**Exp 27: Write a Prolog Program to implement Best First Search algorithm**

**Input:**

% Heuristic values (lower is better)

h(a, 10).

h(b, 8).

h(c, 5).

h(d, 7).

h(e, 3).

h(f, 6).

h(g, 0). % Goal

% Graph edges

edge(a, b).

edge(a, c).

edge(b, d).

edge(b, e).

edge(c, f).

edge(e, g).

edge(f, g).

% Best First Search

best\_first(Start, Goal, Path) :-

search([[Start]], Goal, Path).

% If first path reaches Goal

search([[Goal|Rest]|\_], Goal, [Goal|Rest]).

% Expand first path, pick next best

search([[Node|Rest]|Others], Goal, Path) :-

findall([Next,Node|Rest],

(edge(Node, Next), \+ member(Next, [Node|Rest])),

NewPaths),

add\_and\_sort(NewPaths, Others, Sorted),

search(Sorted, Goal, Path).

% Add new paths and sort by heuristic

add\_and\_sort(New, Old, Sorted) :-

append(New, Old, All),

sort\_paths(All, Sorted).

sort\_paths(Paths, Sorted) :-

map\_list\_to\_pairs(path\_score, Paths, Pairs),

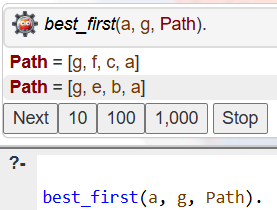
keysort(Pairs, SortedPairs),

pairs\_values(SortedPairs, Sorted).

path\_score([Node|\_], Score) :-

h(Node, Score).

**Output:**

****