INTRODUCTION to ARTIFICIAL INTELLIGENCE
2015-2016 FALL SEMESTER
LABORATORY MANUAL

Experiment 4
Uninformed Search Strategies

Uninformed strategies use only the information available in the problem definition. Uninformed Search includes the following algorithms:

- Breadth First Search (BFS)
- Depth First Search (DFS)
- Uniform Cost Search (UCS)
- Depth Limited Search (DLS)
- Iterative Deepening Search (IDS)
- Bidirectional Search (BS)

BFS

- Expand shallowest unexpanded node.
- In breadth-first search the frontier is implemented as a **FIFO** (first-in, first-out) queue. Thus, the path that is selected from the frontier is the one that was added earliest.
- This approach implies that the paths from the start node are generated in order of the number of arcs in the path. One of the paths with the fewest arcs is selected at each stage.

Pseudocode:

Input: A graph G and a starting vertex v of G

Output: All vertices reachable from v labeled as explored.

A non-recursive implementation of breadth-first search:

```
1 Breadth-First-Search(G, v):
2
3     for each node n in G:
4         n.distance = INFINITY
5         n.parent = NIL
6
7     create empty queue Q
8
9     v.distance = 0
10    Q.enqueue(v)
11
12    while Q is not empty:
13
14        u = Q.dequeue()
15
16        for each node n that is adjacent to u:
17            if n.distance == INFINITY:
18                n.distance = u.distance + 1
19                n.parent = u
20                Q.enqueue(n)
```

DFS

- Expand deepest unexpanded node.
- The first strategy is depth-first search. In depth-first search, the frontier acts like a last-in first-out (**LIFO**) stack. The elements are added to the stack one at a time. The one selected and taken off the frontier at any time is the last element that was added.

Pseudocode:

Input: A graph G and a vertex v of G

Output: All vertices reachable from v labeled as discovered.

A recursive implementation of DFS:

```
1  procedure DFS(G,v):
2      label v as discovered
3      for all edges w in G.adjacentEdges(v) do
4          if vertex w is not labeled as discovered
then
5              recursively call DFS(G,w)
```

A non-recursive implementation of DFS:

```
1  procedure DFS-iterative(G,v):
2      let S be a stack
3      S.push(v)
4      while S is not empty
5            v = S.pop()
6            if v is not labeled as discovered:
7                label v as discovered
8                for all edges from v to w in
G.adjacentEdges(v) do
9                    S.push(w)
```

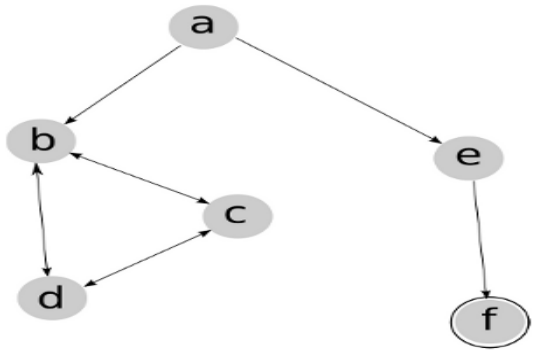

Fig 1.

Exercise 1:

Write a program to implement BFS in Fig 1. Show which sequences of paths are explored by BFS.

Exercise 2:

Write a program to implement DFS in Fig 1. Show which sequences of paths are explored by DFS.