Program 27

Best first search

CODE:

```
% Define edges between nodes and their costs
edge(a, b, 3).
edge(a, c, 6).
edge(b, d, 2).
edge(b, e, 5).
edge(c, f, 9).
edge(d, g, 1).
edge(e, h, 4).
edge(f, i, 7).
edge(g, j, 8).
edge(h, k, 10).

% Define heuristic values for nodes
heuristic(a, 12).
heuristic(b, 8).
heuristic(c, 14).
heuristic(d, 6).
heuristic(e, 10).
heuristic(f, 9).
```

```
heuristic(g, 4).
heuristic(h, 5).
heuristic(i, 3).
heuristic(j, 7).
heuristic(k, 0).

% Best First Search Algorithm
best_first_search(Start, Goal, Path) :-
    best_first_search([[Start]], Goal, Path).

best_first_search([[Goal|Path]|_], Goal, [Goal|Path]).

best_first_search([Path|Paths], Goal, FinalPath) :-
    extend(Path, NewPaths),
    append(Paths, NewPaths, Paths1),
    sort_paths(Paths1, SortedPaths),
    best_first_search(SortedPaths, Goal, FinalPath).

extend([Node|Path], NewPaths) :-
    findall([NewNode, Node|Path],
            (edge(Node, NewNode, _), not(member(NewNode,
Path))),
```

NewPaths).

```
sort_paths(Paths, SortedPaths) :-
    predsort(compare_heuristic, Paths, SortedPaths).

compare_heuristic(Result, Path1, Path2) :-
    heuristic_estimate(Path1, Estimate1),
    heuristic_estimate(Path2, Estimate2),
    compare(Result, Estimate1, Estimate2).

heuristic_estimate([Node|_], Estimate) :-
    heuristic(Node, Estimate).
```

OUTPUT: