Bahria University,
Karachi Campus

LAB EXPERIMENT NO.
09

LIST OF TASKS

TASK NO	OBJECTIVE
1.	Configure the given AI planner in your system
2.	Provide the solution for dockworkerrobot problem using AI Planne

Submitted On:
6/7/2024

TASK NO 1: GUI-based application that leverages transformer-based models to provide text summarization, image classification, speech recognition, and audio sentiment analysis.

```
var strips = require('strips');
strips.verbose = true;
strips.load('./examples/dockworkerrobot/domain.txt', './examples/dockworkerrobot/problem.txt', function(domain, problem) {
   var solutions = strips.solve(domain, problem, cost);
   var solution = solutions[0];
   console.log('- Solution found in ' + solution.steps + ' steps!');
   for (var i = 0; i < solution.path.length; i++) {
      console.log((i + 1) + '. ' + solution.path[i]);}});
function cost(state) {
   var cost = 120;
   for (var i in state.actions) {
      var action = state.actions[i].action;
      if (action == 'in') {
         if (state.actions[i].parameters.indexOf('ca') != -1 && state.actions[i].parameters.indexOf('p2') != -1) {
            cost -= 20; }
         else if (state.actions[i].parameters.indexOf('cb') != -1 && state.actions[i].parameters.indexOf('q2') != -1) {
            cost -= 20; }
         else if (state.actions[i].parameters.indexOf('cc') != -1 && state.actions[i].parameters.indexOf('p2') != -1) {
            cost -= 20; }
         else if (state.actions[i].parameters.indexOf('cd') != -1 && state.actions[i].parameters.indexOf('q2') != -1) {
            cost -= 20;}
         else if (state.actions[i].parameters.indexOf('ce') != -1 && state.actions[i].parameters.indexOf('q2') != -1) {
            cost -= 20; }
         else if (state.actions[i].parameters.indexOf('cf') != -1 && state.actions[i].parameters.indexOf('q2') != -1) {
            cost -= 20; } }} return cost;}
```

OUTPUT:

```
Depth: 34, Current cost: 54, 373 child states.
- Solution found in 35 steps!
1. take k1 l1 cf ce q1
2. load k1 l1 cf r1
3. move r1 l1 l2
4. unload k2 l2 cf r1
5. put k2 l2 cf pallet q2
6. take k1 l1 ce cd q1
7. move r1 l2 l1
8. load k1 l1 ce r1
9. move r1 l1 l2
10. unload k2 l2 ce r1
11. put k2 l2 ce cf q2
12. move r1 l2 l1
13. take k1 l1 cd pallet q1
14. load k1 l1 cd r1
```

```
PS C:\Users\fall2023\Desktop\strips> node dockworkerrobot.js
Using A*.

Depth: 0, Current cost: 120, 3 child states.
Depth: 1, Current cost: 121, 4 child states.
Depth: 1, Current cost: 121, 6 child states.
Depth: 1, Current cost: 121, 8 child states.
Depth: 2, Current cost: 122, 8 child states.
Depth: 2, Current cost: 122, 8 child states.
Depth: 2, Current cost: 122, 8 child states.
Depth: 2, Current cost: 122, 10 child states.
Depth: 2, Current cost: 122, 11 child states.
Depth: 2, Current cost: 122, 11 child states.
Depth: 2, Current cost: 122, 12 child states.
Depth: 2, Current cost: 122, 14 child states.
```

TASK NO 2: You need to develop a user-friendly graphical interface for an Optical Character.

```
;; Specification in PDDL1 of the DWR domain
(define (domain dock-worker-robot-pos)
 (:requirements :strips :typing )
 (:types
  location      ; there are several connected locations in the harbor
  pile          ; is attached to a location
                ; it holds a pallet and a stack of containers
  robot         ; holds at most 1 container, only 1 robot per location
  crane         ; belongs to a location to pickup containers
  container)
;; there are 5 operators in this domain:
;; moves a robot between two adjacent locations
 (:action move
    :parameters (?r - robot ?from - location ?to - location)
    :precondition (and (adjacent ?from ?to)
                (at ?r ?from) (free ?to))
    :effect (and (at ?r ?to) (free ?from)
                                (not (free ?to)) (not (at ?r ?from))
))
;; loads an empty robot with a container held by a nearby crane
 (:action load
    :parameters (?k - crane ?l - location ?c - container ?r - robot)
    :precondition (and (at ?r ?l) (belong ?k ?l)
                (holding ?k ?c) (unloaded ?r))
    :effect (and  (loaded ?r ?c) (not (unloaded ?r))
                (empty ?k) (not (holding ?k ?c))))
;; unloads a robot holding a container with a nearby crane
 (:action unload
    :parameters (?k - crane ?l - location ?c - container ?r - robot)
    :precondition (and (belong ?k ?l) (at ?r ?l)
                (loaded ?r ?c) (empty ?k))
    :effect (and (unloaded ?r) (holding ?k ?c)
            (not (loaded ?r ?c))(not (empty ?k))))
;; takes a container from a pile with a crane
 (:action take
    :parameters (?k - crane ?l - location ?c - container ?else -
container ?p - pile)
    :precondition (and (belong ?k ?l)(attached ?p ?l)
                (empty ?k) (in ?c ?p)
                (top ?c ?p) (on ?c ?else))
    :effect (and (holding ?k ?c) (top ?else ?p)
            (not (in ?c ?p)) (not (top ?c ?p))
            (not (on ?c ?else)) (not (empty ?k))))
;; puts a container held by a crane on a nearby pile
```

```
 (:action put
    :parameters (?k - crane ?l - location ?c - container ?else - container ?p - pile)
    :precondition (and (belong ?k ?l) (attached ?p ?l) (holding ?k ?c) (top ?else ?p))
    :effect (and (in ?c ?p) (top ?c ?p) (on ?c ?else) (not (top ?else ?p)) (not (holding ?k ?c)) (empty ?k))))
```

Problem.txt:

```
;; a simple DWR problem with 1 robot and 2 locations
;; A complete planning graph for this problem (strips.fast: false, isSkipNegativeLiterals: true) should display:
;; P0: 29, A1: 3, P1: 35, A2: 16
;; P1: 35, A2: 16, P2: 51, A3: 62
;; P2: 51, A3: 62, P3: 77, A4: 144
;; P3: 77, A4: 144, P4: 109, A5: 244
;; P4: 109, A5: 244, P5: 123, A6: 334
;; P5: 123, A6: 334, P6: 127, A7: 362
;; P6: 127, A7: 362, P7: 127, A8: 362
(define (problem dwrpb1)
 (:domain dock-worker-robot-pos)
 (:objects
  r1 - robot
  l1 l2 - location
  k1 k2 - crane
  p1 q1 p2 q2 - pile
  ca cb cc cd ce cf pallet - container)
 (:init
  (adjacent l1 l2)
  (adjacent l2 l1)
  (attached p1 l1)
  (attached q1 l1)
  (attached p2 l2)
  (attached q2 l2)
  (belong k1 l1)
  (belong k2 l2)
  (in ca p1)
  (in cb p1)
  (in cc p1)
  (in cd q1)
  (in ce q1)
  (in cf q1)
  (on ca pallet)
  (on cb ca)
  (on cc cb)
  (on cd pallet)
  (on ce cd)
  (on cf ce)
  (top cc p1)
  (top cf q1)
  (top pallet p2)
  (top pallet q2)
  (at r1 l1)
  (unloaded r1)
  (free l2)
  (empty k1)
  (empty k2))
 ;; the task is to move all containers to locations l2
 ;; ca and cc in pile p2, the rest in q2
 (:goal
  (and (in ca p2)
       (in cb q2)
       (in cc p2)
       (in cd q2)
       (in ce q2)
       (in cf q2))))
```

```
PS C:\Users\fall2023\Desktop\strips> node dockworkerrobot.js
Using A*.

Depth: 0, Current cost: 120, 3 child states.
Depth: 1, Current cost: 121, 4 child states.
Depth: 1, Current cost: 121, 6 child states.
Depth: 1, Current cost: 121, 8 child states.
Depth: 2, Current cost: 122, 8 child states.
Depth: 2, Current cost: 122, 8 child states.
Depth: 2, Current cost: 122, 8 child states.
Depth: 2, Current cost: 122, 10 child states.
Depth: 2, Current cost: 122, 11 child states.
Depth: 2, Current cost: 122, 11 child states.
Depth: 2, Current cost: 122, 12 child states.
Depth: 2, Current cost: 122, 14 child states.
Depth: 3, Current cost: 123, 14 child states.
Depth: 3, Current cost: 123, 16 child states.
Depth: 3, Current cost: 123, 17 child states.
Depth: 3, Current cost: 123, 19 child states.
```

```
Depth: 34, Current cost: 54, 373 child states.
- Solution found in 35 steps!
1. take k1 l1 cf ce q1
2. load k1 l1 cf r1
3. move r1 l1 l2
4. unload k2 l2 cf r1
5. put k2 l2 cf pallet q2
6. take k1 l1 ce cd q1
7. move r1 l2 l1
8. load k1 l1 ce r1
9. move r1 l1 l2
10. unload k2 l2 ce r1
11. put k2 l2 ce cf q2
12. move r1 l2 l1
13. take k1 l1 cd pallet q1
14. load k1 l1 cd r1
15. move r1 l1 l2
16. unload k2 l2 cd r1
17. put k2 l2 cd ce q2
18. move r1 l2 l1
18. move r1 l2 l1
19. take k1 l1 cc cb p1
20. load k1 l1 cc r1
21. move r1 l1 l2
22. unload k2 l2 cc r1
23. put k2 l2 cc pallet p2
24. move r1 l2 l1
25. take k1 l1 cb ca p1
26. load k1 l1 cb r1
27. move r1 l1 l2
28. unload k2 l2 cb r1
29. put k2 l2 cb cd q2
30. move r1 l2 l1
31. take k1 l1 ca pallet p1
32. load k1 l1 ca r1
33. move r1 l1 l2
34. unload k2 l2 ca r1
35. put k2 l2 ca cc p2
```