Τεχνητή Νοημοσύνη ΙΙ

Εαρινό Εξάμηνο 2011-2012

Εργασία ΙΙ: Σχεδιασμός

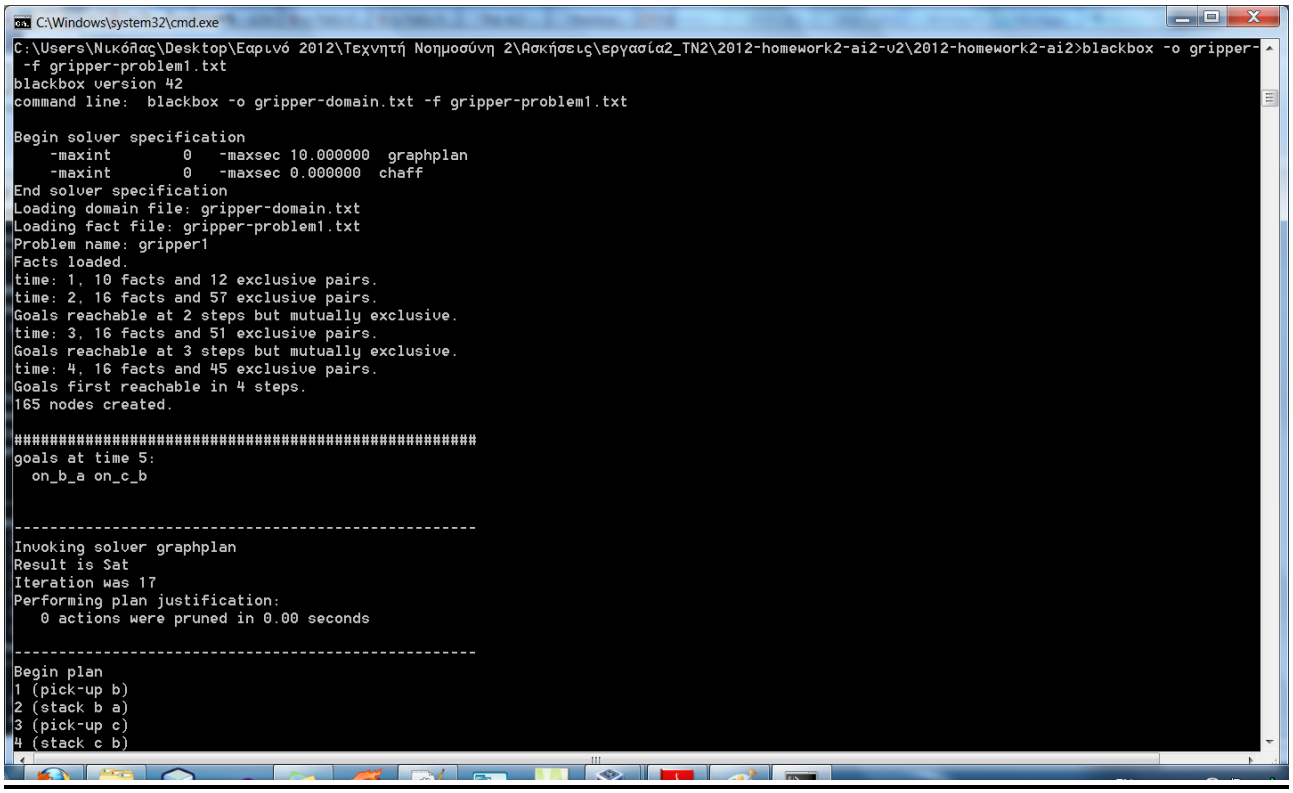
Ημερομηνία Παράδοσης: 26 Μαρτίου 2012

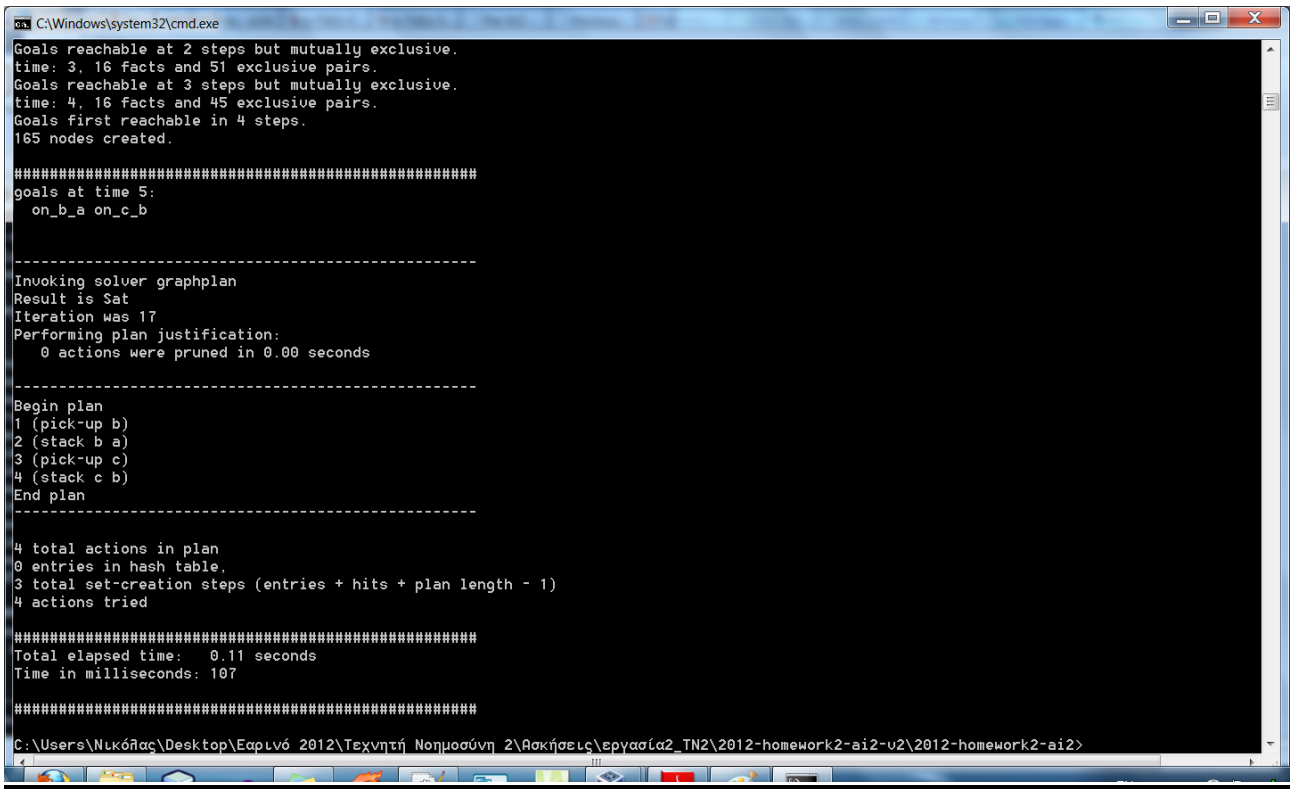
Ον/μο φοιτητή: *Μπεγέτης Νικόλαος*

Α.Μ.: *1115200700281*

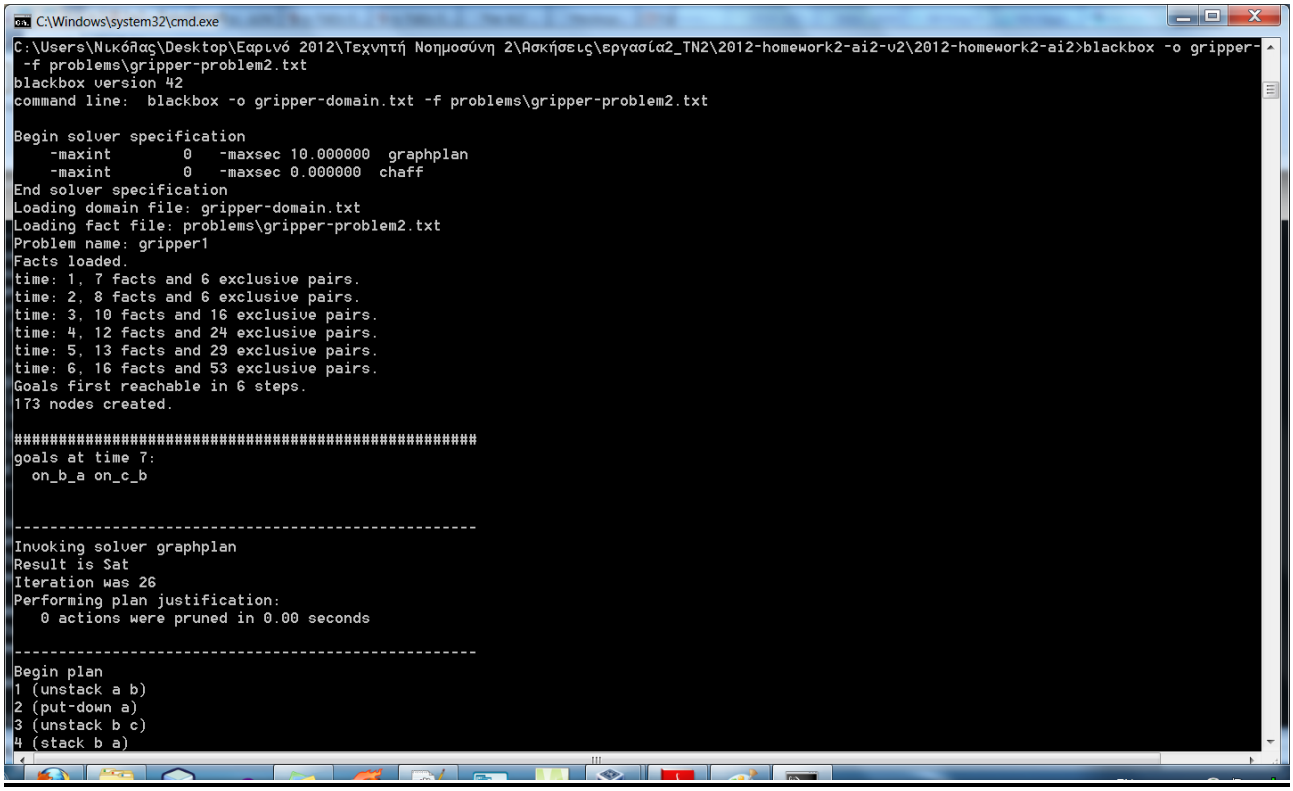
Άσκηση 1(i)

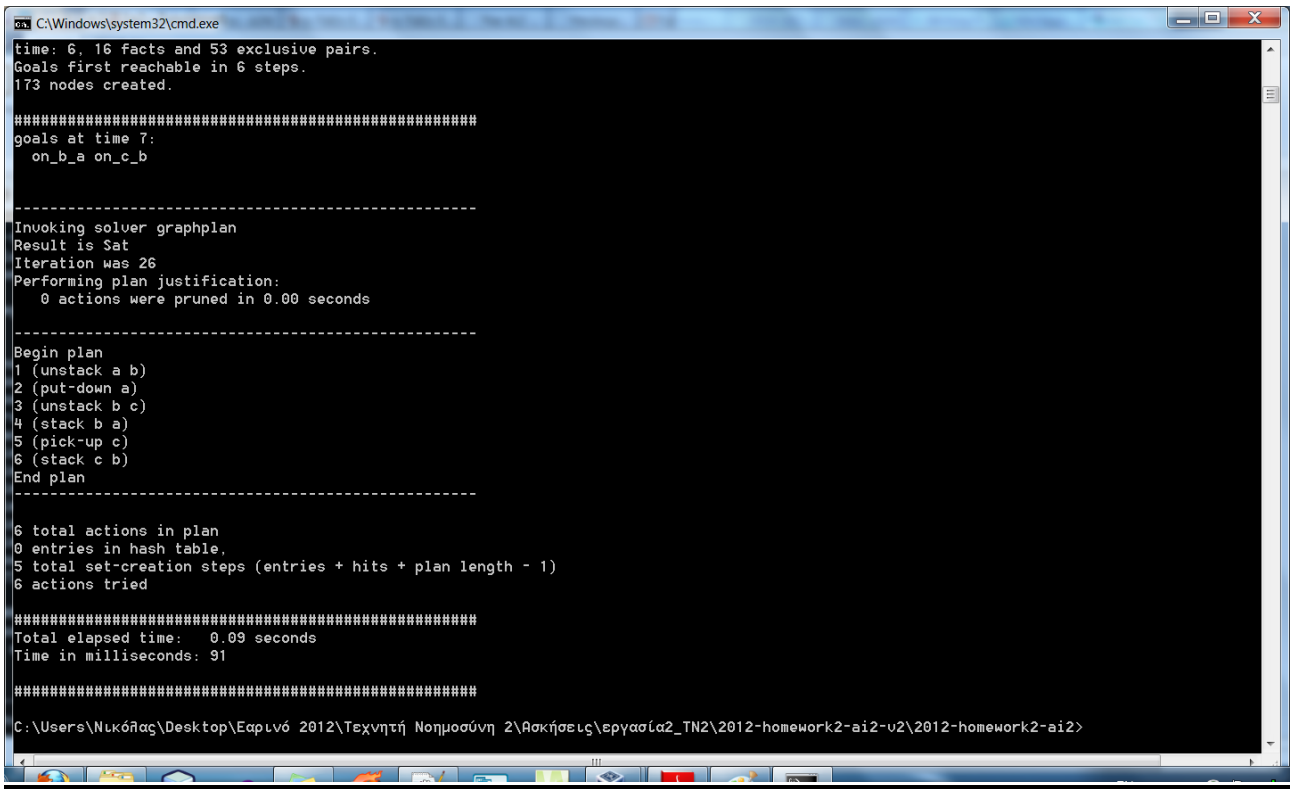
Το πλάνο εκτέλεσης για το πρόβλημα σχεδιασμού (gripper-domain.txt, gripper-problem1.txt):





Το πλάνο εκτέλεσης για το πρόβλημα σχεδιασμού (gripper-domain.txt, gripper-problem2.txt):





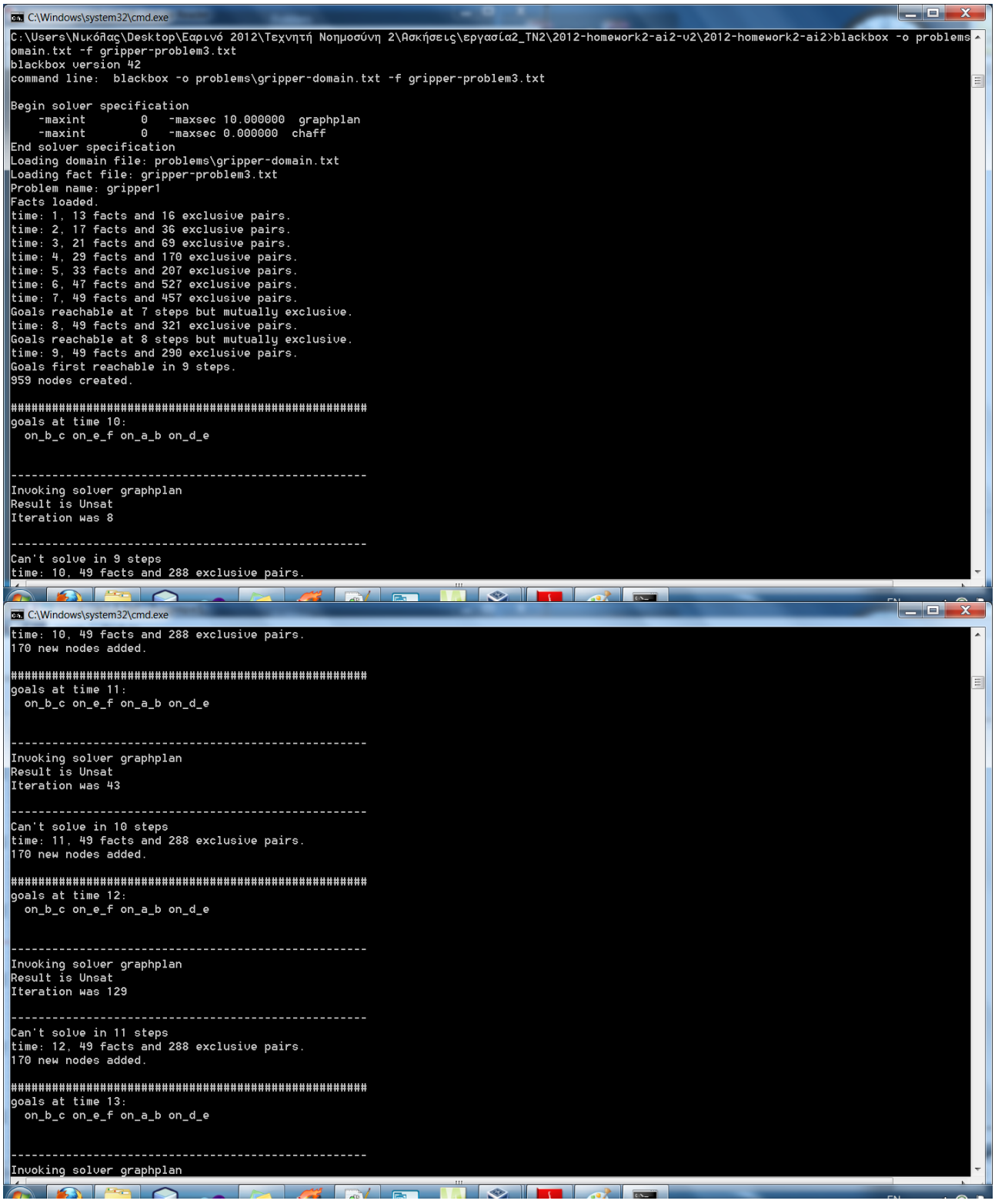
Άσκηση 1(ii)

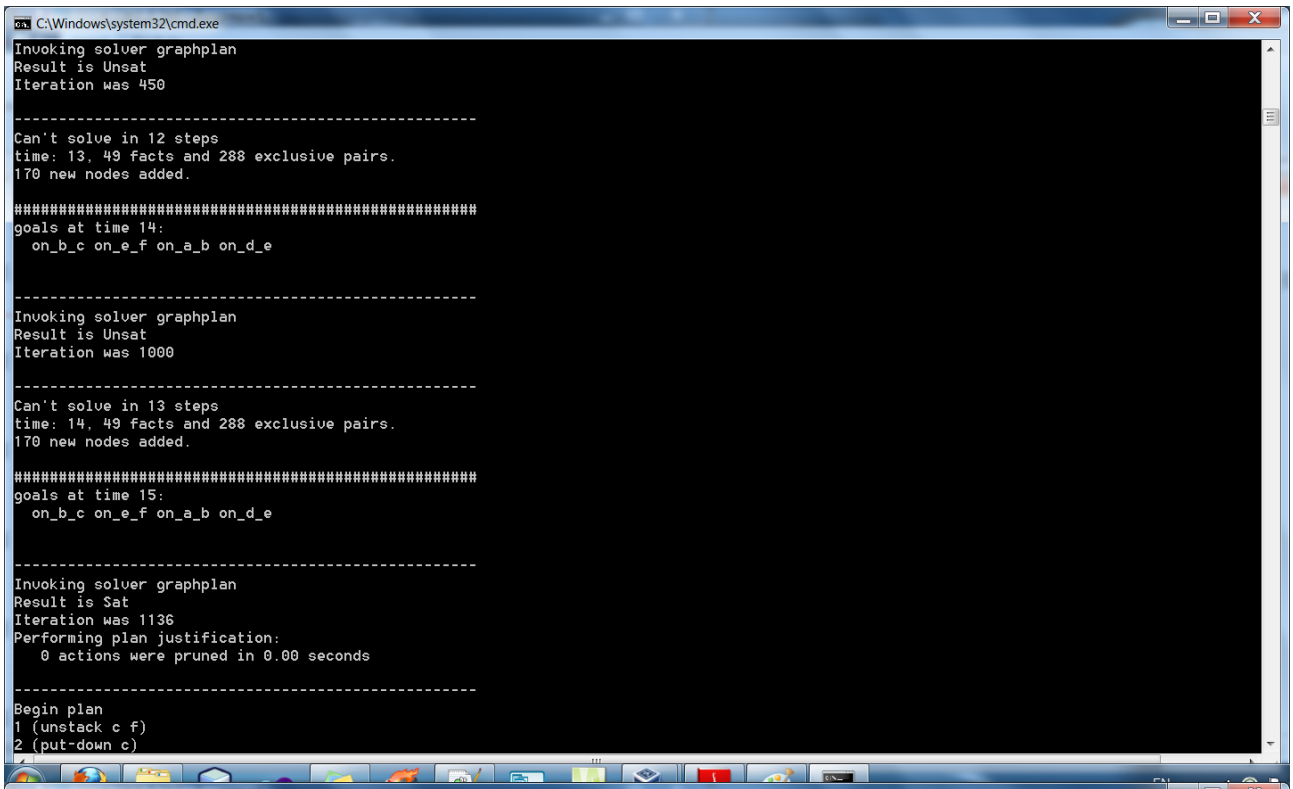
Το αρχείο gripper-problem3.txt το οποίο περιγράφει ένα πρόβλημα σχεδιασμού που εμπλέκει 6 κύβους με την περιγραφή που δίνεται στην εικόνα της εκφώνησης είναι:

```
(define (problem gripper1)
  (:domain gripper)

(:objects a b c d e f )

(:init
(on-table a)
(on-table b)
(on d e)
(on c f)
(on e a)
(on f b)
(clear c)
(clear d)
(grip-empty))
```

```
(:goal (and (on b c) (on e f) (on a b) (on d e)))
)
```

Το πλάνο εκτέλεσης για το πρόβλημα σχεδιασμού (gripper-domain.txt, gripper-problem3.txt) είναι:




C:\Windows\system32\cmd.exe
Invoking solver graphplan
Result is Unsat
Iteration was 450

Can't solve in 12 steps
time: 13, 49 facts and 288 exclusive pairs.
170 new nodes added.
###
goals at time 14:
on_b_c on_e_f on_a_b on_d_e

Invoking solver graphplan
Result is Unsat
Iteration was 1000

Can't solve in 13 steps
time: 14, 49 facts and 288 exclusive pairs.
170 new nodes added.
###
goals at time 15:
on_b_c on_e_f on_a_b on_d_e

Invoking solver graphplan
Result is Sat
Iteration was 1136
Performing plan justification:
0 actions were pruned in 0.00 seconds

Begin plan
1 (unstack c f)
2 (put-down c)


C:\Windows\system32\cmd.exe

Invoking solver graphplan
Result is Sat
Iteration was 1136
Performing plan justification:
0 actions were pruned in 0.00 seconds

Begin plan
1 (unstack c f)
2 (put-down c)
3 (unstack f b)
4 (put-down f)
5 (unstack d e)
6 (put-down d)
7 (unstack e a)
8 (stack e f)
9 (pick-up d)
10 (stack d e)
11 (pick-up b)
12 (stack b c)
13 (pick-up a)
14 (stack a b)
End plan
---
14 total actions in plan
216 entries in hash table, 230 hash hits, avg set size 7
459 total set-creation steps (entries + hits + plan length - 1)
383 actions tried
###
Total elapsed time: 0.30 seconds
Time in milliseconds: 298
###
C:\Users\Νικόλας\Desktop\Εαρινό 2012\Τεχνητή Νοημοσύνη 2\Ασκήσεις\εργασία2_ΤΝ2\2012-homework2-ai2-v2\2012-homework2-ai2>

Άσκηση 2(i)

Όπως αναγράφεται και στην εκφώνηση το κατηγόρημα *dfs(OpenStates, VisitedStates, Solution)* που υπάρχει προς το παρόν στο αρχείο *planner.pl* απλώς παίρνει το πρώτο ζεύγος της λίστας *OpenStates* και ελέγχει αν η κατάσταση που περιγράφεται από το δεύτερο στοιχείο του ζεύγους ικανοποιεί το στόχο. Όπως εύκολα μπορούμε να παρατηρήσουμε για το πρόβλημα *gripper-problem0.txt* η κατάσταση στόχου επιτυγχάνεται ήδη από την αρχική κατάσταση -τοποθέτηση- των κύβων. Συνεπώς στο αρχείο *planner.pl* ικανοποιείται το κατηγόρημα *satisfies_goal(CurrentState), -γραμμή 108-* επιστρέφοντας *true* με αποτέλεσμα η εκτέλεση του κατηγορήματος *dfs(OpenStates, VisitedStates, Solution)* να προχωράει και να μη σταματάει στη *γραμμή 108* και έτσι στο τέλος να εκτυπώνεται η λύση που είναι *true*.

Από την άλλη πλευρά, για το πρόβλημα *gripper-problem1.txt* η κατάσταση στόχου δεν επιτυγχάνεται από την αρχική κατάσταση, δηλαδή δεν ικανοποιείται το κατηγόρημα *satisfies_goal(CurrentState),* το οποίο επιστρέφει *false,* με αποτέλεσμα η εκτέλεση του κατηγορήματος *dfs(OpenStates, VisitedStates, Solution)* να σταματάει στη *γραμμή 108.* Τότε θα εκτελεστεί το επόμενο στη σειρά *dfs(OpenStates, VisitedStates, Solution)* το οποίο θα ελέγξει εάν η *CurrentState* ανήκει στη λίστα με τις *VisitedStates* μέσω του κατηγορήματος *member(CurrentState, VisitedStates),* . Τότε το κατηγόρημα αυτό θα επιστρέψει *false* και έτσι θα εκτελεστεί το επόμενο στη σειρά *dfs(OpenStates, VisitedStates, Solution).* Αυτό το κατηγόρημα θα εκτελεστεί σωστά τυπώνοντας μία τελεία *'.'*, και θα καλέσει ξανά αναδρομικά τον εαυτό του με *dfs(RestOfOpenStates, [CurrentState | VisitedStates], Solution),* αλλάζοντας δηλαδή το δεύτερο όρισμα του κατηγορήματος. Ξεκινώντας πάλι από την αρχή στις εκτυπώσεις παρατηρούμε ότι πλέον η αρχική κατάσταση ανήκει στις *VisitedStates* και ότι δεν υπάρχουν άλλες διαθέσιμες *OpenStates.* Έτσι στη *γραμμή 100* στην εντολή *[FirstPair |_] = OpenStates,* επιστρέφεται false αφού δεν υπάρχουν άλλα *OpenStates.* Έπειτα εκτελείται για δεύτερη φορά το επόμενο *dfs(OpenStates, VisitedStates, Solution)* το οποίο επειδή δεν υπάρχουν άλλα *OpenStates* θα επιστρέψει και αυτό *false*. Τέλος, φτάνοντας στο τελευταίο στη σειρά κατηγόρημα *dfs(OpenStates, VisitedStates, Solution)* θα επιστρέψει και αυτό *false* πάλι για τον ίδιο λόγο με αποτέλεσμα η αναζήτηση *search(Solution),* να ολοκληρωθεί ανεπιτυχώς και να τερματίσει το ερώτημα με *false*.

Συνοψίζοντας, ο λόγος για τον οποίο η ημιτελής υλοποίηση του *planner* αδυνατεί να βρει λύση στη δεύτερη περίπτωση είναι ότι δεν γίνεται κάποια αναζήτηση για εύρεση νέων *OpenStates* που μπορούν να 'παραχθούν' έπειτα από την εφαρμογή κάποιας νέας ενέργειας. Συνεπώς, η λίστα των *OpenStates* δεν επεκτείνεται, και στην περίπτωση που δεν περιλαμβάνει ήδη την κατάσταση στόχου τότε σίγουρα θα οδηγήσει σε αποτυχία. Αυτό γιατί κάποια στιγμή οι καταστάσεις που περιλαμβάνονται στη λίστα των *OpenStates* θα έχουν όλες εξερευνηθεί τουλάχιστον μία φορά χωρίς να βρεθεί σε αυτές η κατάσταση στόχου.

Άσκηση 2(ii)

Το κατηγόρημα *dfs/3* τροποποιήθηκε κατάλληλα ώστε να υλοποιεί μία αναδρομική εκδοχή της αναζήτησης που πραγματοποιείται με το loop της *DepthFirstPlanner* που μας δόθηκε στην πρώτη εργασία του μαθήματος.

Ο κώδικας που υλοποιεί αυτή την αναδρομή είναι ο εξής:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% dfs(+OpenStates, +VisitedStates, -Solution)

%%    dfs/3 should implement a recursive depth-first search for solving

%%    the parsed planning problem using progression.
```

```
%%    At the moment it only checks if the first pair in the list of open
%%    states, that is OpenStates, satisfies the goal.
%%
%% YOU NEED TO ADD CODE HERE!
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Check if the first state in the list of open states satisfies the goal.
dfs(OpenStates, VisitedStates, Solution):-
        % Print useful debug information
        print('*Debug Information* A call to dfs/3 has just been performed\n'),
        print('OpenStates:' ),print(OpenStates), nl,
        print('VisitedStates:' ),print(VisitedStates), nl,
        % Get the first pair from the list OpenStates
        [FirstPair |_] = OpenStates,
        % Get the first element of the pair as the current list of actions
        % and the second element as the current state
        [CurrentListOfActions,CurrentState] = FirstPair,
        % Print some more debug information
        print('CurrentListOfActions:' ),print(CurrentListOfActions), nl,
        print('CurrentState:' ),print(CurrentState), nl,
        % Check if the current state satisfies the goal
        satisfies_goal(CurrentState),
        % If we get up to here then the goal has been satisfied,
        % therefore we are done
        !,
        Solution = CurrentListOfActions.

% Then check if the first state in the list of open states is already visited.
dfs(OpenStates, VisitedStates, Solution):-
        % Get the first pair from the list OpenStates
        [FirstPair | RestOfOpenStates] = OpenStates,
        % Get the first element of the pair as the current list of actions
        % and the second element as the current state
```

```
        [_CurrentListOfActions,CurrentState] = FirstPair,
                % Debug information -- uncomment to print useful information
                % print('*Debug* check state if it is already visited:\n'),
                % print(CurrentState), nl,
        % Check if the state description is already in the list of visited states
        member(CurrentState, VisitedStates),
        nl, nl,nl,print('CurrentState EXISTS in VisitedStates...Try next Action' ), nl,nl,nl,
        % If we get up to here, then it's a state we've seen before.
        % No need to expand this node using the next rule, so no more bactracking!
        !,
                % Debug information -- uncomment to print useful information
                % print('*Debug* state is already visited:\n'),
                % print(CurrentState), nl,
        % Continue search using the remaining list of open states.
        dfs(RestOfOpenStates, VisitedStates, Solution).


% Finally, expand the first state in the list of open states and recurse.
dfs(OpenStates, VisitedStates, Solution):-
        % Get the first pair from the list OpenStates
        [FirstPair | RestOfOpenStates] = OpenStates,
%        nl,print('FirstPair= ' ),print(FirstPair),
%        nl,print('RestOfOpenStates= ' ),print(RestOfOpenStates),
        % Get the first element of the pair as the current list of actions
        % and the second element as the current state
        [CurrentListOfActions,CurrentState] = FirstPair,
                % Debug information -- uncomment to print useful information
                % print('*Debug* expand state:\n'), print(CurrentState), nl,
                % Debug information -- uncomment to print useful information
                % print('*Debug* open states:\n'), print(OpenStates), nl,
        % Update the global variables that hold statistics about the search;
```

```
        % this should be called whenever a pair from OpenStates is processed

        % to produce the successor states that will be inserted in OpenStates.

        stat_node,

        % Print a dot for every node expanded.

        nl,print('.........................'), nl, flush_output,

        % Expand the current state by inserting the successor states in the

        % front of the list of open states, and recurse

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%

        % *** ADD YOUR CODE HERE *** %

        % *** REPLACE NEXT LINE  *** %

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%

        %     Generates all the possible successor states from CurrentState.
Action is

        %    an applicable ground action to State, and NextState is the

        %    successor state when the effects of Action are applied to State.

        findall([[NewAction      |      CurrentListOfActions],      NextState],
progress(CurrentState, NewAction, NextState), NewOpenStates),

        nl,nl,print('ListOfNewOpenStates = ' ),print(NewOpenStates), nl,

        nl,nl,print('ListOfRestOpenStates = ' ),print(RestOfOpenStates), nl,

% Append new openstates with old openstates

      append(NewOpenStates, RestOfOpenStates, OpenStates2),

        nl,nl,print('OpenStates = ' ),print(OpenStates), nl,

        dfs(OpenStates2, [CurrentState | VisitedStates], Solution).
```

Τα αποτελέσματα που παράγονται από την εκτέλεση του *planner* για το πρόβλημα (gripper-domain.txt, gripper-problem1.txt) είναι:

(Παρατίθενται μόνο τα αποτελέσματα της τελευταίας αναδρομικής κλήσης του κατηγορήματος... Αν ζητηθεί μπορώ να συμπεριλάβω και τα υπόλοιπα, αλλιώς μπορείτε να τα δείτε και εάν εκτελέσετε τον planner)

Debug Information A call to dfs/3 has just been performed

OpenStates:[[[stack(c,b),pick-up(c),stack(b,a),pick-up(b)],[grip-empty,clear(c),on-table(a),on(b,a),on(c,b)]],[[unstack(b,a),stack(b,a),pick-up(b)],[clear(a),clear(c),holding(b),on-table(a),on-table(c)]],[[stack(b,c),pick-up(b)],[grip-empty,clear(a),clear(b),on-table(a),on-table(c),on(b,c)]],[[pick-up(c)],[clear(a),clear(b),holding(c),on-table(a),on-table(b)]]]

VisitedStates:[[clear(b),holding(c),on-table(a),on(b,a)],[grip-empty,clear(b),clear(c),on-table(a),on-table(c),on(b,a)],[clear(a),clear(c),holding(b),on-table(a),on-table(c)],[grip-empty,clear(b),on-table(c),on(a,c),on(b,a)],[clear(a),holding(b),on-table(c),on(a,c)],[grip-empty,clear(a),clear(b),on-table(b),on-table(c),on(a,c)],[grip-empty,clear(c),on-table(b),on(a,b),on(c,a)],[clear(a),holding(c),on-table(b),on(a,b)],[grip-empty,clear(a),clear(c),on-table(b),on-

table(c),on(a,b)],[clear(b),clear(c),holding(a),on-table(b),on-table(c)],[grip-empty,clear(a),clear(b),clear(c),on-table(a),on-table(b),on-table(c)]]

CurrentListOfActions:[stack(c,b),pick-up(c),stack(b,a),pick-up(b)]

CurrentState:[grip-empty,clear(c),on-table(a),on(b,a),on(c,b)]

Solution found!

Domain name: gripper

Problem name: gripper1

Time: 0.078 sec

Search length: 11 nodes

Solution length: 4 actions

Solution: (stack c b) (pick-up c) (stack b a) (pick-up b)

true.

Αντίστοιχα, τα αποτελέσματα που παράγονται από την εκτέλεση του *planner* για το πρόβλημα (gripper-domain.txt, gripper-problem2.txt) είναι:

(Παρατίθενται μόνο τα αποτελέσματα της τελευταίας αναδρομικής κλήσης του κατηγορήματος... Αν ζητηθεί μπορώ να συμπεριλάβω και τα υπόλοιπα, αλλιώς μπορείτε να τα δείτε και εάν εκτελέσετε τον planner)

Debug Information A call to dfs/3 has just been performed

OpenStates:[[[stack(c,b),pick-up(c),stack(b,a),unstack(b,c),put-down(a),unstack(a,b)],[grip-empty,clear(c),on-table(a),on(b,a),on(c,b)]],[[unstack(b,a),stack(b,a),unstack(b,c),put-down(a),unstack(a,b)],[clear(a),clear(c),holding(b),on-table(a),on-table(c)]],[[stack(b,c),unstack(b,c),put-down(a),unstack(a,b)],[grip-empty,clear(a),clear(b),on-table(a),on-table(c),on(b,c)]],[[stack(a,b),unstack(a,b)],[grip-empty,clear(a),on-table(c),on(a,b),on(b,c)]]]

VisitedStates:[[clear(b),holding(c),on-table(a),on(b,a)],[grip-empty,clear(b),clear(c),on-table(a),on-table(c),on(b,a)],[grip-empty,clear(a),on-table(b),on(a,c),on(c,b)],[clear(c),holding(a),on-table(b),on(c,b)],[grip-empty,clear(a),clear(c),on-table(a),on-table(b),on(c,b)],[grip-empty,clear(b),on-table(a),on(b,c),on(c,a)],[clear(c),holding(b),on-table(a),on(c,a)],[grip-empty,clear(b),clear(c),on-table(a),on-table(b),on(c,a)],[clear(a),clear(b),holding(c),on-table(a),on-table(b)],[grip-empty,clear(b),on-table(c),on(a,c),on(b,a)],[clear(a),holding(b),on-table(c),on(a,c)],[grip-empty,clear(a),clear(b),on-table(b),on-table(c),on(a,c)],[grip-empty,clear(c),on-table(b),on(a,b),on(c,a)],[clear(a),holding(c),on-table(b),on(a,b)],[grip-empty,clear(a),clear(c),on-table(b),on-table(c),on(a,b)],[clear(b),clear(c),holding(a),on-table(b),on-table(c)],[grip-empty,clear(a),clear(b),clear(c),on-table(a),on-table(b),on-table(c)],[clear(a),clear(c),holding(b),on-table(a),on-table(c)],[grip-empty,clear(a),clear(b),on-table(a),on-table(c),on(b,c)],[clear(b),holding(a),on-table(c),on(b,c)],[grip-empty,clear(a),on-table(c),on(a,b),on(b,c)]]

CurrentListOfActions:[stack(c,b),pick-up(c),stack(b,a),unstack(b,c),put-down(a),unstack(a,b)]

CurrentState:[grip-empty,clear(c),on-table(a),on(b,a),on(c,b)]

Solution found!

Domain name: gripper

Problem name: gripper1

Time: 0.23 4 sec

Search length: 21 nodes

Solution length: 6 actions

Solution: (stack c b) (pick-up c) (stack b a) (unstack b c) (put-down a) (unstack a b)

true.

***Τέλος, αξίζει να σημειωθεί ότι οι κινήσεις μέσω των οποίων φτάνουμε στο στόχο εμφανίζονται από την τελευταία προς την αρχική. Αυτό δεν επηρεάζει το αποτέλεσμα μιας και η λεξικογραφική ταξινόμηση των καταστάσεων γίνεται με βάση μόνο το State ως string όπως χαρακτηριστικά αναφέρεται και στη 2η άσκηση της 1ης εργασίας. Εύκολα θα μπορούσαμε να επεκτείναμε την λίστα ListOfActions βάζοντας την καινούργια ενέργεια (NewAction) κάθε φορά στο τέλος της CurrentListOfActions και όχι στην αρχή της, παίρνοντας έτσι την αντίστροφη σειρά στο Solution.*

Άσκηση 3(i)

Τα αποτελέσματα που παράγονται από την εκτέλεση του *planner* για το πρόβλημα (sokoban-domain.txt, sokoban-problem1.txt) είναι:

(Παρατίθενται μόνο τα αποτελέσματα της τελευταίας αναδρομικής κλήσης του κατηγορήματος... Αν ζητηθεί μπορώ να συμπεριλάβω και τα υπόλοιπα, αλλιώς μπορείτε να τα δείτε και εάν εκτελέσετε τον planner)

Debug Information A call to dfs/3 has just been performed

OpenStates:[[[push(c3-1,c3-2,c3-3,up,box1),move(c2-1,c3-1,right),move(c1-1,c2-1,right)],[empty(c1-1),empty(c1-2),empty(c1-3),empty(c2-1),empty(c3-1),robot-at(c3-2),object-at(box1,c3-3),adjacent(c1-1,c1-2,up),adjacent(c1-1,c2-1,right),adjacent(c1-2,c1-1,down),adjacent(c1-2,c1-3,up),adjacent(c1-2,c2-2,right),adjacent(c1-3,c1-2,down),adjacent(c1-3,c2-3,right),adjacent(c2-1,c1-1,left),adjacent(c2-1,c2-2,up),adjacent(c2-1,c3-1,right),adjacent(c2-2,c1-2,left),adjacent(c2-2,c2-1,down),adjacent(c2-2,c2-3,up),adjacent(c2-2,c3-2,right),adjacent(c2-3,c1-3,left),adjacent(c2-3,c2-2,up),adjacent(c2-3,c3-3,right),adjacent(c3-1,c2-1,left),adjacent(c3-1,c3-2,up),adjacent(c3-2,c2-2,left),adjacent(c3-2,c3-1,down),adjacent(c3-2,c3-3,up),adjacent(c3-3,c2-3,left),adjacent(c3-3,c3-2,down)]]]

VisitedStates:[[empty(c1-1),empty(c1-2),empty(c1-3),empty(c2-1),empty(c3-3),robot-at(c3-1),object-at(box1,c3-2),adjacent(c1-1,c1-2,up),adjacent(c1-1,c2-1,right),adjacent(c1-2,c1-1,down),adjacent(c1-2,c1-3,up),adjacent(c1-2,c2-2,right),adjacent(c1-3,c1-2,down),adjacent(c1-3,c2-3,right),adjacent(c2-1,c1-1,left),adjacent(c2-1,c2-2,up),adjacent(c2-1,c3-1,right),adjacent(c2-2,c1-2,left),adjacent(c2-2,c2-1,down),adjacent(c2-2,c2-3,up),adjacent(c2-2,c3-2,right),adjacent(c2-3,c1-3,left),adjacent(c2-3,c2-2,up),adjacent(c2-3,c3-3,right),adjacent(c3-1,c2-1,left),adjacent(c3-1,c3-2,up),adjacent(c3-2,c2-2,left),adjacent(c3-2,c3-1,down),adjacent(c3-2,c3-3,up),adjacent(c3-3,c2-3,left),adjacent(c3-3,c3-2,down)],[empty(c1-1),empty(c1-2),empty(c1-3),empty(c3-1),empty(c3-3),robot-at(c2-1),object-at(box1,c3-2),adjacent(c1-1,c1-2,up),adjacent(c1-1,c2-1,right),adjacent(c1-2,c1-1,down),adjacent(c1-2,c1-3,up),adjacent(c1-2,c2-2,right),adjacent(c1-3,c1-2,down),adjacent(c1-3,c2-3,right),adjacent(c2-1,c1-1,left),adjacent(c2-1,c2-2,up),adjacent(c2-1,c3-1,right),adjacent(c2-2,c1-2,left),adjacent(c2-2,c2-

1,down),adjacent(c2-2,c2-3,up),adjacent(c2-2,c3-2,right),adjacent(c2-3,c1-3,left),adjacent(c2-3,c2-2,up),adjacent(c2-3,c3-3,right),adjacent(c3-1,c2-1,left),adjacent(c3-1,c3-2,up),adjacent(c3-2,c2-2,left),adjacent(c3-2,c3-1,down),adjacent(c3-2,c3-3,up),adjacent(c3-3,c2-3,left),adjacent(c3-3,c3-2,down)],[empty(c1-1),empty(c1-2),empty(c2-1),empty(c3-1),empty(c3-3),robot-at(c1-3),object-at(box1,c3-2),adjacent(c1-1,c1-2,up),adjacent(c1-1,c2-1,right),adjacent(c1-2,c1-1,down),adjacent(c1-2,c1-3,up),adjacent(c1-2,c2-2,right),adjacent(c1-3,c1-2,down),adjacent(c1-3,c2-3,right),adjacent(c2-1,c1-1,left),adjacent(c2-1,c2-2,up),adjacent(c2-1,c3-1,right),adjacent(c2-2,c1-2,left),adjacent(c2-2,c2-1,down),adjacent(c2-2,c2-3,up),adjacent(c2-2,c3-2,right),adjacent(c2-3,c1-3,left),adjacent(c2-3,c2-2,up),adjacent(c2-3,c3-3,right),adjacent(c3-1,c2-1,left),adjacent(c3-1,c3-2,up),adjacent(c3-2,c2-2,left),adjacent(c3-2,c3-1,down),adjacent(c3-2,c3-3,up),adjacent(c3-3,c2-3,left),adjacent(c3-3,c3-2,down)],[empty(c1-1),empty(c1-3),empty(c2-1),empty(c3-1),empty(c3-3),robot-at(c1-2),object-at(box1,c3-2),adjacent(c1-1,c1-2,up),adjacent(c1-1,c2-1,right),adjacent(c1-2,c1-1,down),adjacent(c1-2,c1-3,up),adjacent(c1-2,c2-2,right),adjacent(c1-3,c1-2,down),adjacent(c1-3,c2-3,right),adjacent(c2-1,c1-1,left),adjacent(c2-1,c2-2,up),adjacent(c2-1,c3-1,right),adjacent(c2-2,c1-2,left),adjacent(c2-2,c2-1,down),adjacent(c2-2,c2-3,up),adjacent(c2-2,c3-2,right),adjacent(c2-3,c1-3,left),adjacent(c2-3,c2-2,up),adjacent(c2-3,c3-3,right),adjacent(c3-1,c2-1,left),adjacent(c3-1,c3-2,up),adjacent(c3-2,c2-2,left),adjacent(c3-2,c3-1,down),adjacent(c3-2,c3-3,up),adjacent(c3-3,c2-3,left),adjacent(c3-3,c3-2,down)],[empty(c1-2),empty(c1-3),empty(c2-1),empty(c3-1),empty(c3-3),robot-at(c1-1),object-at(box1,c3-2),adjacent(c1-1,c1-2,up),adjacent(c1-1,c2-1,right),adjacent(c1-2,c1-1,down),adjacent(c1-2,c1-3,up),adjacent(c1-2,c2-2,right),adjacent(c1-3,c1-2,down),adjacent(c1-3,c2-3,right),adjacent(c2-1,c1-1,left),adjacent(c2-1,c2-2,up),adjacent(c2-1,c3-1,right),adjacent(c2-2,c1-2,left),adjacent(c2-2,c2-1,down),adjacent(c2-2,c2-3,up),adjacent(c2-2,c3-2,right),adjacent(c2-3,c1-3,left),adjacent(c2-3,c2-2,up),adjacent(c2-3,c3-3,right),adjacent(c3-1,c2-1,left),adjacent(c3-1,c3-2,up),adjacent(c3-2,c2-2,left),adjacent(c3-2,c3-1,down),adjacent(c3-2,c3-3,up),adjacent(c3-3,c2-3,left),adjacent(c3-3,c3-2,down)]]

CurrentListOfActions:[push(c3-1,c3-2,c3-3,up,box1),move(c2-1,c3-1,right),move(c1-1,c2-1,right)]

CurrentState:[empty(c1-1),empty(c1-2),empty(c1-3),empty(c2-1),empty(c3-1),robot-at(c3-2),object-at(box1,c3-3),adjacent(c1-1,c1-2,up),adjacent(c1-1,c2-1,right),adjacent(c1-2,c1-1,down),adjacent(c1-2,c1-3,up),adjacent(c1-2,c2-2,right),adjacent(c1-3,c1-2,down),adjacent(c1-3,c2-3,right),adjacent(c2-1,c1-1,left),adjacent(c2-1,c2-2,up),adjacent(c2-1,c3-1,right),adjacent(c2-2,c1-2,left),adjacent(c2-2,c2-1,down),adjacent(c2-2,c2-3,up),adjacent(c2-2,c3-2,right),adjacent(c2-3,c1-3,left),adjacent(c2-3,c2-2,up),adjacent(c2-3,c3-3,right),adjacent(c3-1,c2-1,left),adjacent(c3-1,c3-2,up),adjacent(c3-2,c2-2,left),adjacent(c3-2,c3-1,down),adjacent(c3-2,c3-3,up),adjacent(c3-3,c2-3,left),adjacent(c3-3,c3-2,down)]

Solution found!

Domain name: sokoban

Problem name: sokoban1

Time: 0.031 sec

Search length: 5 nodes

Solution length: 3 actions

Solution: (push c3-1 c3-2 c3-3 up box1) (move c2-1 c3-1 right) (move c1-1 c2-1 right)

true.

Αντίστοιχα, τα αποτελέσματα που παράγονται από την εκτέλεση του *planner* για το πρόβλημα (sokoban-

domain.txt, sokoban-problem2.txt) είναι:

(Παρατίθενται μόνο τα τελικά αποτελέσματα, γιατί μόνο τα αποτελέσματα από την τελευταία αναδρομική κλήση του κατηγορήματος ξεπερνούν τις 10 σελίδες… Αν ζητηθεί μπορώ να συμπεριλάβω και τα υπόλοιπα, αλλιώς μπορείτε να τα δείτε και εάν εκτελέσετε τον planner)

Debug Information A call to dfs/3 has just been performed

…(Τα αποτελέσματα από την τελευταία εκτέλεση παραλείπονται λόγω πάρα πολύ μεγάλης έκτασης)…

Solution found!

Domain name: sokoban

Problem name: sokoban2

Time: 24.008 sec

Search length: 1274 nodes

Solution length: 107 actions

Solution: (push c7-1 c7-2 c7-3 up box1) (move c6-1 c7-1 right) (move c5-1 c6-1 right) (move c4-1 c5-1 right) (move c4-2 c4-1 down) (move c5-2 c4-2 left) (move c6-2 c5-2 left) (push c5-2 c6-2 c7-2 right box1) (move c4-2 c5-2 right) (move c4-1 c4-2 up) (move c5-1 c4-1 left) (move c6-1 c5-1 left) (move c7-1 c6-1 left) (move c7-2 c7-1 down) (move c7-3 c7-2 down) (move c7-4 c7-3 down) (move c6-4 c7-4 right) (move c5-4 c6-4 right) (move c5-5 c5-4 down) (move c5-6 c5-5 down) (move c5-7 c5-6 down) (move c5-8 c5-7 down) (move c4-8 c5-8 right) (move c4-7 c4-8 up) (move c3-7 c4-7 right) (move c3-8 c3-7 down) (move c2-8 c3-8 right) (move c2-7 c2-8 up) (move c2-6 c2-7 up) (move c2-5 c2-6 up) (move c2-4 c2-5 up) (move c2-3 c2-4 up) (move c3-3 c2-3 left) (move c4-3 c3-3 left) (move c5-3 c4-3 left) (move c6-3 c5-3 left) (push c6-4 c6-3 c6-2 down box1) (move c5-4 c6-4 right) (move c5-5 c5-4 down) (move c5-6 c5-5 down) (move c5-7 c5-6 down) (move c5-8 c5-7 down) (move c4-8 c5-8 right) (move c4-7 c4-8 up) (move c3-7 c4-7 right) (move c3-8 c3-7 down) (move c2-8 c3-8 right) (move c2-7 c2-8 up) (move c2-6 c2-7 up) (move c2-5 c2-6 up) (move c2-4 c2-5 up) (move c2-3 c2-4 up) (move c3-3 c2-3 left) (move c4-3 c3-3 left) (move c5-3 c4-3 left) (push c4-3 c5-3 c6-3 right box1) (move c3-3 c4-3 right) (move c2-3 c3-3 right) (move c2-4 c2-3 down) (move c2-5 c2-4 down) (move c2-6 c2-5 down) (move c2-7 c2-6 down) (move c3-7 c2-7 left) (move c4-7 c3-7 left) (move c5-7 c4-7 left) (move c5-6 c5-7 up) (move c5-5 c5-6 up) (move c5-4 c5-5 up) (push c5-5 c5-4 c5-3 down box1) (push c5-6 c5-5 c5-4 down box1) (push c5-7 c5-6 c5-5 down box1) (push c5-8 c5-7 c5-6 down box1) (move c4-8 c5-8 right) (move c3-8 c4-8 right) (move c2-8 c3-8 right) (move c2-7 c2-8 up) (move c3-7 c2-7 left) (move c4-7 c3-7 left) (push c3-7 c4-7 c5-7 right box1) (push c2-7 c3-7 c4-7 right box1) (move c2-8 c2-7 down) (move c3-8 c2-8 left) (move c4-8 c3-8 left) (move c4-7 c4-8 up) (move c5-7 c4-7 left) (move c5-6 c5-7 up) (move c5-5 c5-6 up) (move c5-4 c5-5 up) (move c5-3 c5-4 up) (move c5-2 c5-3 up) (move c5-1 c5-2 up) (move c4-1 c5-1 right) (move c4-2 c4-1 down) (move c4-3 c4-2 down) (move c3-3 c4-3 right) (move c2-3 c3-3 right) (move c2-4 c2-3 down) (move c2-5 c2-4 down) (move c2-6 c2-5 down) (move c3-6 c2-6 left) (push c3-5 c3-6 c3-7 up box1) (move c2-5 c3-5 right) (move c2-4 c2-5 up) (move c2-3 c2-4 up) (move c3-3 c2-3 left) (move c4-3 c3-3 left) (move c5-3 c4-3 left)

true.

Τέλος, τα αποτελέσματα που παράγονται από την εκτέλεση του *planner* για το πρόβλημα (sokoban-domain.txt, sokoban-problem3.txt) είναι:

(Παρατίθενται μόνο τα τελικά αποτελέσματα, γιατί μόνο τα αποτελέσματα από την τελευταία αναδρομική

κλήση του κατηγορήματος ξεπερνούν τις 10 σελίδες... Αν ζητηθεί μπορώ να συμπεριλάβω και τα υπόλοιπα, αλλιώς μπορείτε να τα δείτε και εάν εκτελέσετε τον planner)

Debug Information A call to dfs/3 has just been performed

…(Τα αποτελέσματα από την τελευταία εκτέλεση παραλείπονται λόγω πάρα πολύ μεγάλης έκτασης)…

Solution found!

Domain name: sokoban

Problem name: sokoban3

Time: 20.951 sec

Search length: 3069 nodes

Solution length: 108 actions

Solution: (push c2-3 c3-3 c4-3 right box1) (push c1-3 c2-3 c3-3 right box1) (move c1-2 c1-3 up) (move c2-2 c1-2 left) (push c2-1 c2-2 c2-3 up box1) (move c1-1 c2-1 right) (move c1-2 c1-1 down) (move c1-3 c1-2 down) (move c2-3 c1-3 left) (move c3-3 c2-3 left) (move c3-2 c3-3 up) (push c4-2 c3-2 c2-2 left box1) (move c4-3 c4-2 down) (push c3-3 c4-3 c5-3 right box2) (push c2-3 c3-3 c4-3 right box2) (push c1-3 c2-3 c3-3 right box2) (move c1-2 c1-3 up) (move c2-2 c1-2 left) (push c2-1 c2-2 c2-3 up box2) (move c1-1 c2-1 right) (move c1-2 c1-1 down) (move c1-3 c1-2 down) (move c2-3 c1-3 left) (move c3-3 c2-3 left) (move c4-3 c3-3 left) (move c4-2 c4-3 up) (push c5-2 c4-2 c3-2 left box1) (move c5-3 c5-2 down) (move c5-4 c5-3 down) (move c4-4 c5-4 right) (move c4-3 c4-4 up) (push c4-4 c4-3 c4-2 down box1) (move c5-4 c4-4 left) (move c5-3 c5-4 up) (push c6-3 c5-3 c4-3 left box1) (move c6-4 c6-3 down) (move c5-4 c6-4 right) (move c4-4 c5-4 right) (move c4-3 c4-4 up) (move c3-3 c4-3 right) (move c3-2 c3-3 up) (push c4-2 c3-2 c2-2 left box2) (move c4-3 c4-2 down) (push c3-3 c4-3 c5-3 right box1) (move c2-3 c3-3 right) (move c1-3 c2-3 right) (move c1-2 c1-3 up) (move c2-2 c1-2 left) (push c1-2 c2-2 c3-2 right box2) (move c1-3 c1-2 down) (move c2-3 c1-3 left) (move c3-3 c2-3 left) (move c3-2 c3-3 up) (push c4-2 c3-2 c2-2 left box2) (push c5-2 c4-2 c3-2 left box2) (move c5-3 c5-2 down) (push c6-3 c5-3 c4-3 left box1) (move c6-4 c6-3 down) (move c5-4 c6-4 right) (move c4-4 c5-4 right) (move c4-3 c4-4 up) (push c3-3 c4-3 c5-3 right box1) (push c2-3 c3-3 c4-3 right box1) (push c1-3 c2-3 c3-3 right box1) (move c1-2 c1-3 up) (move c2-2 c1-2 left) (move c3-2 c2-2 left) (move c3-3 c3-2 down) (push c4-3 c3-3 c2-3 left box1) (push c4-4 c4-3 c4-2 down box2) (move c5-4 c4-4 left) (move c5-3 c5-4 up) (move c5-2 c5-3 up) (move c4-2 c5-2 right) (move c3-2 c4-2 right) (move c2-2 c3-2 right) (move c2-1 c2-2 up) (move c1-1 c2-1 right) (move c1-2 c1-1 down) (move c1-3 c1-2 down) (move c2-3 c1-3 left) (push c1-3 c2-3 c3-3 right box1) (move c1-2 c1-3 up) (move c2-2 c1-2 left) (push c2-1 c2-2 c2-3 up box1) (move c1-1 c2-1 right) (move c1-2 c1-1 down) (move c1-3 c1-2 down) (move c2-3 c1-3 left) (move c3-3 c2-3 left) (move c3-2 c3-3 up) (push c4-2 c3-2 c2-2 left box1) (push c5-2 c4-2 c3-2 left box1) (move c5-3 c5-2 down) (push c6-3 c5-3 c4-3 left box2) (move c6-4 c6-3 down) (move c5-4 c6-4 right) (move c4-4 c5-4 right) (move c4-3 c4-4 up) (push c3-3 c4-3 c5-3 right box2) (push c2-3 c3-3 c4-3 right box2) (push c1-3 c2-3 c3-3 right box2) (move c1-2 c1-3 up) (move c2-2 c1-2 left) (move c3-2 c2-2 left) (move c3-3 c3-2 down) (push c4-3 c3-3 c2-3 left box2) (push c4-4 c4-3 c4-2 down box1)

true.

***Τέλος, αξίζει να σημειωθεί ότι οι κινήσεις μέσω των οποίων φτάνουμε στο στόχο εμφανίζονται από την τελευταία προς την αρχική. Αυτό δεν επηρεάζει το αποτέλεσμα μιας και η λεξικογραφική ταξινόμηση των καταστάσεων γίνεται με βάση μόνο το State ως string όπως χαρακτηριστικά αναφέρεται και στη 2η άσκηση της 1ης εργασίας. Εύκολα θα μπορούσαμε να επεκτείναμε την λίστα ListOfActions βάζοντας την καινούργια ενέργεια (NewAction) κάθε φορά στο τέλος της CurrentListOfActions και όχι στην αρχή της, παίρνοντας έτσι την αντίστροφη σειρά στο Solution.*

Άσκηση 3(ii)

Το κατηγόρημα *astar/3* τροποποιήθηκε κατάλληλα ώστε να υλοποιεί μία αναδρομική εκδοχή της αναζήτησης που πραγματοποιείται με το loop της *AStarPlanner* που μας ζητήθηκε να υλοποιήσουμε στην 3η άσκηση της 1ης εργασίας του μαθήματος.

Ο κώδικας που υλοποιεί αυτή την αναδρομή είναι ο εξής:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% astar(+OpenStates, +VisitedStates, -Solution)

%%    astar/3 should implement a recursive depth-first search using an

%%    heurestic function for solving the parsed planning problem using progression.

%%    At the moment it only checks if the first pair in the list of open

%%    states, that is OpenStates, satisfies the goal.

%%

%% YOU NEED TO ADD CODE HERE!

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Check if the first state in the list of open states satisfies the goal.

astar(OpenStates, VisitedStates, Solution):-

        % Print useful debug information

        print('*Debug Information* A call to astar/3 has just been performed\n'),

%        print('OpenStates:' ),print(OpenStates), nl,

%        print('VisitedStates:' ),print(VisitedStates), nl,

        % Get the first pair from the list OpenStates

        [FirstPair |_] = OpenStates,

%        print('FirstPair:' ),print(FirstPair), nl,

        % Get the first element of the pair as the current list of actions

        % and the second element as the current state

        [Evaluation,CurrentListOfActions,CurrentState] = FirstPair,

        % Print some more debug information

%        print('Evaluation:' ),print(Evaluation), nl,

%        print('CurrentListOfActions:' ),print(CurrentListOfActions), nl,

%        print('CurrentState:' ),print(CurrentState), nl,

        % Check if the current state satisfies the goal
```

```
        satisfies_goal(CurrentState),
        % If we get up to here then the goal has been satisfied,
        % therefore we are done
        !,
        Solution = CurrentListOfActions.

% Then check if the first state in the list of open states is already visited.
astar(OpenStates, VisitedStates, Solution):-
        % Get the first pair from the list OpenStates
        [FirstPair | RestOfOpenStates] = OpenStates,
        % Get the first element of the pair as the current list of actions
        % and the second element as the current state
        [Evaluation,_CurrentListOfActions,CurrentState] = FirstPair,
                % Debug information -- uncomment to print useful information
                % print('*Debug* check state if it is already visited:\n'),
                % print(CurrentState), nl,
        % Check if the state description is already in the list of visited states
        member(CurrentState, VisitedStates),
        nl, nl,nl,print('CurrentState EXISTS in VisitedStates...Try next Action' ), nl,nl,nl,
        % If we get up to here, then it's a state we've seen before.
        % No need to expand this node using the next rule, so no more bactracking!
        !,
                % Debug information -- uncomment to print useful information
                % print('*Debug* state is already visited:\n'),
                % print(CurrentState), nl,
        % Continue search using the remaining list of open states.
        astar(RestOfOpenStates, VisitedStates, Solution).

% Finally, expand the first state in the list of open states and recurse.
astar(OpenStates, VisitedStates, Solution):-
```

```
        % Get the first pair from the list OpenStates
        [FirstPair | RestOfOpenStates] = OpenStates,
%        nl,print('FirstPair= ' ),print(FirstPair),
%        nl,print('RestOfOpenStates= ' ),print(RestOfOpenStates),
        % Get the first element of the pair as the current list of actions
        % and the second element as the current state
        [Evaluation,CurrentListOfActions,CurrentState] = FirstPair,
                % Debug information -- uncomment to print useful information
                % print('*Debug* expand state:\n'), print(CurrentState), nl,
                % Debug information -- uncomment to print useful information
                % print('*Debug* open states:\n'), print(OpenStates), nl,
        % Update the global variables that hold statistics about the search;
        % this should be called whenever a pair from OpenStates is processed
        % to produce the successor states that will be inserted in OpenStates.
        stat_node,
        % Print a dot for every node expanded.
        nl,print('.........................'), nl, flush_output,
        % Expand the current state by inserting the successor states in the
        % front of the list of open states, and recurse

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % *** ADD YOUR CODE HERE *** %
        % *** REPLACE NEXT LINE  *** %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %      Generates all the possible successor states from CurrentState. Action is
        %    an applicable ground action to State, and NextState is the
        %    successor state when the effects of Action are applied to State.
        findall([[NewAction      |      CurrentListOfActions],      NextState], progress(CurrentState, NewAction, NextState), NewOpenStates_noEval),
      % NewOpenStates_noEval is a list currying all NewOpenStates but without being evaluated
      % we want for every NewOpenState to find the Evaluation and make a new list of NewOpenStates where in the front there is the Evaluation
```

```
      TempOpenStates = [],

      % iteration predicate recursively evaluates all NewOpenStates_noEval resulting in the new evaluated NewOpenStates by using the helping list TempOpenStates

      iteration(NewOpenStates_noEval, TempOpenStates, NewOpenStates),

%        nl,print('NewOpenStates = ' ),print(NewOpenStates), nl,

% Append new OpenStates2 with old NewOpenStates

      append(NewOpenStates, RestOfOpenStates, OpenStates2),

%        nl,print('OpenStates2 = ' ),print(OpenStates2), nl,

% Sorting all of the OpenStates2 resulting in OpenStates3

      sort(OpenStates2,OpenStates3),

%        nl,print('SORTED OpenStates3 = ' ),print(OpenStates3), nl,

        astar(OpenStates3, [CurrentState | VisitedStates], Solution).



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% iteration(+NewOpenStates_noEval, +TempOpenStates, -NewOpenStates)

%%    iteration/3 should implement a recursive evaluation to all

%%             NewOpenStates_noEval(Old)   by   using   the   temporary   list TempOpenStates(Temp)

%%    for saving the current phases of the NewOpenList(New). When

%%    NewOpenStates_noEval(Old) becomes an empty list then the second iteration

%%    predicate is executed, assigning the TempOpenStates(Temp) to the

%%    NewOpenStates(New) so that to be returned!

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

iteration(Old, Temp, New):-

      [FirstNewOpenState | RestNewOpenStates] = Old,

      NewOpenStates = Temp,

%        nl,nl,print('--- FirstNewOpenState = ' ),print(FirstNewOpenState), nl,

%        nl,nl,print('--- RestNewOpenStates = ' ),print(RestNewOpenStates), nl,

        [NewListOfActions, NextState] = FirstNewOpenState,

      length(NewListOfActions,ListEvaluation),             % ListEvaluation is the length of NewListOfActions

%       nl,nl,print('--- ListEvaluation = ' ),print(ListEvaluation), nl,
```

```
      h(NextState,HeuresticEvaluation),                          %
HeuresticEvaluation is the heurestic cost of NewState

%        nl,nl,print('--- HeuresticEvaluation = ' ),print(HeuresticEvaluation),
nl,

%     NewEvaluation     =     g(NewListOfActions)     +     h(NewState,Goal)     =
length(NewListOfActions,ListEvaluation) + h(NextState,HeuresticEvaluation)

      NewEvaluation is ListEvaluation + HeuresticEvaluation,

%        nl,nl,print('--- NewEvaluation = ' ),print(NewEvaluation), nl,

      NewOpenState    =    [NewEvaluation,NewListOfActions,NextState],    %    new
OpenState created

        Temp2 = [NewOpenState | NewOpenStates],

      iteration(RestNewOpenStates, Temp2, New).


iteration(Old, Temp, New):-

        New = Temp.
```

Τα αποτελέσματα που παράγονται από την εκτέλεση του *planner* για το πρόβλημα (sokoban-domain.txt, sokoban-problem1.txt) είναι:

(Παρατίθενται μόνο τα αποτελέσματα της τελευταίας αναδρομικής κλήσης του κατηγορήματος... Αν ζητηθεί μπορώ να συμπεριλάβω και τα υπόλοιπα, αλλιώς μπορείτε να τα δείτε και εάν εκτελέσετε τον planner)

Debug Information A call to dfs/3 has just been performed

OpenStates:[[3,[push(c3-1,c3-2,c3-3,up,box1),move(c2-1,c3-1,right),move(c1-1,c2-1,right)],[empty(c1-1),empty(c1-2),empty(c1-3),empty(c2-1),empty(c3-1),robot-at(c3-2),object-at(box1,c3-3),adjacent(c1-1,c1-2,up),adjacent(c1-1,c2-1,right),adjacent(c1-2,c1-1,down),adjacent(c1-2,c1-3,up),adjacent(c1-2,c2-2,right),adjacent(c1-3,c1-2,down),adjacent(c1-3,c2-3,right),adjacent(c2-1,c1-1,left),adjacent(c2-1,c2-2,up),adjacent(c2-1,c3-1,right),adjacent(c2-2,c1-2,left),adjacent(c2-2,c2-1,down),adjacent(c2-2,c2-3,up),adjacent(c2-2,c3-2,right),adjacent(c2-3,c1-3,left),adjacent(c2-3,c2-2,up),adjacent(c2-3,c3-3,right),adjacent(c3-1,c2-1,left),adjacent(c3-1,c3-2,up),adjacent(c3-2,c2-2,left),adjacent(c3-2,c3-1,down),adjacent(c3-2,c3-3,up),adjacent(c3-3,c2-3,left),adjacent(c3-3,c3-2,down)]],[4,[move(c1-3,c1-2,down),move(c1-2,c1-3,up),move(c1-1,c1-2,up)],[empty(c1-1),empty(c1-3),empty(c2-1),empty(c3-1),empty(c3-3),robot-at(c1-2),object-at(box1,c3-2),adjacent(c1-1,c1-2,up),adjacent(c1-1,c2-1,right),adjacent(c1-2,c1-1,down),adjacent(c1-2,c1-3,up),adjacent(c1-2,c2-2,right),adjacent(c1-3,c1-2,down),adjacent(c1-3,c2-3,right),adjacent(c2-1,c1-1,left),adjacent(c2-1,c2-2,up),adjacent(c2-1,c3-1,right),adjacent(c2-2,c1-2,left),adjacent(c2-2,c2-1,down),adjacent(c2-2,c2-3,up),adjacent(c2-2,c3-2,right),adjacent(c2-3,c1-3,left),adjacent(c2-3,c2-2,up),adjacent(c2-3,c3-3,right),adjacent(c3-1,c2-1,left),adjacent(c3-1,c3-2,up),adjacent(c3-2,c2-2,left),adjacent(c3-2,c3-1,down),adjacent(c3-2,c3-3,up),adjacent(c3-3,c2-3,left),adjacent(c3-3,c3-2,down)]],[4,[move(c3-1,c2-1,left),move(c2-1,c3-1,right),move(c1-1,c2-1,right)],[empty(c1-1),empty(c1-2),empty(c1-3),empty(c3-1),empty(c3-3),robot-at(c2-1),object-at(box1,c3-2),adjacent(c1-1,c1-2,up),adjacent(c1-1,c2-1,right),adjacent(c1-2,c1-1,down),adjacent(c1-2,c1-3,up),adjacent(c1-2,c2-2,right),adjacent(c1-3,c1-2,down),adjacent(c1-3,c2-3,right),adjacent(c2-1,c1-1,left),adjacent(c2-1,c2-2,up),adjacent(c2-1,c3-1,right),adjacent(c2-2,c1-2,left),adjacent(c2-2,c2-1,down),adjacent(c2-2,c2-3,up),adjacent(c2-2,c3-2,right),adjacent(c2-3,c1-3,left),adjacent(c2-3,c2-2,up),adjacent(c2-3,c3-3,right),adjacent(c3-1,c2-1,left),adjacent(c3-1,c3-

2,up),adjacent(c3-2,c2-2,left),adjacent(c3-2,c3-1,down),adjacent(c3-2,c3-3,up),adjacent(c3-3,c2-3,left),adjacent(c3-3,c3-2,down)]]]

VisitedStates:[[empty(c1-1),empty(c1-2),empty(c1-3),empty(c2-1),empty(c3-3),robot-at(c3-1),object-at(box1,c3-2),adjacent(c1-1,c1-2,up),adjacent(c1-1,c2-1,right),adjacent(c1-2,c1-1,down),adjacent(c1-2,c1-3,up),adjacent(c1-2,c2-2,right),adjacent(c1-3,c1-2,down),adjacent(c1-3,c2-3,right),adjacent(c2-1,c1-1,left),adjacent(c2-1,c2-2,up),adjacent(c2-1,c3-1,right),adjacent(c2-2,c1-2,left),adjacent(c2-2,c2-1,down),adjacent(c2-2,c2-3,up),adjacent(c2-2,c3-2,right),adjacent(c2-3,c1-3,left),adjacent(c2-3,c2-2,up),adjacent(c2-3,c3-3,right),adjacent(c3-1,c2-1,left),adjacent(c3-1,c3-2,up),adjacent(c3-2,c2-2,left),adjacent(c3-2,c3-1,down),adjacent(c3-2,c3-3,up),adjacent(c3-3,c2-3,left),adjacent(c3-3,c3-2,down)],[empty(c1-1),empty(c1-2),empty(c2-1),empty(c3-1),empty(c3-3),robot-at(c1-3),object-at(box1,c3-2),adjacent(c1-1,c1-2,up),adjacent(c1-1,c2-1,right),adjacent(c1-2,c1-1,down),adjacent(c1-2,c1-3,up),adjacent(c1-2,c2-2,right),adjacent(c1-3,c1-2,down),adjacent(c1-3,c2-3,right),adjacent(c2-1,c1-1,left),adjacent(c2-1,c2-2,up),adjacent(c2-1,c3-1,right),adjacent(c2-2,c1-2,left),adjacent(c2-2,c2-1,down),adjacent(c2-2,c2-3,up),adjacent(c2-2,c3-2,right),adjacent(c2-3,c1-3,left),adjacent(c2-3,c2-2,up),adjacent(c2-3,c3-3,right),adjacent(c3-1,c2-1,left),adjacent(c3-1,c3-2,up),adjacent(c3-2,c2-2,left),adjacent(c3-2,c3-1,down),adjacent(c3-2,c3-3,up),adjacent(c3-3,c2-3,left),adjacent(c3-3,c3-2,down)],[empty(c1-1),empty(c1-2),empty(c1-3),empty(c3-1),empty(c3-3),robot-at(c2-1),object-at(box1,c3-2),adjacent(c1-1,c1-2,up),adjacent(c1-1,c2-1,right),adjacent(c1-2,c1-1,down),adjacent(c1-2,c1-3,up),adjacent(c1-2,c2-2,right),adjacent(c1-3,c1-2,down),adjacent(c1-3,c2-3,right),adjacent(c2-1,c1-1,left),adjacent(c2-1,c2-2,up),adjacent(c2-1,c3-1,right),adjacent(c2-2,c1-2,left),adjacent(c2-2,c2-1,down),adjacent(c2-2,c2-3,up),adjacent(c2-2,c3-2,right),adjacent(c2-3,c1-3,left),adjacent(c2-3,c2-2,up),adjacent(c2-3,c3-3,right),adjacent(c3-1,c2-1,left),adjacent(c3-1,c3-2,up),adjacent(c3-2,c2-2,left),adjacent(c3-2,c3-1,down),adjacent(c3-2,c3-3,up),adjacent(c3-3,c2-3,left),adjacent(c3-3,c3-2,down)],[empty(c1-1),empty(c1-3),empty(c2-1),empty(c3-1),empty(c3-3),robot-at(c1-2),object-at(box1,c3-2),adjacent(c1-1,c1-2,up),adjacent(c1-1,c2-1,right),adjacent(c1-2,c1-1,down),adjacent(c1-2,c1-3,up),adjacent(c1-2,c2-2,right),adjacent(c1-3,c1-2,down),adjacent(c1-3,c2-3,right),adjacent(c2-1,c1-1,left),adjacent(c2-1,c2-2,up),adjacent(c2-1,c3-1,right),adjacent(c2-2,c1-2,left),adjacent(c2-2,c2-1,down),adjacent(c2-2,c2-3,up),adjacent(c2-2,c3-2,right),adjacent(c2-3,c1-3,left),adjacent(c2-3,c2-2,up),adjacent(c2-3,c3-3,right),adjacent(c3-1,c2-1,left),adjacent(c3-1,c3-2,up),adjacent(c3-2,c2-2,left),adjacent(c3-2,c3-1,down),adjacent(c3-2,c3-3,up),adjacent(c3-3,c2-3,left),adjacent(c3-3,c3-2,down)],[empty(c1-2),empty(c1-3),empty(c2-1),empty(c3-1),empty(c3-3),robot-at(c1-1),object-at(box1,c3-2),adjacent(c1-1,c1-2,up),adjacent(c1-1,c2-1,right),adjacent(c1-2,c1-1,down),adjacent(c1-2,c1-3,up),adjacent(c1-2,c2-2,right),adjacent(c1-3,c1-2,down),adjacent(c1-3,c2-3,right),adjacent(c2-1,c1-1,left),adjacent(c2-1,c2-2,up),adjacent(c2-1,c3-1,right),adjacent(c2-2,c1-2,left),adjacent(c2-2,c2-1,down),adjacent(c2-2,c2-3,up),adjacent(c2-2,c3-2,right),adjacent(c2-3,c1-3,left),adjacent(c2-3,c2-2,up),adjacent(c2-3,c3-3,right),adjacent(c3-1,c2-1,left),adjacent(c3-1,c3-2,up),adjacent(c3-2,c2-2,left),adjacent(c3-2,c3-1,down),adjacent(c3-2,c3-3,up),adjacent(c3-3,c2-3,left),adjacent(c3-3,c3-2,down)]]

CurrentListOfActions:[push(c3-1,c3-2,c3-3,up,box1),move(c2-1,c3-1,right),move(c1-1,c2-1,right)]

CurrentState:[empty(c1-1),empty(c1-2),empty(c1-3),empty(c2-1),empty(c3-1),robot-at(c3-2),object-at(box1,c3-3),adjacent(c1-1,c1-2,up),adjacent(c1-1,c2-1,right),adjacent(c1-2,c1-1,down),adjacent(c1-2,c1-3,up),adjacent(c1-2,c2-2,right),adjacent(c1-3,c1-2,down),adjacent(c1-3,c2-3,right),adjacent(c2-1,c1-1,left),adjacent(c2-1,c2-2,up),adjacent(c2-1,c3-1,right),adjacent(c2-2,c1-2,left),adjacent(c2-2,c2-1,down),adjacent(c2-2,c2-3,up),adjacent(c2-2,c3-2,right),adjacent(c2-3,c1-3,left),adjacent(c2-3,c2-2,up),adjacent(c2-3,c3-3,right),adjacent(c3-1,c2-1,left),adjacent(c3-1,c3-2,up),adjacent(c3-2,c2-2,left),adjacent(c3-2,c3-1,down),adjacent(c3-2,c3-3,up),adjacent(c3-3,c2-3,left),adjacent(c3-3,c3-2,down)]

Solution found!

Domain name: sokoban

Problem name: sokoban1

Time: 0.063 sec

Search length: 5 nodes

Solution length: 3 actions

Solution: (push c3-1 c3-2 c3-3 up box1) (move c2-1 c3-1 right) (move c1-1 c2-1 right)

true

Αντίστοιχα, τα αποτελέσματα που παράγονται από την εκτέλεση του *planner* για το πρόβλημα (sokoban-domain.txt, sokoban-problem2.txt) είναι:

(Παρατίθενται μόνο τα τελικά αποτελέσματα, γιατί μόνο τα αποτελέσματα από την τελευταία αναδρομική κλήση του κατηγορήματος ξεπερνούν τις 10 σελίδες… Αν ζητηθεί μπορώ να συμπεριλάβω και τα υπόλοιπα, αλλιώς μπορείτε να τα δείτε και εάν εκτελέσετε τον planner)

Debug Information A call to dfs/3 has just been performed

…(Τα αποτελέσματα από την τελευταία εκτέλεση παραλείπονται λόγω πάρα πολύ μεγάλης έκτασης)…

Solution found!

Domain name: sokoban

Problem name: sokoban2

Time: 19.687 sec

Search length: 983 nodes

Solution length: 25 actions

Solution: (push c5-3 c6-3 c7-3 right box1) (push c4-3 c5-3 c6-3 right box1) (move c4-2 c4-3 up) (move c5-2 c4-2 left) (move c6-2 c5-2 left) (move c6-3 c6-2 down) (move c6-4 c6-3 down) (move c5-4 c6-4 right) (push c5-5 c5-4 c5-3 down box1) (push c5-6 c5-5 c5-4 down box1) (push c5-7 c5-6 c5-5 down box1) (push c5-8 c5-7 c5-6 down box1) (move c4-8 c5-8 right) (move c4-7 c4-8 up) (push c3-7 c4-7 c5-7 right box1) (push c2-7 c3-7 c4-7 right box1) (move c2-6 c2-7 up) (move c3-6 c2-6 left) (push c3-5 c3-6 c3-7 up box1) (move c2-5 c3-5 right) (move c2-4 c2-5 up) (move c2-3 c2-4 up) (move c3-3 c2-3 left) (move c4-3 c3-3 left) (move c5-3 c4-3 left)

true .

Τέλος, τα αποτελέσματα που παράγονται από την εκτέλεση του *planner* για το πρόβλημα (sokoban-domain.txt, sokoban-problem3.txt) είναι:

(Παρατίθενται μόνο τα τελικά αποτελέσματα, γιατί μόνο τα αποτελέσματα από την τελευταία αναδρομική κλήση του κατηγορήματος ξεπερνούν τις 10 σελίδες… Αν ζητηθεί μπορώ να συμπεριλάβω και τα υπόλοιπα, αλλιώς μπορείτε να τα δείτε και εάν εκτελέσετε τον planner)

Debug Information A call to dfs/3 has just been performed

…(Τα αποτελέσματα από την τελευταία εκτέλεση παραλείπονται λόγω πάρα πολύ μεγάλης έκτασης)…

Solution found!

Domain name: sokoban

Problem name: sokoban3

Time: 8.518 sec

Search length: 1358 nodes

Solution length: 30 actions

Solution: (push c2-3 c3-3 c4-3 right box1) (push c1-3 c2-3 c3-3 right box1) (move c1-2 c1-3 up) (move c2-2 c1-2 left) (push c2-1 c2-2 c2-3 up box1) (move c1-1 c2-1 right) (move c1-2 c1-1 down) (move c1-3 c1-2 down) (move c2-3 c1-3 left) (move c3-3 c2-3 left) (move c4-3 c3-3 left) (push c3-3 c4-3 c5-3 right box2) (move c3-2 c3-3 up) (push c4-2 c3-2 c2-2 left box1) (push c5-2 c4-2 c3-2 left box1) (move c5-3 c5-2 down) (push c6-3 c5-3 c4-3 left box2) (move c6-4 c6-3 down) (move c5-4 c6-4 right) (move c4-4 c5-4 right) (move c4-3 c4-4 up) (push c3-3 c4-3 c5-3 right box2) (push c2-3 c3-3 c4-3 right box2) (push c1-3 c2-3 c3-3 right box2) (move c1-2 c1-3 up) (move c2-2 c1-2 left) (move c3-2 c2-2 left) (move c3-3 c3-2 down) (push c4-3 c3-3 c2-3 left box2) (push c4-4 c4-3 c4-2 down box1)

True

***Τέλος, αξίζει να σημειωθεί ότι οι κινήσεις μέσω των οποίων φτάνουμε στο στόχο εμφανίζονται από την τελευταία προς την αρχική. Αυτό δεν επηρεάζει το αποτέλεσμα μιας και η λεξικογραφική ταξινόμηση των καταστάσεων γίνεται με βάση μόνο το State ως string όπως χαρακτηριστικά αναφέρεται και στη 2η άσκηση της 1ης εργασίας. Εύκολα θα μπορούσαμε να επεκτείναμε την λίστα ListOfActions βάζοντας την καινούργια ενέργεια (NewAction) κάθε φορά στο τέλος της CurrentListOfActions και όχι στην αρχή της, παίρνοντας έτσι την αντίστροφη σειρά στο Solution.*

Άσκηση 3(iii)

	sokoban-problem1			**sokoban-problem2**			**sokoban-problem3**		
	Time (sec)	***Search Length (nodes)***	***Solution Length (actions)***	***Time (sec)***	***Search Length (nodes)***	***Solution Length (actions)***	***Time (sec)***	***Search Length (nodes)***	***Solution Length (actions)***
3(i). dfs/3	*0.031*	*5*	*3*	*24.008*	*1274*	*107*	*20.951*	*3069*	*108*
3(ii). astar/3	*0.063*	*5*	*3*	*19.687*	*983*	*25*	*8.518*	*1358*	*30*

Παρατηρήσεις:

1) ***Search Length:*** Σε καθένα από τα προβλήματα παρατηρούμε ότι το μήκος της λίστας των καταστάσεων που εξερευνήθηκαν είναι ίδιο ή μικρότερο για την περίπτωση που η αναζήτηση έγινε με τον αλγόριθμο *AStarPlanner* σε αντίθεση με τον *DepthFirstPlanner*. Αυτό συμβαίνει γιατί η *dfs* αναζήτηση είναι πιθανό να "μπερδεύεται" και να αναζητά τη λύση ανάμεσα σε χιλιάδες μονοπάτια καταστάσεων και άρα να αργεί πολύ μέχρις ότου να πέσει στο σωστό μονοπάτι, το οποίο με τη σειρά του εξαρτάται από τις λεπτομέρειες της υλοποίησης που καθορίζουν πώς τοποθετούνται οι διάδοχες καταστάσεις στην αρχή της *OpenStates*, όπου εδώ:
   i. ανάμεσα σε δύο λεκτικά με τον ίδιο αριθμό ορισμάτων καθορίζεται από την *prolog* να προηγείται αυτό που προηγείται λεξικογραφικά αν συγκρίνουμε τα δυο λεκτικά σαν *strings*. Π.χ., το *Clear(C)* προηγείται του *OnTable(A)*, και το *On(A,B)* προηγείται του *On(A,C)* και
   ii. ανάμεσα σε δύο λεκτικά με διαφορετικό αριθμό ορισμάτων να προηγείται αυτό με τα λιγότερα ορίσματα. Π.χ., το *GripEmpty* που δεν έχει κανένα όρισμα προηγείται του *Clear(C)* το οποίο προηγείται του *On(A,B)*.

   Από την άλλη πλευρά η *astar* αναζήτηση επειδή κάνει αξιολόγηση κάθε φορά σε όλες τις καταστάσεις και τις ταξινομεί ανάλογα με μία ευρετική συνάρτηση (που στη συγκεκριμένη περίπτωση εξαρτάται από το μήκος της λίστας των ενεργειών και την ικανοποίηση μέρους του τελικού στόχου). Συνεπώς η *astar* επιλέγει έξυπνα την επόμενη κατάσταση που θα εξερευνήσει ώστε να πλησιάζει στο στόχο σε κάθε επόμενο βήμα και μάλιστα με τις λιγότερες δυνατές κινήσεις. Έτσι προφανώς, σε όλες τις περιπτώσεις η *astar* θα βρει την καλύτερη λύση και στις περισσότερες περιπτώσεις θα βρει τη λύση ψάχνοντας λιγότερους κόμβους, εν αντιθέσει με την *dfs*. Εκτός και εάν η τελευταία τύχει και βρει λύση στα πρώτα μονοπάτια που θα πάει να αναζητήσει.
2) ***Solution Length:*** Και σε αυτό το στατιστικό στοιχείο παρατηρούμε ότι η *AStarPlanner* είναι καλύτερη από την *DepthFirstPlanner*. Αυτό οφείλεται στο λόγο ότι η *dfs* κάνει ταξινόμηση μόνο στα καινούργια *OpenStates* τα οποία στη συνέχεια απλά τα βάζει μπροστά από τα παλιότερα *OpenStates* που είχαν βρεθεί από παλιότερες ενέργειες. Λειτουργεί ακριβώς δηλαδή όπως πρέπει να ορίζεται η αναζήτηση πρώτα σε βάθος*(dfs)*, πρώτα ψάχνει εντελώς ένα μονοπάτι και έπειτα αν δεν βρει λύση σε αυτό τότε συνεχίζει την αναζήτηση σε κάποιο άλλο μονοπάτι. Αντίθετα, η *astar* όπως είπαμε λόγω του τρόπου που λειτουργεί η *prolog* και τον οποίο περιγράψαμε παραπάνω, κάνει μία αρχική ταξινόμηση στα *OpenStates* με τον ίδιο τρόπο που κάνει και η *dfs*, αλλά έπειτα ξανακάνει ταξινόμηση σε <u>όλες</u> τις καταστάσεις (παλιές και καινούργιες) με βάση μία συνάρτηση αξιολόγησης η οποία όπως προαναφέραμε εξαρτάται από το μήκος της λίστας των ενεργειών και την ικανοποίηση μέρους του τελικού στόχου. Έτσι πάντα διαλέγει την πιο σύντομη και συνάμα κοντινή κίνηση στο στόχο. Σε αυτό οφείλεται και ότι για παράδειγμα στο *sokoban-problem3* η *astar* βρίσκει λύση σε λιγότερες από το *1/3* κινήσεις από αυτές που βρίσκει λύση η *dfs*. Πιο αναλυτικά, αυτό συμβαίνει γιατί η *dfs* ακολούθησε μία πολύ μεγάλη διαδρομή μέσω ενός ή περισσότερων λάθος μονοπατιών από τα οποία τελικά κατέληξε στο τελευταίο μονοπάτι που ακολούθησε να βρει λύση αλλά μέσω πολλών κινήσεων. Από την άλλη πλευρά η *astar* αξιολόγησε σωστά την κάθε κίνησή της και ακολούθησε σίγουρα κάποιο διαφορετικό και πολύ πιο σύντομο μονοπάτι για να βρει λύση.
3) ***Time:*** Στα δύο δυσκολότερα προβλήματα η *astar* επειδή όπως είπαμε στις προηγούμενες παρατηρήσεις ακολούθησε σωστότερα μονοπάτια και άρα εξερεύνησε λιγότερους κόμβους για να φτάσει στο στόχο έκανε λιγότερο χρόνο από την αναζήτηση με *dfs*. Παρατηρούμε όμως ότι για το πρόβλημα *sokoban-problem1* όπου έχουν εξερευνηθεί ακριβώς οι ίδιες σε αριθμό καταστάσεις και με τους δύο αλγορίθμους η αναζήτηση με *dfs* είναι ταχύτερη από την αναζήτηση με *astar*. Αυτό οφείλεται κυρίως γιατί στο τέλος του αλγορίθμου της αναζήτησης *astar* γίνεται άλλη μία ταξινόμηση και μάλιστα σε όλα τα στοιχεία της λίστας *OpenStates* με αποτέλεσμα μία μικρή καθυστέρηση. Τέλος, μικρό μέρος της καθυστέρησης οφείλεται και στις επιπλέον εντολές που εκτελούνται εντός του κατηγορήματος *iteration* που κατασκευάσαμε για να βρίσκει ο αλγόριθμος το *evaluation* του κάθε νέου *NewOpenState* μετά από την εκτέλεση κάθε ενέργειας.

Άσκηση 4

Παρατηρήσεις:

1) Παρατηρούμε όπως εξηγήσαμε και στο προηγούμενο ερώτημα της προηγούμενης άσκηση ότι όταν τρέξουμε το πρόβλημα *simplegame-problem1.txt* με τον *planner* που κάνει *dfs* αναζήτηση σε σχέση με τον *planner* που κάνει *astar* αναζήτηση θα δούμε ότι θα βρεθεί λύση πολύ πιο γρήγορα με τον *astar* σε σχέση με τον *dfs*. Οι λόγοι είναι ακριβώς αυτοί που περιγράψαμε στο προηγούμενο ερώτημα.
2) Επίσης παρατηρούμε ότι και ο *planner_astar* που κατασκευάσαμε και το εκτελέσιμο *blackbox* μας δίνουν ακριβώς την ίδια λύση στον ίδιο περίπου χρόνο, οπότε μάλλον η σχεδίαση του planner μας έγινε σωστά.

Σχόλια:

1) Στην σχεδίαση την ενέργειας
(:action shoot
:parameters (?locationnpc ?locplayer ?direction ?gun)
:precondition (and (npc-at ?locationnpc)
(npc-facing ?direction)
(player-at ?locplayer)
(adjacent ?locationnpc ?locplayer ?direction)
(npc-holding ?gun)
(gun ?gun))
:effect (player-down))
)
προσθέσαμε ακόμα το precondition με το κόκκινο χρώμα ώστε να μην πυροβολεί το npc τον παίχτη από διαγώνια θέση.
2) Τέλος, για άλλη μία φορά αξίζει να σημειωθεί ότι οι κινήσεις μέσω των οποίων φτάνουμε στο στόχο εμφανίζονται από την τελευταία προς την αρχική. Αυτό δεν επηρεάζει το αποτέλεσμα μιας και η λεξικογραφική ταξινόμηση των καταστάσεων γίνεται με βάση μόνο το State ως string όπως χαρακτηριστικά αναφέρεται και στη $2^{η}$ άσκηση της $1^{ης}$ εργασίας. Εύκολα θα μπορούσαμε να επεκτείναμε την λίστα ListOfActions βάζοντας την καινούργια ενέργεια (NewAction) κάθε φορά στο τέλος της CurrentListOfActions και όχι στην αρχή της, παίρνοντας έτσι την αντίστροφη σειρά στο Solution.