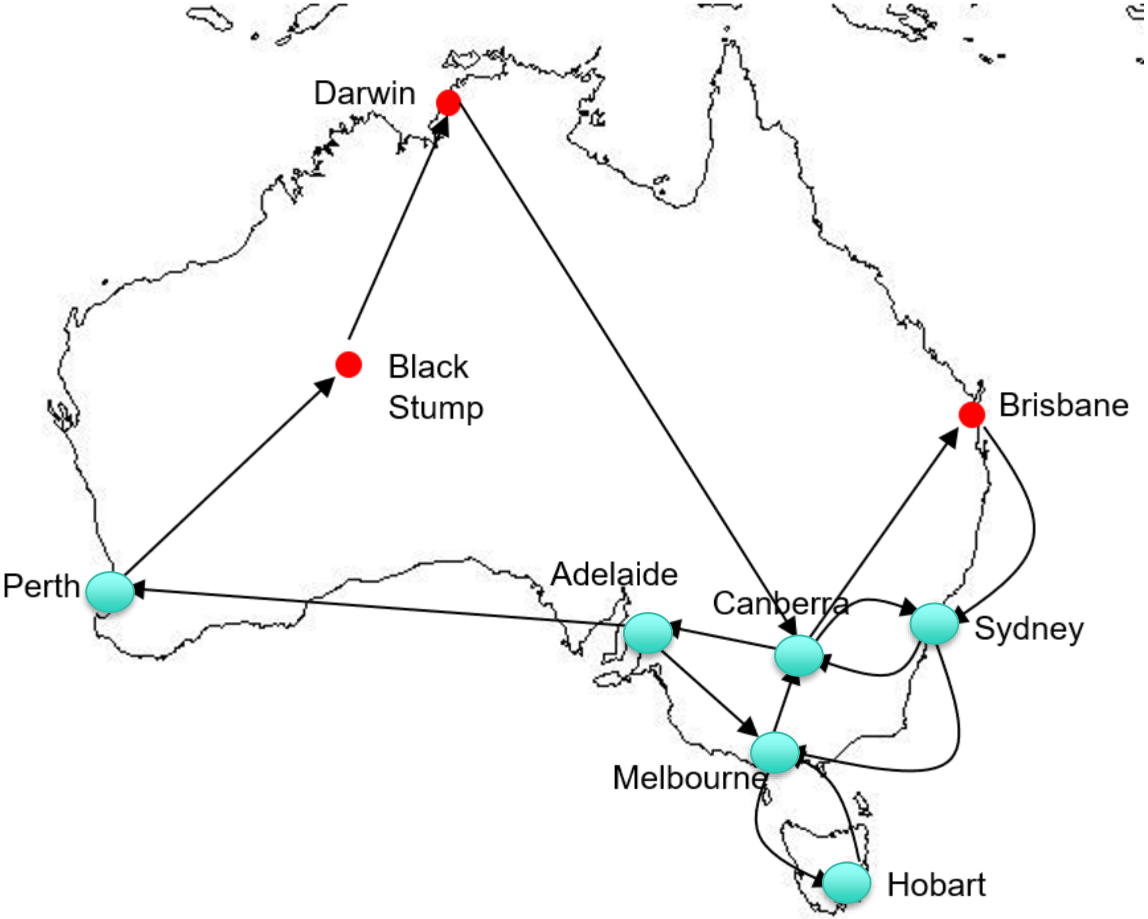
- DSF

Sydney
Canberra
Adelaide
Melbourne



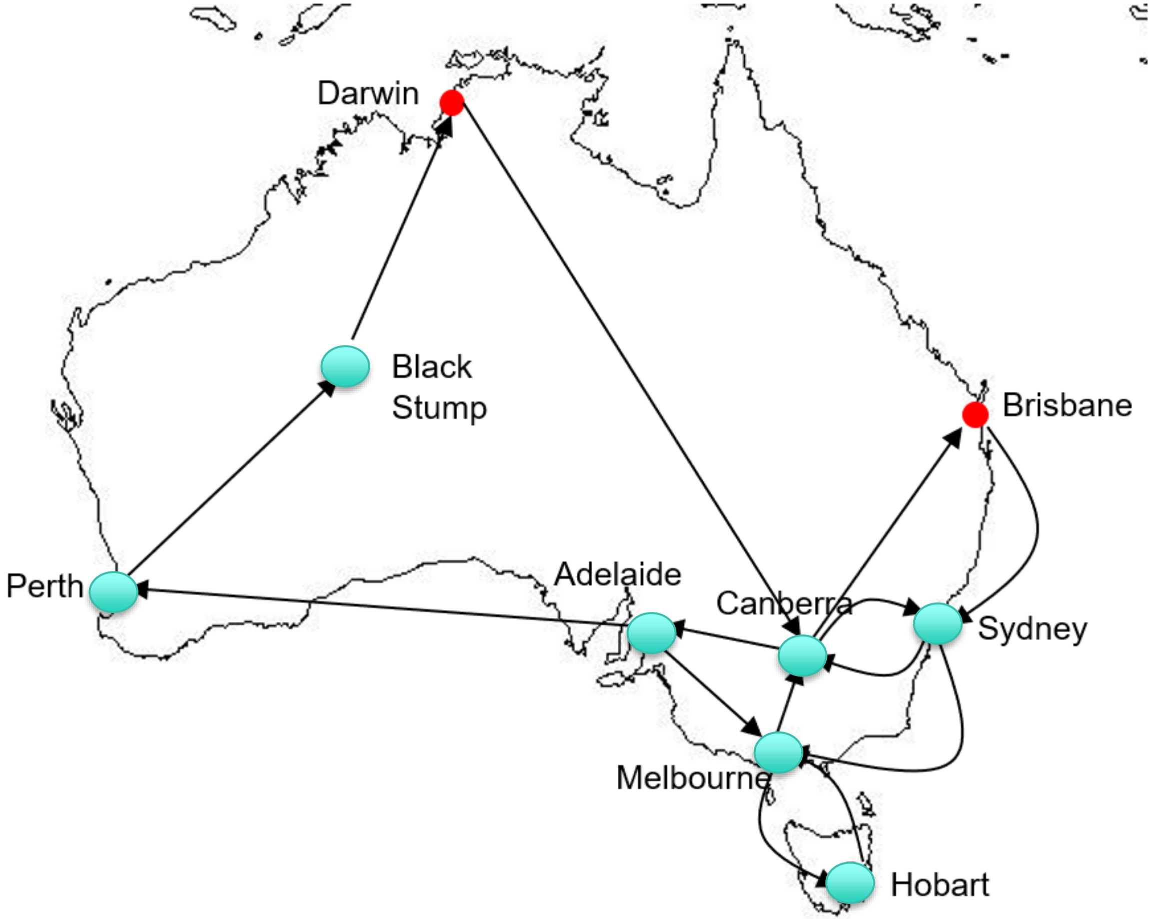
- DSF

Sydney
Canberra
Adelaide
Melbourne
Hobart



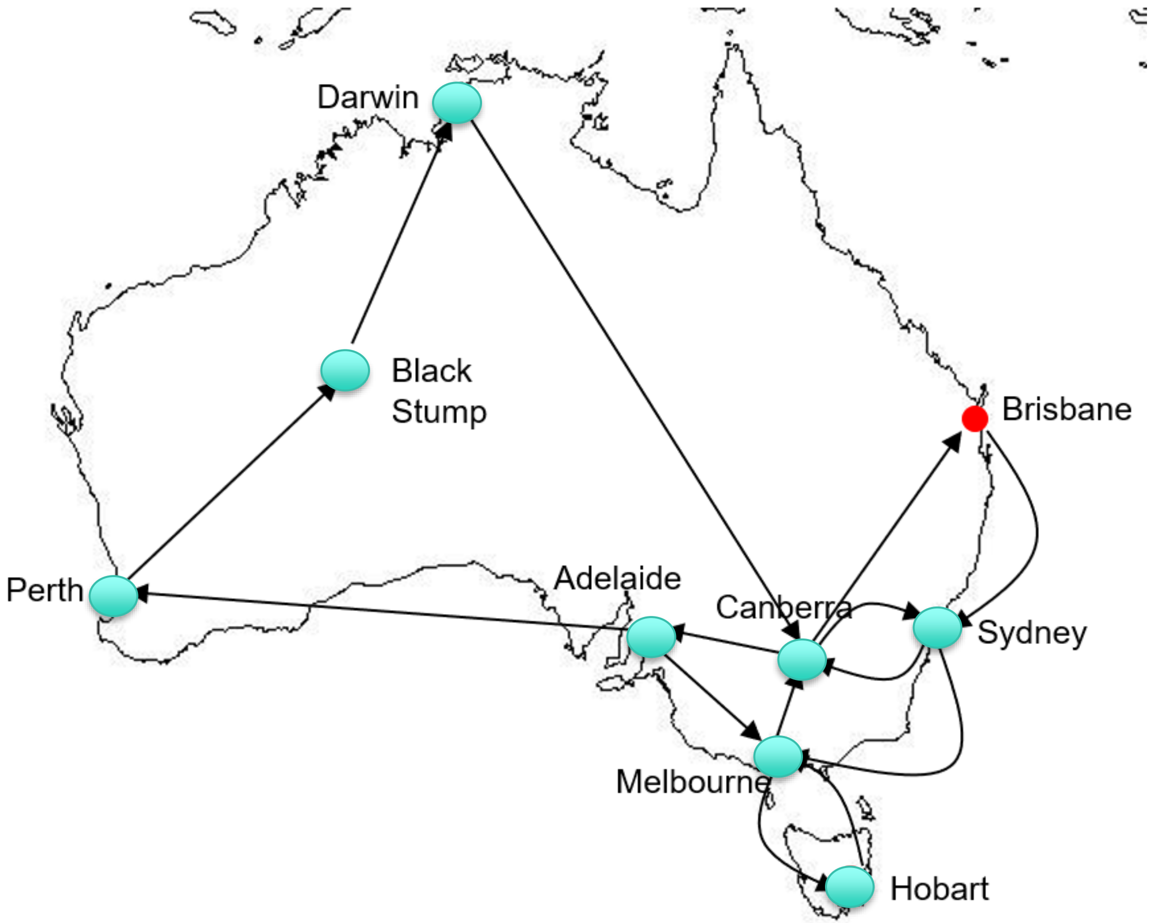
- DSF

Sydney
Canberra
Adelaide
Melbourne
Hobart
Perth



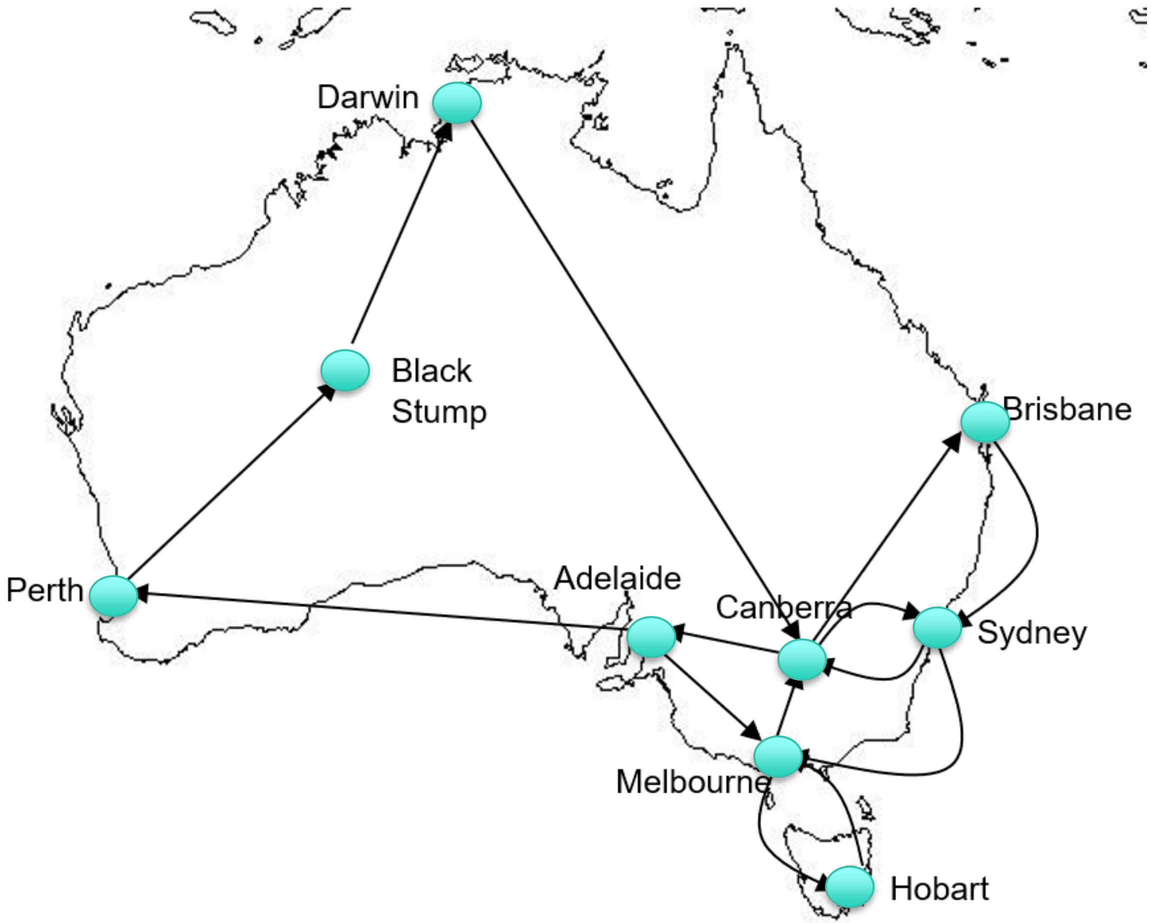
- DSF

Sydney
Canberra
Adelaide
Melbourne
Hobart
Perth
Black Strump



- DSF

Sydney
Canberra
Adelaide
Melbourne
Hobart
Perth
Black Strump
Darwin



- DSF

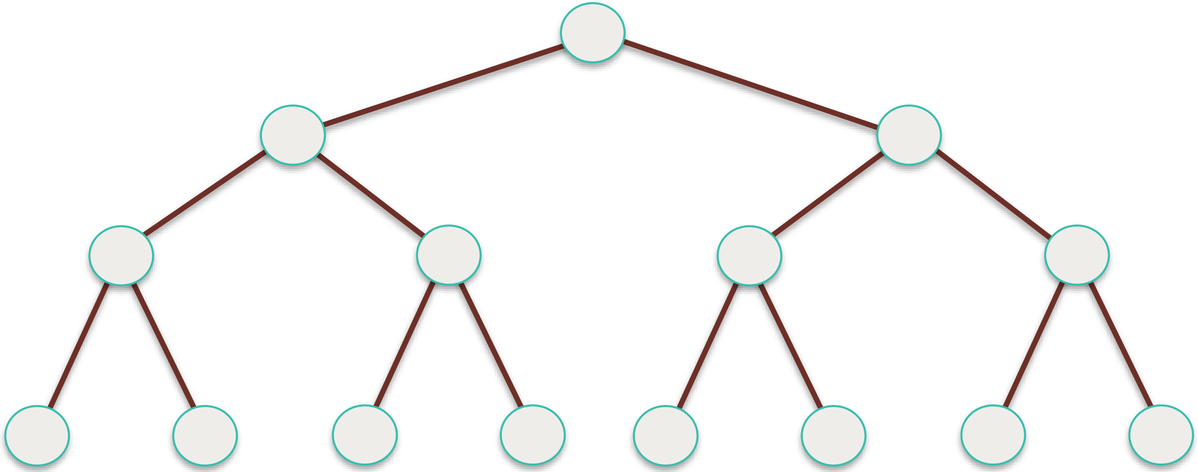
Sydney
Canberra
Adelaide
Melbourne
Hobart
Perth
Black Strump
Darwin
Brisbane

BFS and DFS Parallelism

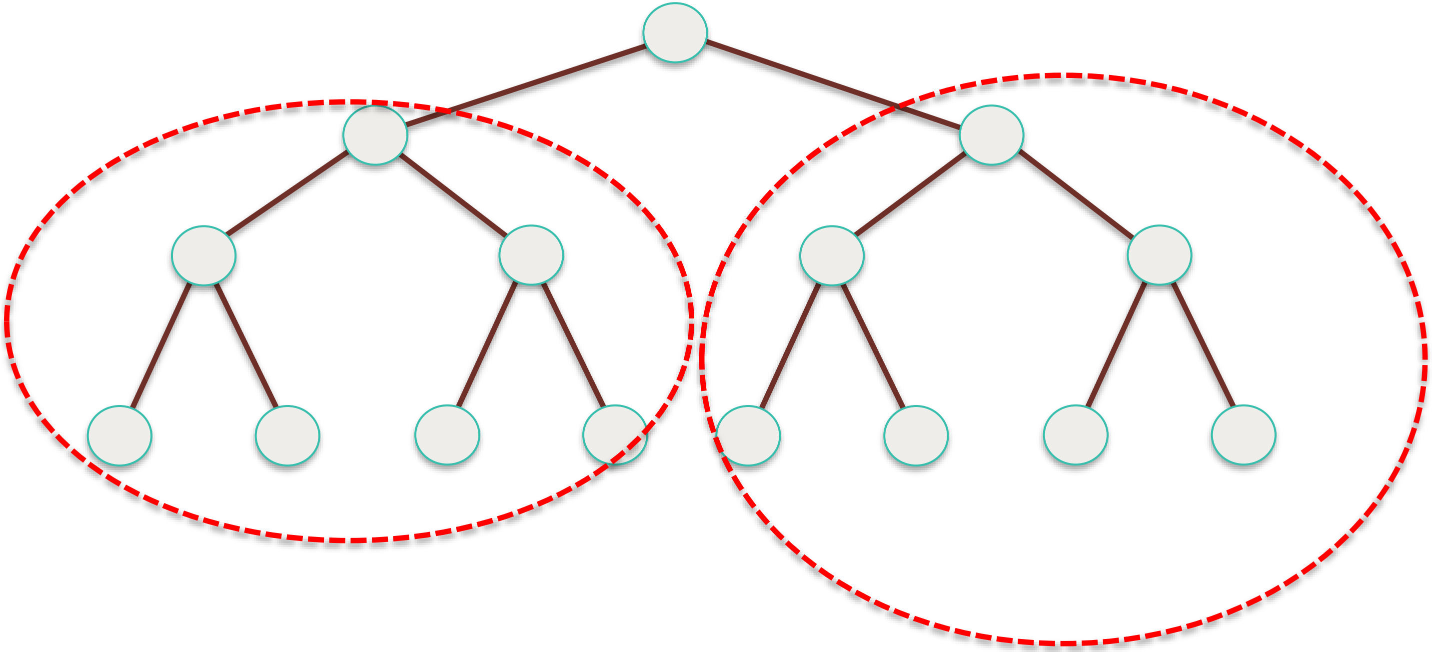
BSF and DFS Parallelism

- How the search space is splitted among processors?
- Different subtrees can be searched concurrently
- However, subtrees can be very different in size
- It is difficult to estimate the size of a subtree rooted at a node
- Dynamic load balancing is required

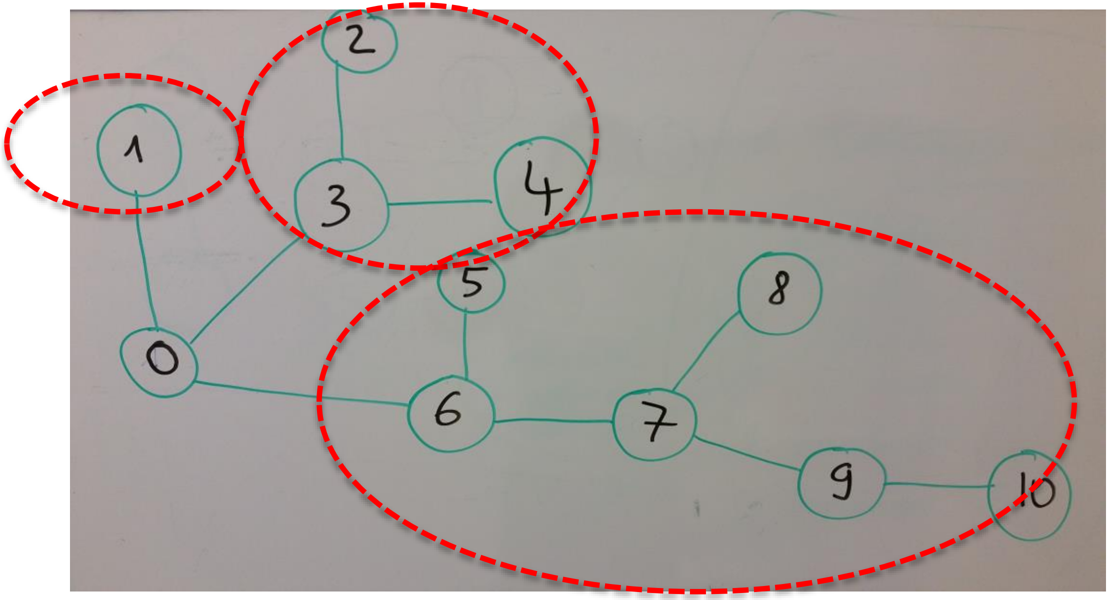
Breadth-First Search (BFS): Tree



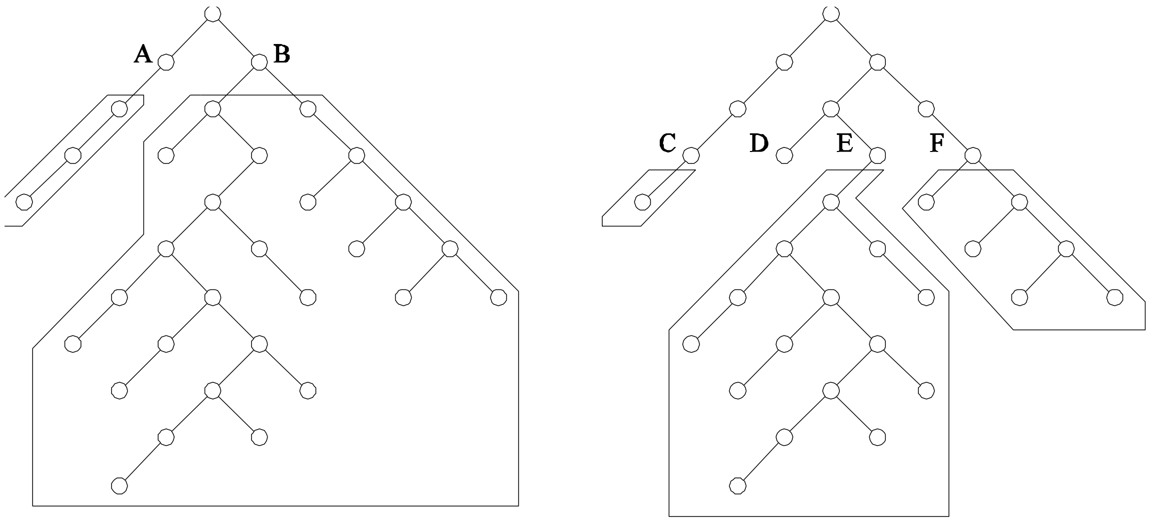
Breadth-First Search (BFS): Tree



BSF



DFS



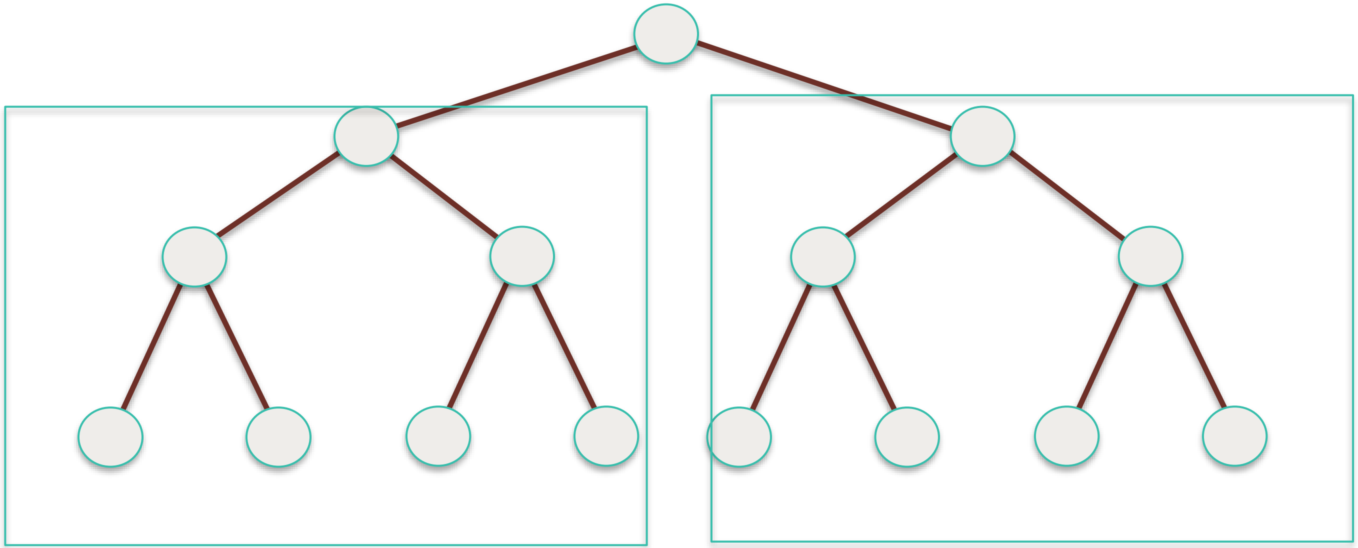
(a) (b)

The unstructured nature of tree search and the imbalance resulting from static partitioning

BSF Pseudo-code (parallel)

- Ideas:
- Process level in parallel
  - Synchronization is needed at each level
- Process an entire level in parallel
  - Load balancing is needed

Parallel Depth-First Search

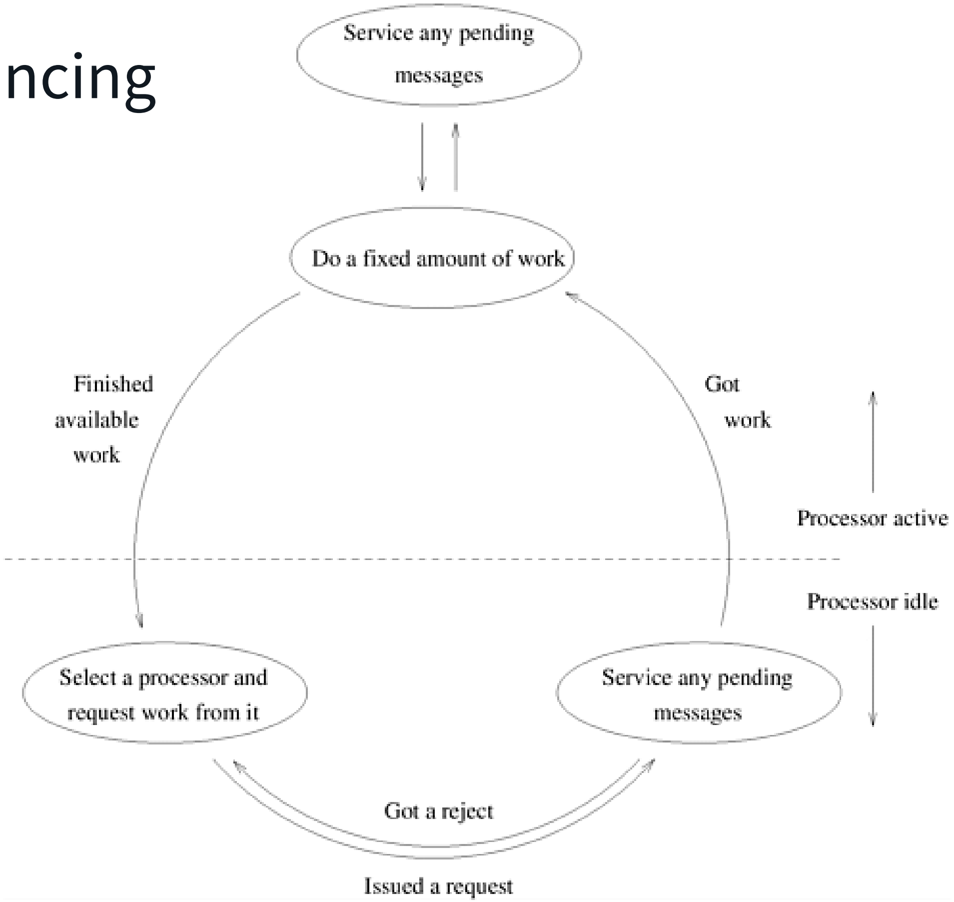


DFS: Dynamic Load Balancing

- When a processor runs out of work, it gets more work from another processor
- This is done using work requests and responses in message passing machines and locking and extracting work in shared address space machines
- On reaching final state at a processor, all processors terminate
- Unexplored states can be conveniently stored as local stacks at processors
- The entire space is assigned to one processor to begin with

DFS: Dynamic Load Balancing

Applied with distributed memory system

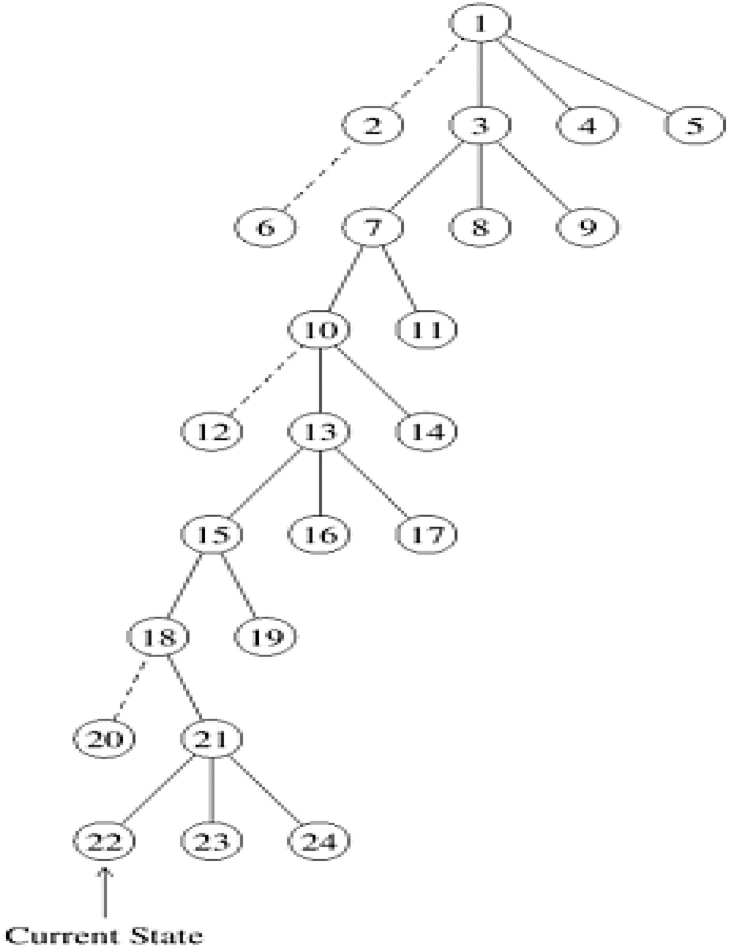


Parallel DFS: Work splitting

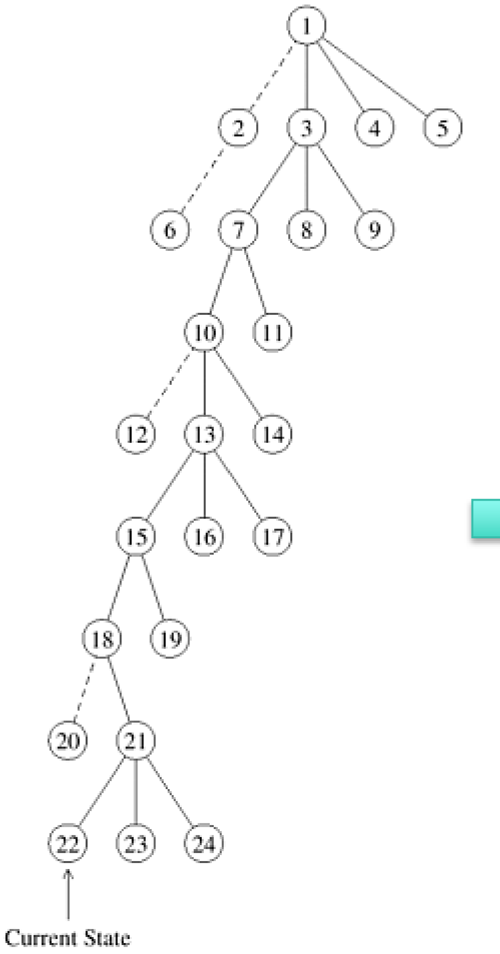
- Work is split by splitting the stack into two
- Ideally, we do not want either of the split pieces to be small
- Select nodes near the bottom of the stack (node splitting)
- Select some nodes from each level (stack splitting).
- The second strategy generally yields a more even split of the space

Parallel DFS: Work splitting

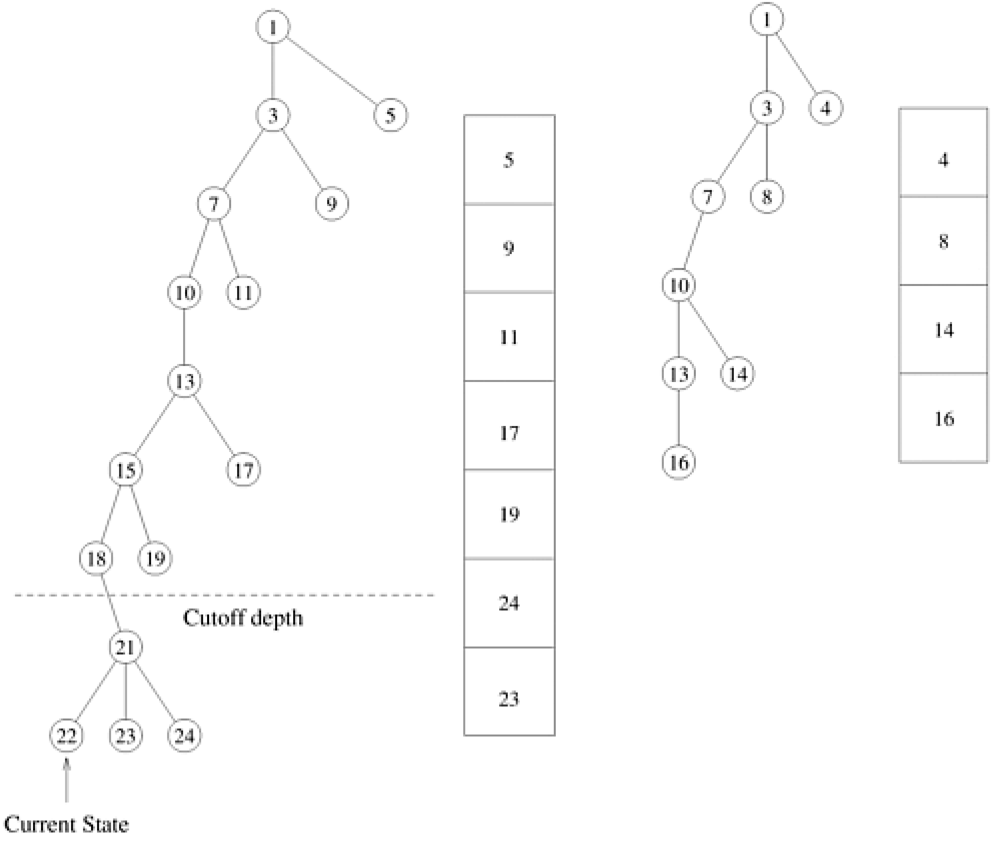
- Original works



(a)



(a)



(a) (b)

Splitting the DFS tree: the two subtrees along with their stack representations are shown in (a) and (b).

Parallel Depth-First Search: Pseudocode

```
DFS-recursive(G, s):
    mark s as visited
    for all neighbours w of s in Graph G:
        if w is not visited:
            DFS-recursive(G, w)
```

References

[1] *Introduction to Parallel Computing,* Ananth Grama, George Karypis, Vipin Kumar, Anshul Gupta, Addison Wesley, 2003.


Q&A