```
% Das Programm wird mit solve(depth), solve(breadth) oder solve(informed) aufgerufen.
solve(Strategy):-
start_description(StartState),
solve((start,StartState,_),Strategy).


% Prädikat search:
% 1. Argument ist die Liste aller Pfade. Der aktuelle Pfad ist an erster Stelle.
% Jeder Pfad ist als Liste von Zuständen repräsentiert, allerdings in falscher
% Reihenfolge, d.h. der Startzustand ist an letzter Position.
% 2. Argument ist die Strategie
% 3. Argument ist der Ergebnis-Pfad.
%
solve(StartNode,Strategy) :-
start_node(StartNode),
search([[StartNode]],Strategy,Path),
reverse(Path,Path_in_correct_order),
write_solution(Path_in_correct_order).



write_solution(Path):-
nl,write('SOLUTION:'),nl,
write_actions(Path).

write_actions([]).

write_actions([(Action,_,_)|Rest]):-
write('Action: '),write(Action),nl,
write_actions(Rest).




% Abbruchbedingung: Wenn ein Zielzustand erreicht ist, wird der aktuelle Pfad an den
% dritten Parameter übertragen.
%
search([[FirstNode|Predecessors]|_],_,[FirstNode|Predecessors]) :-
goal_node(FirstNode),
nl,write('SUCCESS'),nl,!.


search([[FirstNode|Predecessors]|RestPaths],Strategy,Solution) :-
expand(FirstNode,Children),                                    % Nachfolge-Zustände berechnen
generate_new_paths(Children,[FirstNode|Predecessors],NewPaths), % Nachfolge-Zustände einbauen
insert_new_paths(Strategy,NewPaths,RestPaths,AllPaths), % Neue Pfade einsortieren
```


```
% Das Programm wird mit solve(depth), solve(breadth) oder solve(informed) aufgerufen.
solve(Strategy):-
start_description(StartState),
solve((start,StartState,_),Strategy).


% Prädikat search:
% 1. Argument ist die Liste aller Pfade. Der aktuelle Pfad ist an erster Stelle.
% Jeder Pfad ist als Liste von Zuständen repräsentiert, allerdings in falscher
% Reihenfolge, d.h. der Startzustand ist an letzter Position.
% 2. Argument ist die Strategie
% 3. Argument ist der Ergebnis-Pfad.
%
solve(StartNode,Strategy) :-
start_node(StartNode),
search([[StartNode]],Strategy,Path),
reverse(Path,Path_in_correct_order),
write_solution(Path_in_correct_order).



write_solution(Path):-
nl,write('SOLUTION:'),nl,
write_actions(Path).

write_actions([]).

write_actions([(Action,_,_)|Rest]):-
write('Action: '),write(Action),nl,
write_actions(Rest).




% Abbruchbedingung: Wenn ein Zielzustand erreicht ist, wird der aktuelle Pfad an den
% dritten Parameter übertragen.
%
search([[FirstNode|Predecessors]|_],_,[FirstNode|Predecessors]) :-
goal_node(FirstNode),
nl,write('SUCCESS'),nl,!.


search([[FirstNode|Predecessors]|RestPaths],Strategy,Solution) :-
expand(FirstNode,Children),                                    % Nachfolge-Zustände berechnen
generate_new_paths(Children,[FirstNode|Predecessors],NewPaths), % Nachfolge-Zustände einbauen
insert_new_paths(Strategy,NewPaths,RestPaths,AllPaths), % Neue Pfade einsortieren
```

```
search(AllPaths,Strategy,Solution).
```

```
generate_new_paths(Children,Path,NewPaths):-
maplist(get_state,Path,States),
generate_new_paths_help(Children,Path,States,NewPaths).
```

```
% Abbruchbedingung, wenn alle Kindzustände abgearbeitet sind.
%
generate_new_paths_help([],_,_,[]).
```

```
% Falls der Kindzustand bereits im Pfad vorhanden war, wird der gesamte Pfad verworfen,
% denn er würde nur in einem Zyklus enden. (Dies betrifft nicht die Fortsetzung des
% Pfades mit Geschwister-Kindern.) Es wird nicht überprüft, ob der Kindzustand in einem
% anderen Pfad vorkommt, denn möglicherweise ist dieser Weg der günstigere.
%
generate_new_paths_help([FirstChild|RestChildren],Path,States,RestNewPaths):-
get_state(FirstChild,State),state_member(State,States),!,
generate_new_paths_help(RestChildren,Path,States,RestNewPaths).
```

```
% Ansonsten, also falls der Kindzustand noch nicht im Pfad vorhanden war, wird er als
% Nachfolge-Zustand eingebaut.
%
generate_new_paths_help([FirstChild|RestChildren],Path,States,
[[FirstChild|Path]|RestNewPaths]):-
generate_new_paths_help(RestChildren,Path,States,RestNewPaths).
```


```
search(AllPaths,Strategy,Solution).
```

```
generate_new_paths(Children,Path,NewPaths):-
maplist(get_state,Path,States),
generate_new_paths_help(Children,Path,States,NewPaths).
```

```
% Abbruchbedingung, wenn alle Kindzustände abgearbeitet sind.
%
generate_new_paths_help([],_,_,[]).
```

```
% Falls der Kindzustand bereits im Pfad vorhanden war, wird der gesamte Pfad verworfen,
% denn er würde nur in einem Zyklus enden. (Dies betrifft nicht die Fortsetzung des
% Pfades mit Geschwister-Kindern.) Es wird nicht überprüft, ob der Kindzustand in einem
% anderen Pfad vorkommt, denn möglicherweise ist dieser Weg der günstigere.
%
generate_new_paths_help([FirstChild|RestChildren],Path,States,RestNewPaths):-
get_state(FirstChild,State),state_member(State,States),!,
generate_new_paths_help(RestChildren,Path,States,RestNewPaths).
```

```
% Ansonsten, also falls der Kindzustand noch nicht im Pfad vorhanden war, wird er als
% Nachfolge-Zustand eingebaut.
%
generate_new_paths_help([FirstChild|RestChildren],Path,States,
[[FirstChild|Path]|RestNewPaths]):-
generate_new_paths_help(RestChildren,Path,States,RestNewPaths).
```

```
97
98
99 get_state((_,State,_),State).
100
101
102
103 %%% Strategie:
104
105 write_action([[(Action,_)|_]|_]):-
106 nl,write('Action: '),write(Action),nl.
107
108 write_next_state([[_,(_,State)|_]|_]):-
109 nl,write('Go on with: '),write(State),nl.
110
111 write_state([[(_,State)|_]|_]):-
112 write('New State: '),write(State),nl.
113
114 write_fail(depth,[[(_,State)|_]|_]):-
115 nl,write('FAIL, go on with: '),write(State),nl.
116
117 write_fail(_,_):- nl,write('FAIL').
118
119 % Alle Strategien: Keine neuen Pfade vorhanden
120 insert_new_paths(Strategy,[],OldPaths,OldPaths):-
121 write_fail(Strategy,OldPaths),!.
122
123 % Tiefensuche
124 insert_new_paths(depth,NewPaths,OldPaths,AllPaths):-
125 append(NewPaths,OldPaths,AllPaths),
126 write_action(NewPaths).
127
128 % Breitensuche
129 insert_new_paths(breadth,NewPaths,OldPaths,AllPaths):-
130 append(OldPaths,NewPaths,AllPaths),
131 write_next_state(AllPaths),
132 write_action(AllPaths).
133
134
135 % Informierte Suche
```

```
97
98
99 get_state((_,State,_),State).
100
101
102
103 %%% Strategie:
104
105 write_action([[(Action,_)|_]|_]):-
106 nl,write('Action: '),write(Action),nl.
107
108 write_next_state([[_,(_,State)|_]|_]):-
109 nl,write('Go on with: '),write(State),nl.
110
111 write_state([[(_,State)|_]|_]):-
112 write('New State: '),write(State),nl.
113
114 write_fail(depth,[[(_,State)|_]|_]):-
115 nl,write('FAIL, go on with: '),write(State),nl.
116
117 write_fail(_,_):- nl,write('FAIL').
118
119 % Alle Strategien: Keine neuen Pfade vorhanden
120 insert_new_paths(Strategy,[],OldPaths,OldPaths):-
121 write_fail(Strategy,OldPaths),!.
122
123 % Tiefensuche
124 insert_new_paths(depth,NewPaths,OldPaths,AllPaths):-
125 append(NewPaths,OldPaths,AllPaths),
126 write_action(NewPaths).
127
128 % Breitensuche
129 insert_new_paths(breadth,NewPaths,OldPaths,AllPaths):-
130 append(OldPaths,NewPaths,AllPaths),
131 write_next_state(AllPaths),
132 write_action(AllPaths).
133
134 % Optimistisches Bergsteigen
135 insert_new_paths(hill_climbing,NewPaths,_,[BestPath]):-
136 eval_paths(NewPaths),
137 insert_new_paths_informed(NewPaths,[],[BestPath|_]),
138 write_action([BestPath]),
139 write_state([BestPath]).
140
141 % Optimistisches Bergsteigen mit Backtracking
142 insert_new_paths(hill_climbing_bt,NewPaths,OldPaths,AllPaths):-
143 eval_paths(NewPaths),
144 insert_new_paths_informed(NewPaths,[],SortedPaths),
145 append(SortedPaths,OldPaths,AllPaths),
146 write_action(SortedPaths),
147 write_state(SortedPaths).
148
149 % Informierte Suche
150 % (A und gierige Bestensuche)
```

```
insert_new_paths(informed,NewPaths,OldPaths,AllPaths):-
eval_paths(NewPaths),
insert_new_paths_informed(NewPaths,OldPaths,AllPaths),
write_action(AllPaths),
write_state(AllPaths).
```


```
insert_new_paths(informed,NewPaths,OldPaths,AllPaths):-
eval_paths(NewPaths),
insert_new_paths_informed(NewPaths,OldPaths,AllPaths),
write_action(AllPaths),
write_state(AllPaths).
```