Experiment No. 8: Partial Order Planning for block world problem

Aim: Implementation of Partial Order Planning for block world problem

Theory:

(a) Write about Partial Order Planning, its application

Ans: Any algorithm that can place two actions into a plan without specifying which comes first is called Partial Order Planning. Partial Order Planning works on several sub-goals independently & solves them with several sub-plans & thereafter combines them. With Partial Order Planning, problems can be decomposed so it can work well with non-cooperative environments. Thus, it is a plan which specifies all actions that need to be taken, but only specifies the order between actions when necessary. It uses a least commitment strategy. It consists of two states: “start” & “finish” where,

1. Start: No preconditions & Effects are initial states.
2. Finish: No effects & Preconditions are goals.

Applications:

1. Scheduling problems with action choices as well as resource handling requirements Problems in supply chain management

HSTS (Hubble Space Telescope scheduler)

Workflow management

2. Autonomous agents

RAX/PS (The NASA Deep Space planning agent)

3. Software module integrators

Test case generation (Pittsburgh)

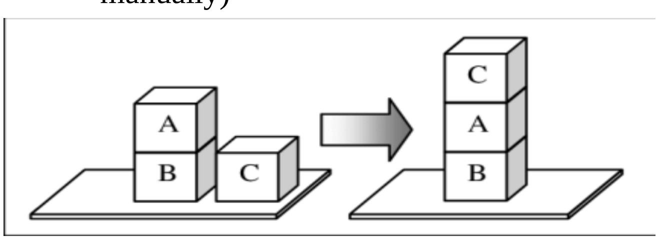
4. decision support

Monitoring sub goal interactions

5. Plan-based interfaces

(b)Problem formulation for block world problem

(c) Create a partial order plan for a given block world problem scenario. (solve the numerical manually)



Algorithm:

1. Make the initial plan, i.e. the one that contains only the Start and Finish steps.

2. Do until you have a solution plan
    a. Take an unachieved precondition from the plan; achieve it.
    b. Resolve any threats using promotion or demotion

```
STRIPS(init-state, goals, ops)
   Let current-state be init-state;
   For each goal in goals do
      If goal cannot be proven in current state
         Pick an operator instance, op, s.t. goal ∈ adds(op);
         ;; Solve preconditions
         STRIPS(current-state, preconds(op), ops);
         ;; Apply operator
         current-state := current-state + adds(op) - dels(ops);
   ;; Patch any clobbered goals
   Let rgoals be any goals which are not provable in
   current-state;
   STRIPS(current-state, rgoals, ops).
```

Implementation:

Code:

```
#include <bits/stdc++.h>
#define print(n) cout<<n<<endl
#define pb push_back
#define f first
#define s second

using namespace std;

void explore(vector<vector<int>> &vi,vector<vector<int>> &vf){
    int vi_sz=vi.size();
    int vf_sz=vf.size();
    vector<int> moveable(vi_sz,-1);
    int c=0;
    for (auto it: vi){
        int j=0;
        while(j<it.size() && it[j]==vf[0][j]){
            j++;
        }
        c++;
        //if(j==it.size()) continue;
        moveable[c-1]=j;
    }
    set<int> unst;

    for(int i=0;i<vi_sz;i++){
        int j=vi[i].size()-1;
        while(j>=0 && j>=moveable[i]){
            if(vi[i][j]!=0){
```

```
            unst.insert(vi[i][j]);
            if(j>0) print("Unstack: "<<vi[i][j]);
          }
          j--;
        }
    }

    int n=vf[0].size();
    for(int i=0;i<n;i++){
        if(unst.find(vf[0][i])!=unst.end()){
            print("Pick: "<<vf[0][i]);
            print("Stack: "<<vf[0][i]);
        }
    }
}

int main()
{
    int n,ir,ic,k;
    print("Enter Number of Blocks: ");
    cin>>n;
    print("For Initial State");
    print("Enter Number of Columns: ");
    cin>>ic;
    vector<vector<int>> vi;
    for(int i=0;i<ic;i++){
        print("Enter Stack "<<i+1<<" Size: ");
        int sz;
        cin>>sz;
        print("Enter "<<sz<<" Space Seperated arrangements from Ground");
        vector<int> st;
        for (int j=0;j<sz;j++){
            cin>>k;
            st.pb(k);
        }
        vi.pb(st);
    }
    for (auto it: vi){
        for (auto x: it){
            cout<<x<<" ";
        }
        cout<<endl;
    }
    int fr,fc=1;
    print("For Final State");
    vector<vector<int>> vf;
    for(int i=0;i<fc;i++){
        print("Enter Stack Size for a Single Column: ");
        int sz;
```

```
        cin>>sz;
        print("Enter "<<sz<<" Space Seperated arrangements from Ground");
        vector<int> st;
        for (int j=0;j<sz;j++){
            cin>>k;
            st.pb(k);
        }
        vf.pb(st);
    }
    explore(vi,vf);
    return 0;
}
```

Output:

```
Enter Number of Blocks:
3
For Initial State
Enter Number of Columns:
2
Enter Stack 1 Size:
2
Enter 2 Space Seperated arrangements from Ground
2 1
Enter Stack 2 Size:
1
Enter 1 Space Seperated arrangements from Ground
3
2 1
3
For Final State
Enter Stack Size for a Single Column:
3
Enter 3 Space Seperated arrangements from Ground
2 1 3
Pick: 3
Stack: 3


...Program finished with exit code 0
Press ENTER to exit console.
```

```
Enter Number of Blocks:
6
For Initial State
Enter Number of Columns:
3
Enter Stack 1 Size:
3
Enter 3 Space Seperated arrangements from Ground
2 1 3
Enter Stack 2 Size:
2
Enter 2 Space Seperated arrangements from Ground
4 5
Enter Stack 3 Size:
1
Enter 1 Space Seperated arrangements from Ground
6
2 1 3
4 5
6
For Final State
Enter Stack Size for a Single Column:
6
Enter 6 Space Seperated arrangements from Ground
1 2 3 4 5 6
```

```
2 1 3
4 5
6
For Final State
Enter Stack Size for a Single Column:
6
Enter 6 Space Seperated arrangements from Ground
1 2 3 4 5 6
Unstack: 3
Unstack: 1
Unstack: 5
Pick: 1
Stack: 1
Pick: 2
Stack: 2
Pick: 3
Stack: 3
Pick: 4
Stack: 4
Pick: 5
Stack: 5
Pick: 6
Stack: 6
```

Conclusion: Write your basic understanding about partial order planning.