ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

ΠΡΟΒΛΗΜΑ ΕΚΚΕΝΩΣΗΣ ΚΤΙΡΙΟΥ

ΟΝΟΜΑΤΕΠΩΝΥΜΟ:

ΠΙΚΡΙΔΑΣ ΜΕΝΕΛΑΟΣ (141291)

ΡΗΓΑΣ ΑΝΔΡΕΑΣ (141257)

ΣΤΑΘΑΚΟΠΟΥΛΟΣ ΚΩΝΣΤΑΝΤΙΝΟΣ (161041)

ΤΜΗΜΑ: ΔΕΥΕΤΡΑ 14:00-16:00 (Β2)

Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών Πανεπιστημίου Δυτικής Αττικής

ΠΕΡΙΕΧΟΜΕΝΑ

1.Διατύπωση του προβλήματος

ΤΟ ΠΡΟΒΛΗΜΑ ΤΗΣ ΕΚΚΕΝΩΣΗΣ ΚΤΗΡΙΟΥ

Σε μία πολυώροφη πολυκατοικία υπάρχει ένα άδειο ασανσέρ στο ισόγειο,2 κάτοικοι στον $1^{ο}$ όροφο, 6 κάτοικοι στον $2^{ο}$ και 4 κάτοικοι στον $3^{ο}$ όροφο. Το ασανσέρ είναι αρχικά άδειο και βρίσκεται στο ισόγειο. Θέλουμε να εκκενώσουμε το κτίριο. Οι δυνατές ενέργειες είναι:

- **Μετακίνηση του ασανσέρ από οποιονδήποτε όροφο σε οποιονδήποτε όροφο, αρκεί να μην είναι πλήρες.**
- **Το ασανσέρ μπορεί να χωρέσει μέχρι και 5 άτομα από οποιονδήποτε όροφο.**
- **Αν το ασανσέρ είναι πλήρες πρέπει να πάει στο ισόγειο.**
- **Το ασανσέρ μπορεί να αδειάσει στο ισόγειο, αν έχει γεμίσει ή αν δεν έχει γεμίσει και δεν υπάρχουν άτομα στους υπόλοιπους ορόφους.**

2.Επίλυση προβλήματος με χώρο καταστάσεων.

ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ

ΧΩΡΟΣ ΚΑΤΑΣΤΑΣΕΩΝ

Οι επιτρεπτές καταστάσεις του προβλήματος αποτελούνται από το συνδυασμό των σχέσεων μεταξύ των αντικειμένων του.

ΑΡΧΙΚΗ ΚΑΤΑΣΤΑΣΗ

Το ασανσέρ βρίσκεται στο ισόγειο και είναι άδειο.

ΤΕΛΙΚΗ ΚΑΤΑΣΤΑΣΗ

Το κτίριο είναι άδειο.

ΤΕΛΕΣΤΕΣ ΠΡΟΒΛΗΜΑΤΟΣ

Σε αυτό το περιβάλλον μπορούν να εφαρμοστούν σε κάθε κατάσταση μόνο οι παρακάτω τελεστές:

- **Ανέβασμα του ασανσέρ στον $1^{Ο}$ όροφο: ElevatorUP -UP1.**
- **Ανέβασμα του ασανσέρ στον $2^{Ο}$ όροφο: ElevatorUP -UP2.**
- **Ανέβασμα του ασανσέρ στον $3^{Ο}$ όροφο: ElevatorUP -UP3.**
- **Άδειασμα του ασανσέρ: ElevatorEMPTY -E.**

ΜΟΝΤΕΛΟ ΜΕΤΑΒΑΣΗΣ

Κάθε ένας από τους τελεστές μετάβασης έχει τα αναμενόμενα αποτελέσματα του όταν υπάρχουν οι απαραίτητες προϋποθέσεις μετάβασης.
Δεν επιτρέπονται:

- **Οι μετακινήσεις των κατοίκων από οποιονδήποτε σε οποιονδήποτε όροφο πλην του ισογείου.**
- **Η μετάβαση του ασανσέρ στο ισόγειο ενώ δεν είναι γεμάτο, αν υπάρχουν και άλλοι κάτοικοι σε οποιονδήποτε όροφο.**
- **Η μετάβαση του ασανσέρ στο ισόγειο ενώ είναι άδειο.**

Ο ΚΟΣΜΟΣ ΤΗΣ ΕΚΚΕΝΩΣΗΣ ΤΟΥ ΚΤΙΡΙΟΥ

ΑΝΤΙΚΕΙΜΕΝΑ	ΙΔΙΟΤΗΤΕΣ	ΣΧΕΣΕΙΣ
ΑΣΑΝΣΕΡ	ΒΡΙΣΚΕΤΑΙ ΣΤΟ ΙΣΟΓΕΙΟ	ΤΟ ΑΣΑΝΣΕΡ ΠΕΡΝΑΕΙ ΑΠΟ ΟΛΟΥΣ ΤΟΥΣ ΟΡΟΦΟΥΣ
-||-	ΕΙΝΑΙ ΑΔΕΙΟ	ΜΕΤΑΦΕΡΕΙ ΕΝΟΙΚΟΥΣ
ΙΣΟΓΕΙΟ 1ος ΟΡΟΦΟΣ	0 ΚΑΤΟΙΚΟΙ 2 ΚΑΤΟΙΚΟΙ	ΟΙ ΟΡΟΦΟΙ ΕΧΟΥΝ ΚΑΤΟΙΚΟΥΣ ΠΕΡΝΑΕΙ ΤΟ ΑΣΑΝΣΕΡ ΑΠΟ ΑΥΤΟΥΣ
2ος ΟΡΟΦΟΣ 3ος ΟΡΟΦΟΣ	6 ΚΑΤΟΙΚΟΙ 4 ΚΑΤΟΙΚΟΙ	-||-
ΚΑΤΟΙΚΟΙ	ΠΕΡΙΜΕΝΟΥΝ ΤΟ ΑΣΑΝΣΕΡ ΣΤΟΝ ΟΡΟΦΟ ΤΟΥΣ	ΠΕΡΝΟΥΝ ΤΟ ΑΣΑΝΣΕΡ ΚΑΙ ΠΑΝΕ ΣΤΟ ΙΣΟΓΕΙΟ

ΔΕΝΤΡΑ ΑΝΑΖΗΤΗΣΗΣ & ΜΙΝΙ ΕΠΕΞΕΓΗΣΗ ΤΩΝ ΑΛΓΟΡΙΘΜΩΝ ΑΝΑΖΗΤΗΣΗΣ.
Μέθοδοι:

1)Πρώτα σε Βάθος Αναζήτηση (DFS)

Στην πρώτα σε βάθος αναζήτηση ο αλγόριθμος παίρνει το αριστερότερο από ένα γονικό κόμβο και συνεχίζει να πηγαίνει αριστερά μέχρι να βγει σε αδιέξοδο ή να βρει την λύση. Ουσιαστικά, επεκτείνει το μέτωπο αναζήτησης προς αριστερά και προς το βάθος του δένδρου αναζήτησης. Η διαδικασία ολοκληρώνεται, όταν εντοπιστεί ένας κόμβος που αντιστοιχεί σε μια επιθυμητή τελική κατάσταση ή όταν εξαντληθεί η αναζήτηση, δηλαδή όταν όλα τα μονοπάτια καταλήξουν σε κόμβους που δεν μπορούν να επιλεγούν ως γονικοί κόμβοι.

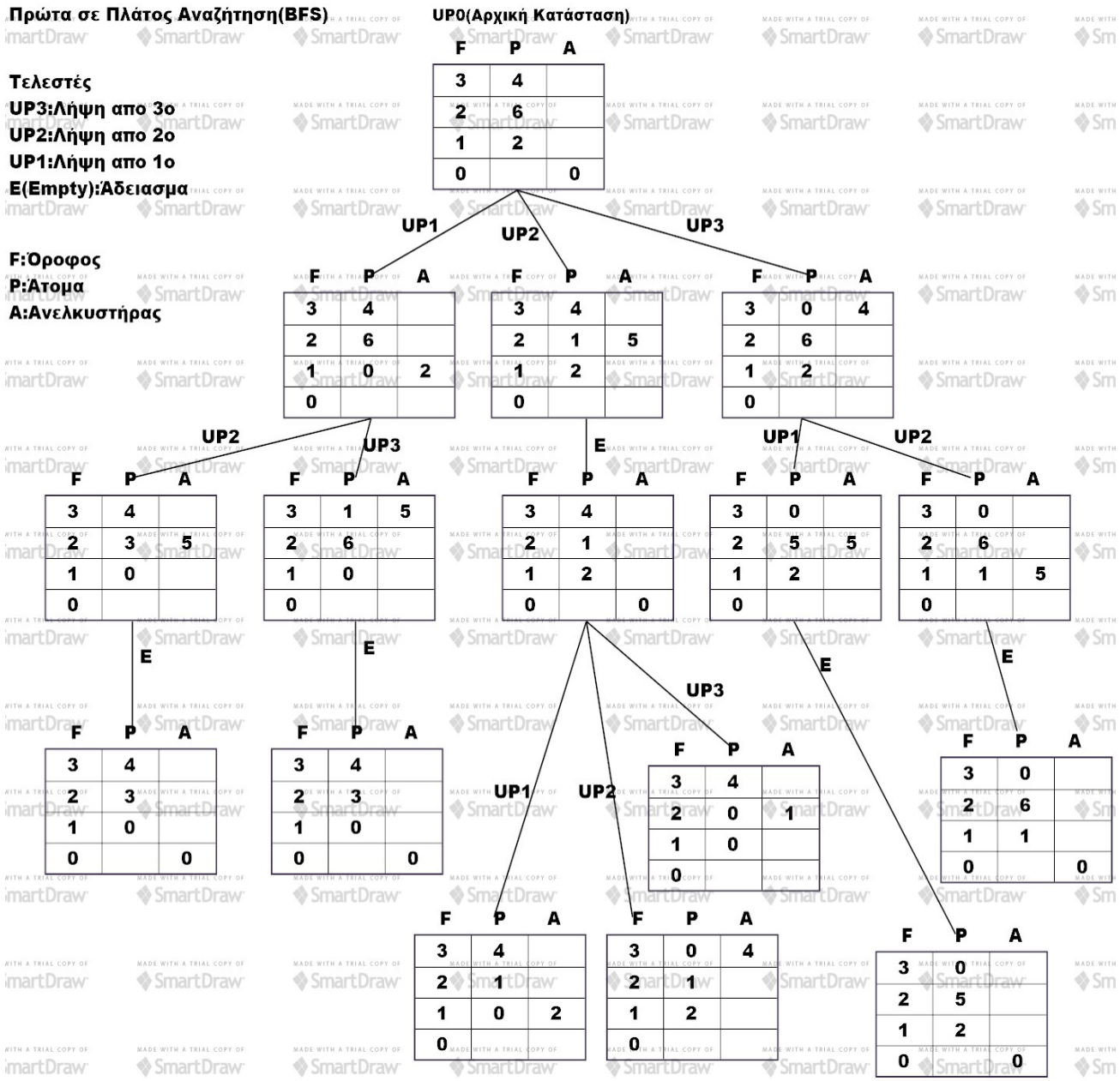
2)Πρώτα σε Πλάτος Αναζήτηση (BFS)

Στην κατά πλάτος αναζήτηση ο αλγόριθμος παίρνει από το αριστερά προς τα δεξιά τα παιδιά ενός γονικού κόμβου και τα αναλύει, δεν αλλάζει επίπεδο όμως αν δεν έχει αναλύσει όλους τους κόμβους-παιδιά ενός γονικού κόμβου. Ουσιαστικά, επεκτείνει το μέτωπο αναζήτησης ανά επίπεδο του δένδρου αναζήτησης. Η διαδικασία ολοκληρώνεται, όταν εντοπιστεί ένας κόμβος που αντιστοιχεί σε μια επιθυμητή τελική κατάσταση ή όταν εξαντληθεί η αναζήτηση, δηλαδή όταν όλα τα μονοπάτια καταλήξουν σε κόμβους που δεν μπορούν να επιλεγούν ως γονικοί κόμβοι.

Πρώτα σε Πλάτος Αναζήτηση(BFS)

Τελεστές
UP3:Λήψη απο 3ο
UP2:Λήψη απο 2ο
UP1:Λήψη απο 1ο
E(Empty):Άδειασμα

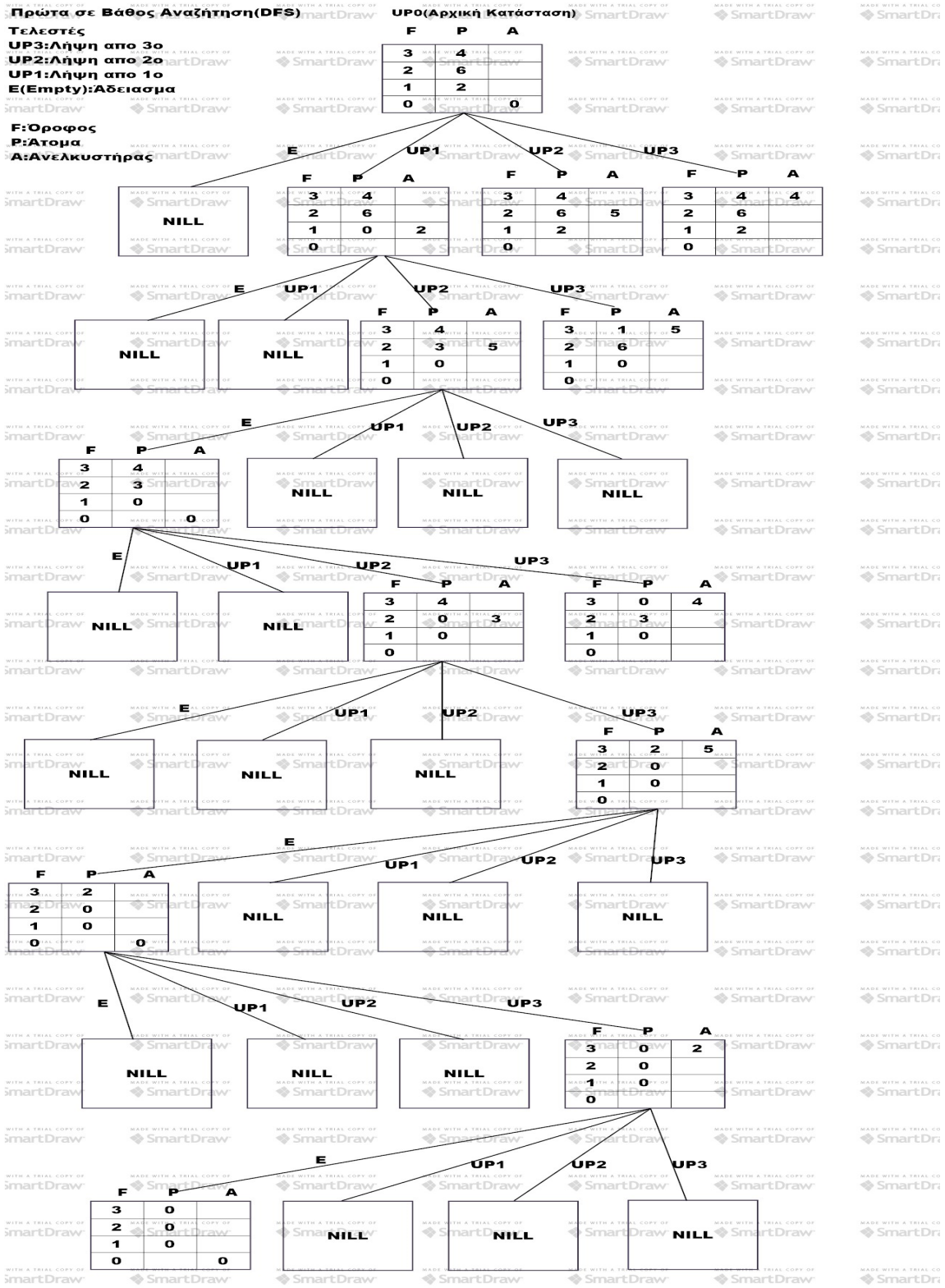
F:Όροφος
P:Άτομα
A:Ανελκυστήρας



Με παρακολούθηση του μετώπου αναζήτησης

Μέτωπο Αναζήτησης	Γονικός Κόμβος	Διαδοχοι Κόμβοι	Κλειστό Σύνολο
<UPO>	UP0	UP1,UP2,UP3	{ }
<UP1 UP2 UP3>	UP1	UP2,UP3	{UP0}
<UP2 UP3 UP2 UP3>	UP2	E	{UPO,UP1}
<UP3 UP2 UP3 E>	UP3	UP1,UP2	{UP0,UP1,UP2}
<UP2,UP3,E,UP1,UP2>	UP2	E	{UP0,UP1,UP2,UP3}
<UP3,E,UP1,UP2,E>	UP3	E	{UP0,UP1,UP2,UP3,UP2}
<E,UP1,UP2,E,E>	E	UP1,UP2,UP3	{UP0,UP1,UP2,UP3,UP2,UP3}
<UP1,UP2,E,E,UP1,UP2,UP3>	UP1	E	<UP0,UP1,UP2,UP3,UP2,UP3,E}
<UP2,E,E,UP1,UP2,UP3,E>	UP2	E	<UP0,UP1,UP2,UP3,UP2,UP3,E,UP1}

Μέχρι εδώ δεν έχει βρεθεί στόχος



Με παρακολούθηση του μετώπου αναζήτησης

Μέτωπο Αναζήτησης	Γονικός Κόμβος	Διάδοχοι Αναζήτης	Κλειστό Σύνολο
<UP0>	UP0	UP1,UP2,UP3	{ }
<UP1 UP2 UP3>	UP1	UP2, UP3	{UP0}
<UP2 UP3 UP2UP3>	UP2	E	{UPO,UP1}
<E UP3 UP2 UP3>	E	UP2,UP3	{UP0,UP1,UP2}
<UP2,UP3,UP3,UP2UP3>	UP2	UP3	{UP0,UP1,UP2,E}
<UP3 UP3 UP3 UP2 UP3>	UP3	E	{UP0,UP1,UP2,E,UP2,UP3}
<E UP3 UP3 UP2 UP3>	E	UP3	{UP0,UP1,UP2,E,UP2,UP3,E}
<UP3 UP3 UP3 UP2 UP3>	UP3	E	{UP0,UP1,UP2,E,UP2,UP3,E}
<E UP3 UP3 UP2 UP3>	E	-	{UPO,UP1,UP2,E,UP2,UP3,E,UP3}

ΒΡΕΘΗΚΕ Ο ΣΤΟΧΟΣ ΣΤΟΥΣ ΕΝΙΑ ΠΡΩΤΟΥΣ ΚΟΜΒΟΥΣ

3. Κωδικοποίηση του κόσμου του προβλήματος

ΟΡΙΣΜΟΣ ΚΑΤΑΣΤΑΣΗΣ

Κάθε κατάσταση στο πρόβλημα ορίζεται από μία λίστα (state) που περιέχει 5 τιμές, μια για την θέση του ασανσέρ, μια για τους κατοίκους που βρίσκονται μέσα στο ασανσέρ, μια για τους κατοίκους του 1ου ορόφου, μια για τους κατοίκους του 2ου ορόφου και μια για τους κατοίκους του 3ου ορόφου. Η μέγιστη χωρητικότητα του ασανσέρ ορίζεται από μία σταθερά (MAX) που περιέχει τον μέγιστο αριθμό κατοίκων που μπορεί να έχει το ασανσέρ.

ΤΡΟΠΟΣ ΚΩΔΙΚΟΠΟΙΗΣΗΣ

Το πρόβλημα σε αυτή τη μορφή αποτελείται από 1 ασανσέρ,3 ορόφους και από τους κατοίκους ανά όροφο, και μπορούμε να κωδικοποιήσουμε μία κατάσταση μέσα σε μία λίστα 5 στοιχείων. Κάθε στοιχείο της λίστας δηλώνει τον όροφο στον οποίο βρίσκεται το ασανσέρ, τον αριθμό των κατοίκων που υπάρχουν στο ασανσέρ, τον αριθμό των κατοίκων που υπάρχουν στον 1ο όροφο, τον αριθμό των κατοίκων που υπάρχουν στον 2ο όροφο και τον αριθμό των κατοίκων που υπάρχουν στον 1ο όροφο. Για παράδειγμα (x y a b c).

ΠΑΡΑΔΕΙΓΜΑ ΕΠΙΤΡΕΠΤΩΝ ΚΑΤΑΣΤΑΣΕΩΝ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ

1. **(0 0 2 6 4):**

   Το ασανσέρ βρίσκεται στο ισόγειο, έχει μέσα 0 κατοίκους, στον 1ο όροφο έχει 2 κατοίκους, στον 2ο όροφο έχει κατοίκους και στον 3ο όροφο έχει 4.

2. **(1 2 0 6 4):**

   Το ασανσέρ βρίσκεται στον 1ο όροφο, έχει μέσα 2 κατοίκους, στον 1ο όροφο έχει 0 κατοίκους, στον 2ο όροφο έχει 6 κατοίκους και στον 3ο όροφο έχει 4 κατοίκους.

3. **(3 4 0 6 0):**

   Το ασανσέρ βρίσκεται στον 3ο όροφο, έχει μέσα 4 κατοίκους, στον 1ο όροφο έχει 2 κατοίκους, στον 2ο όροφο έχει 6 κατοίκους και στον 3ο όροφο έχει 0 κατοίκους.

4. **(0 0 0 0 0):**

   Το ασανσέρ βρίσκεται στο ισόγειο, έχει μέσα 0 κατοίκους, στον 1ο όροφο έχει 0 κατοίκους, στον 2ο όροφο έχει 0 κατοίκους και στον 3ο όροφο έχει 0 κατοίκους. Το κτίριο είναι άδειο.

4.Ορισμός και Κωδικοποίηση των τελεστών μετάβασης

ΤΕΛΕΣΤΕΣ ΜΕΤΑΒΑΣΗΣ

ΟΙ ΤΕΛΕΣΤΕΣ ΜΕΤΑΒΑΣΗΣ ΓΙΑ ΤΟ ΣΥΓΚΕΚΡΙΜΕΝΟ ΠΡΟΒΛΗΜΑ ΕΙΝΑΙ 4:

ElevatorUP1

Ανέβασμα του ασανσέρ στον 1ο όροφο.

Προϋπόθεση: Το ασανσέρ να βρίσκεται Σε οποιονδήποτε όροφο και να μην είναι γεμάτο.

Αποτέλεσμα: Το ασανσέρ θα βρίσκεται στον 1ο όροφο.

ΚΩΔΙΚΑΣ

```
(defun elevatorup1 (state)
  (cond
    ((and (= (third state) 2) (<= (second state) 3))
            (list 1 (+ (second state) (third state)) 0 (fourth state) (fifth state)))
    ((and (= (second state) 4) (> (third state) 1))
            (list 1 5 (- (third state) 1) (fourth state) (fifth state)))
    ((and (= (third state) 1) (<= (second state) 4))
            (list 1 (+ (second state) (third state)) 0 (fourth state) (fifth state)))
    ;If the floor has 2 or less people they get into the elevator.
    (T NIL) ;It returns the list with next state
))


(elevatorup1 '(0 5 0 0 0)) ;Elevator_full
(elevatorup1 '(0 3 2 0 0)) ;(1 5 0 0 0)
(elevatorup1 '(0 2 0 6 2)) ;NIL
(elevatorup1 '(0 1 1 5 0)) ;(1 2 0 5 0)
(elevatorup1 '(0 4 1 0 0)) ;(1 5 0 0 0)
```

ElevatorUP2

Ανέβασμα του ασανσέρ στον 2ο όροφο.

<u>Προϋπόθεση:</u> Το ασανσέρ να βρίσκεται Σε οποιονδήποτε όροφο και να μην είναι γεμάτο.

<u>Αποτέλεσμα:</u> Το ασανσέρ θα βρίσκεται στον 2Ο όροφο.

ΚΩΔΙΚΑΣ

```
(defun elevatorup2 (state)
  (cond
      ((and (= (second state) 1) (>= (fourth state) 1) (<= (fourth state) 4))
                  (list 2 (+ (fourth state) (second state)) (third state) 0 (fifth state)))
      ((and (= (second state) 1) (= (fourth state) 5))
                  (list 2 5 (third state) 1 (fifth state)))
      ((and (= (second state) 1) (= (fourth state) 6))
                  (list 2 5 (third state) 2 (fifth state)))
      ((and (= (second state) 2) (>= (fourth state) 1) (<= (fourth state) 3))
                  (list 2 (+ (fourth state) (second state)) (third state) 0 (fifth state)))
      ((and (= (second state) 2) (= (fourth state) 4))
                  (list 2 5 (third state) 1 (fifth state)))
      ((and (= (second state) 2) (= (fourth state) 5))
                  (list 2 5 (third state) 2 (fifth state)))
      ((and (= (second state) 2) (= (fourth state) 6))
                  (list 2 5 (third state) 3 (fifth state)))
      ((and (= (second state) 3) (>= (fourth state) 1) (<= (fourth state) 2))
                  (list 2 (+ (fourth state) (second state)) (third state) 0 (fifth state)))
      ((and (= (second state) 3) (= (fourth state) 3))
                  (list 2 5 (third state) 1 (fifth state)))
      ((and (= (second state) 3) (= (fourth state) 4))
                  (list 2 5 (third state) 2 (fifth state)))
    ((and (= (second state) 3) (= (fourth state) 5))
                  (list 2 5 (third state) 3 (fifth state)))
      ((and (= (second state) 3) (= (fourth state) 6))
                  (list 2 5 (third state) 4 (fifth state)))
```

ΚΩΔΙΚΑΣ ΣΥΝΕΧΕΙΑ

```
 ((and (= (second state) 4) (>= (fourth state) 1) (<= (fourth state) 1))
                   (list 2 (+ (fourth state) (second state)) (third state) 0 (fifth state)))
      ((and (= (second state) 4) (= (fourth state) 2))
                   (list 2 5 (third state) 1 (fifth state)))
      ((and (= (second state) 4) (= (fourth state) 3))
                   (list 2 5 (third state) 2 (fifth state)))
    ((and (= (second state) 4) (= (fourth state) 4))
                   (list 2 5 (third state) 3 (fifth state)))
      ((and (= (second state) 4) (= (fourth state) 5))
                   (list 2 5 (third state) 4 (fifth state)))
    ((and (= (second state) 4) (= (fourth state) 6))
                   (list 2 5 (third state) 5 (fifth state)))
    ((and (= (fourth state) 6) (< (second state) volume))
            (list 2 5 (third state) (- (fourth state) (- volume (second state))) (fifth state)))
    ((and (< (fourth state) 6) (> (fourth state) 0) (< (second state)  volume))
            (list 2 (- volume (- volume (fourth state))) (third state) 0 (fifth state)))
    ;If the floor has 6 or less people they get into the elevator.
    (T NIL) ;It returns the list with next state
))

(elevatorup2 '(0 3 0 6 4)) ;(2 5 0 4 4)
(elevatorup2 '(0 4 0 6 4)) ;(2 5 0 5 4)
(elevatorup2 '(0 3 0 5 4)) ;(2 5 0 3 4)
(elevatorup2 '(0 4 0 3 4)) ;(2 5 0 2 4)
(elevatorup2 '(0 1 0 5 4)) ;(2 5 0 1 4)
(elevatorup2 '(0 0 0 5 4)) ;(2 5 0 0 4)
```

ElevatorUP3

Ανέβασμα του ασανσέρ στον 3ο όροφο.

<u>Προϋπόθεση:</u> Το ασανσέρ να βρίσκεται Σε οποιονδήποτε όροφο και να μην είναι γεμάτο.

<u>Αποτέλεσμα:</u> Το ασανσέρ θα βρίσκεται στον 3Ο όροφο.

ΚΩΔΙΚΑΣ

```
(defun elevatorup3 (state)
(cond
((and (<= (fifth state) 4) (= (second state) 0))
(list 3 (+ (second state) (fifth state)) (third state) (fourth state) 0))
((and (= (fifth state) 4) (<= (second state) 4))
(list 3 (+ (second state) (- volume (second state))) (third state) (fourth state)(- (fifth
state) (- volume (second state)))))
((and  (< (fifth state) 4) (> (fifth state) 0) (< (second state) 3))
(list 3 (+ (second state) (fifth state)) (third state) (fourth state)0))
((and  (< (fifth state) 3) (> (fifth state) 1) (< (second state) 4))
(list 3 (+ (second state) (fifth state)) (third state) (fourth state)
(- (fifth state) (- volume (second state))) ))
((and  (= (fifth state) 3)  (= (second state) 3))
(list 3 5 (third state) (fourth state) 1))
((and  (= (fifth state) 1) (= (second state) 3))
(list 3 4 (third state) (fourth state) 0))
((and (< (fifth state) 4) (> (fifth state) 0) (= (second state) 4))
(list 3 5 (third state) (fourth state) (- (fifth state) 1)))
 ;If the floor has 4 or less people they get into the elevator.
      (T NIL) ;It returns the list with next state
))
(elevatorup3 '(0 3 0 6 4)) ;NIL
(elevatorup3 '(0 1 0 6 4)) ;(3 5 0 6 0)
(elevatorup3 '(0 3 0 6 3)) ;(3 5 0 6 1)
(elevatorup3 '(0 4 0 6 3)) ;(3 5 0 6 2)
```

ElevatorEMPTY

Άδειασμα του ασανσέρ.

<u>Προϋπόθεση:</u> Το ασανσέρ να βρίσκεται στο ισόγειο και να έχει τουλάχιστον 1 άτομο μέσα.

<u>Αποτέλεσμα:</u> Το ασανσέρ θα βρίσκεται στο ισόγειο και θα είναι άδειο.

<u>ΚΩΔΙΚΑΣ</u>

```
(defun empty (state)
(cond
(
 (and (>= (second state) 0) (= 0 (third state)) (= 0 (fourth state)) (= 0 (fifth state)))
 (list 0 0 0 0 0)
 ; CHECKS IF ALL THE FLOORS ARE CLEAR AND IF THERE ARE PEPLE IN THE ELEVATOR IN ORDER TO
BRING THEM TO THE GORUND FLOOR.
)
(
 (= (second state) 5)
 (list 0 0 (third state) (fourth state) (fifth state))
)
 ;CHECK IF THE ELEVATOR IS FULL IN ORDER TO BRING THEM TO THE GROUND FLOOR.
      (T nil)
 ))
```

<u>ΣΥΝΑΡΤΗΣΗ FINDCHILDREN</u>

```
(defun findchildren (state)
(setq volume 5)
 (list
      (elevatorup1 state)
      (elevatorup2 state)
      (elevatorup3 state)
      (empty state)) ;It returns a list with the results of the 4
functions
)
```

5. Ο ευριστικός Αλγόριθμος Αναζήτησης «Best First»

Λίγα λόγια για τον Αλγόριθμο

Στον αλγόριθμο Best FS γίνεται πάντα επιλογή του καλύτερου. Ο συγκεκριμένος αλγόριθμος αναζήτησης, ορίζει τον πρώτο κόμβο του μετώπου ως γονικό, βρίσκει τα παιδιά του και τα τοποθετεί στην αρχή του μετώπου ενώ παράλληλα βάζει τον γονικό κόμβο στο κλειστό.

Πλεονεκτήματα του αλγόριθμου.

- **Προσπαθεί να δώσει μια πιο γρήγορη λύση σε κάποιο πρόβλημα.**
- **Εξαρτάται πολύ από τον ευριστικό μηχανισμό.**
- **Είναι πλήρης.**

Μειονεκτήματα του αλγόριθμου

- **Το μέτωπο αναζήτησης μεγαλώνει με υψηλό ρυθμό και μαζί του ο χώρος που χρειάζεται για την αποθήκευσή του.**
- **Δεν εγγυάται ότι η λύση που θα βρεθεί είναι η βέλτιστη.**

Ευρετικός Μηχανισμός:

Ο κατάλληλος ευρετικός μηχανισμός για το πρόβλημά μας(Εκκένωση Κτηρίου) θα μπορούσε να είναι ότι ο όροφος(F1,F2,F3) ,να γίνεται ο γονικός κόμβος εάν είναι εκείνος όπου θα έχει τον ιδανικό αριθμό ατόμων για να γεμίσει ο ανελκυστήρας, δηλαδή στην δικιά μας περίπτωση στα πέντε άτομα.

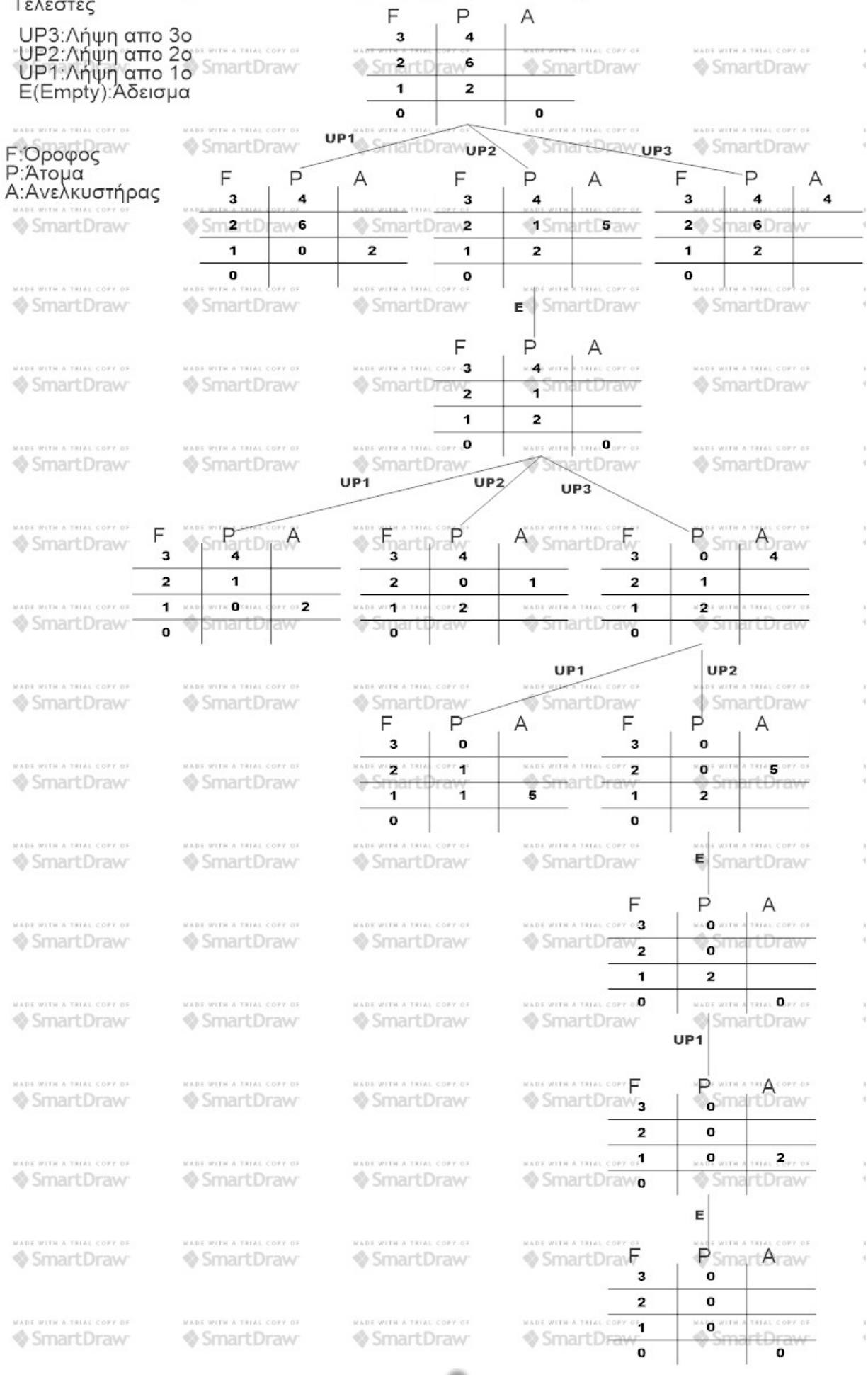
Αναζήτηση Καλύτερης Πρώτης

Τελεστές

UP3:Λήψη απο 3ο
UP2:Λήψη απο 2ο
UP1:Λήψη απο 1ο
E(Empty):Άδεισμα

F:Όροφος
P:Άτομα
A:Ανελκυστήρας

UP0(Αρχική Κατάσταση)



ΟΛΟΚΛΗΡΩΜΕΝΟΣ ΚΩΔΙΚΑΣ

```
;starting search


(defun searchProblem (start-state goal method )
  (print '____BEGIN_SEARCHING_____ )
    (findSolution (MakeFront start-state) (MakeQueue start-state) () goal method )
)

;Basic recursive function to create search tree (recursive tree expantion


(defun FindSolution (front queue closed goal method )
  (cond
    ((null front)                'no_solution)
    ((mymember (car front) closed) (FindSolution (cdr front) (cdr queue) closed  goal method  ))
    ((equal (car front) goal)      (print "This is the solution: ") (reverse (first queue)))
    (T      (FindSolution (ExpandFront front method) (ExtendQueue queue method)  (cons (car front)
closed) goal method  ))
  )
)

;initialization of front


(defun MakeFront (node)
  (list node)
)

;expanding front


(defun ExpandFront (front method)
  (cond
```

```
   ( (eq method 'DFS)    (append  (removeNils (findchildren (car front)))    (cdr front)))
   ( (eq method 'BFS)    (append (cdr front) (removeNils (findchildren (car front)))))
                         ( (eq method 'Bestfs) (append (rest front) (removeNils (findchildren (first front)))))
  )
)

;initialization of queue

(defun MakeQueue (node)
  (list (list node))
)



;expanding queue



(defun ExtendQueue (queue method)
  (cond
    ( (eq method 'DFS)    (append  (growPath (car queue))     (rest queue)  )  )
    ( (eq method 'BFS)    (append (rest queue) (growPath (car queue))))
    ( (eq method 'BestFS) (SORT  (append (rest queue)(growPath (first queue)) ) #'best-choice2) )
  )
)

;growing path towards each different child of the selected parent node



(defun growPath (path)
  (removecycles (grow1 path (removeNils (findchildren (car path)))))
)
```

```
(defun grow1 (path children)
  (cond
    ((null children) nil                                    )
    ( T          (cons (cons (car children) path) (grow1 path (cdr children)))  )
  )
)



;Supportive functions



(defun mymember(x Y)
  (cond
    ((endp y)          nil               )
    ((equal x (first y)) T                 )
    (T              (mymember x (rest y))  )
  )
)

(defun removeNils (X)
  (cond
    ((endp x)          nil                          )
    ((eq (first x) NIL) (removeNils (rest x))              )
    (T              (cons (first x) (removeNils (rest x)))  )
  )
)

(defun removecycles (paths)
  (cond
```

```
  ((null paths)                  nil                              )
  ((member (caar paths) (cdar paths)) (removecycles (cdr paths))              )
  (T                          (cons (car paths) (removecycles (cdr paths))) )
 )
)

(defun best-choice2 (state1 state2)
(cond
  ((< (FIRST (FIRST state1)) (FIRST (FIRST state2)))nil)
  (T nil)
))

;operators

(setq volume 5)

(defun elevatorup1 (state)
 (cond
                        ((or (= (third state) 0) (>= (third state) 3))nil)
  ((and (= (third state) 2) (<= (second state) 3))
       (list 1 (+ (second state) (third state)) 0 (fourth state) (fifth state)))
  ((and (= (second state) 4) (> (third state) 1))
       (list 1 5 (- (third state) 1) (fourth state) (fifth state)))
  ((and (= (third state) 1) (<= (second state) 4))
       (list 1 (+ (second state) (third state)) 0 (fourth state) (fifth state)))
  ;If the floor has 2 or less people they get into the elevator.
  (T NIL) ;It returns the list with next state
))
```

```
(defun elevatorup2 (state)
 (cond
                ((and (= (second state) 1) (>= (fourth state) 1) (<= (fourth state) 4))
                        (list 2 (+ (fourth state) (second state)) (third state) 0 (fifth state)))
                ((and (= (second state) 1) (= (fourth state) 5))
                        (list 2 5 (third state) 1 (fifth state)))
                ((and (= (second state) 1) (= (fourth state) 6))
                        (list 2 5 (third state) 2 (fifth state)))
                ((and (= (second state) 2) (>= (fourth state) 1) (<= (fourth state) 3))
                        (list 2 (+ (fourth state) (second state)) (third state) 0 (fifth state)))
                ((and (= (second state) 2) (= (fourth state) 4))
                        (list 2 5 (third state) 1 (fifth state)))
                ((and (= (second state) 2) (= (fourth state) 5))
                        (list 2 5 (third state) 2 (fifth state)))
                ((and (= (second state) 2) (= (fourth state) 6))
                        (list 2 5 (third state) 3 (fifth state)))
                ((and (= (second state) 3) (>= (fourth state) 1) (<= (fourth state) 2))
                        (list 2 (+ (fourth state) (second state)) (third state) 0 (fifth state)))
                ((and (= (second state) 3) (= (fourth state) 3))
                        (list 2 5 (third state) 1 (fifth state)))
                ((and (= (second state) 3) (= (fourth state) 4))
                        (list 2 5 (third state) 2 (fifth state)))
  ((and (= (second state) 3) (= (fourth state) 5))
                        (list 2 5 (third state) 3 (fifth state)))
                ((and (= (second state) 3) (= (fourth state) 6))
                        (list 2 5 (third state) 4 (fifth state)))
                ((and (= (second state) 4) (>= (fourth state) 1) (<= (fourth state) 1))
                        (list 2 (+ (fourth state) (second state)) (third state) 0 (fifth state)))
                ((and (= (second state) 4) (= (fourth state) 2))
                        (list 2 5 (third state) 1 (fifth state)))
```

```
                ((and (= (second state) 4) (= (fourth state) 3))
                                (list 2 5 (third state) 2 (fifth state)))
  ((and (= (second state) 4) (= (fourth state) 4))
                                (list 2 5 (third state) 3 (fifth state)))
                ((and (= (second state) 4) (= (fourth state) 5))
                                (list 2 5 (third state) 4 (fifth state)))
  ((and (= (second state) 4) (= (fourth state) 6))
                                (list 2 5 (third state) 5 (fifth state)))
  ((and (= (fourth state) 6) (< (second state) volume))
      (list 2 5 (third state) (- (fourth state) (- volume (second state))) (fifth state)))
  ((and (< (fourth state) 6) (> (fourth state) 0) (< (second state)  volume))
      (list 2 (- volume (- volume (fourth state))) (third state) 0 (fifth state)))
  ;If the floor has 6 or less people they get into the elevator.
  (T NIL) ;It returns the list with next state
))


(defun elevatorup3 (state)
 (cond
  ((= (fifth state) 0)nil)
  ((and (<= (fifth state) 4) (= (second state) 0))
      (list 3 (+ (second state) (fifth state)) (third state) (fourth state) 0))
  ((and (= (fifth state) 4) (<= (second state) 4))
      (list 3 (+ (second state) (- volume (second state))) (third state) (fourth state) (- (fifth state) (-
volume (second state)))))
  ((and  (< (fifth state) 4) (> (fifth state) 0) (< (second state) 3))
      (list 3 (+ (second state) (fifth state)) (third state) (fourth state) 0))
                ((and (< (fifth state) 4) (> (fifth state) 0) (= (second state) 4))
      (list 3 5 (third state) (fourth state) (- (fifth state) 1)))
  ((and  (< (fifth state) 3) (> (fifth state) 1) (< (second state) 4))
      (list 3 (+ (second state) (fifth state)) (third state) (fourth state) (- (fifth state) (- volume
```

```
(second state))) ))
   ((and  (= (fifth state) 3)  (= (second state) 3))
          (list 3 5 (third state) (fourth state) 1))
   ((and  (= (fifth state) 1) (= (second state) 3))
          (list 3 4 (third state) (fourth state) 0))
   ;If the floor has 4 or less people they get into the elevator.
   (T NIL) ;It returns the list with next state
))

(defun empty (state)
     (cond
        ((and (>= (second state) 0) (= 0 (third state)) (= 0 (fourth state)) (= 0 (fifth state))
            (list 0 0 0 0 0)))
        ; CHECKS IF ALL THE FLOORS ARE CLEAR AND IF THERE ARE PEOPLE IN THE ELEVATOR IN
ORDER TO BRING THEM TO THE GORUND FLOOR.

        ((= (second state) 5) (list 0 0 (third state) (fourth state) (fifth state)))
        ; CHECK IF THE ELEVATOR IS FULL IN ORDER TO BRING THE PEOPLE TO THE GROUND FLOOR.
        (T nil)
 ))
;function to find the children nodes of a parent state node

 (defun findchildren (state)
     (setq volume 5)
     (list  (elevatorup1 state)
          (elevatorup2 state)
          (elevatorup3 state)
                          (empty state))
          ;IT RETURNS A LIST WITH THE RESULTS OF THE 4 FUNCTIONS.
 )
```

ΑΠΟΤΕΛΕΣΜΑΤΑ ΚΩΔΙΚΑ.

Εκτέλεση με χρήση της trace για τον αλγόριθμο DFS.

```
CL-USER 33 > (searchproblem '(0 0 2 6 4) '(0 0 0 0 0) 'DFS)

____BEGIN_SEARCHING_____

"This is the solution: "
((0 0 2 6 4) (1 2 0 6 4) (2 5 0 3 4) (0 0 0 3 4) (2 3 0 0 4) (3 5 0 0 2) (0 0 0 0 2) (3
5 0 0 0) (0 0 0 0 0))

CL-USER 61 > (searchproblem '(0 0 2 6 4) '(1 2 0 6 4) 'DFS)
```

Εκτέλεση με χρήση της trace για τον αλγόριθμο BFS.

```
CL-USER 2 > (SEARCHPROBLEM '(0 0 2 6 4) '(0 0 0 0 0) 'BFS)

0 SEARCHPROBLEM > ...

  >> START-STATE : (0 0 2 6 4)

  >> GOAL       : (0 0 0 0 0)

  >> METHOD     : BFS


____BEGIN_SEARCHING_____

"This is the solution: "

0 SEARCHPROBLEM < ...

  << VALUE-0 : ((0 0 2 6 4) (2 5 2 1 4) (0 0 2 1 4) (2 1 2 0 4) (3 5 2 0 0) (0 0 2 0 0) (1 2 0 0 0) (0 0 0 0 0))

((0 0 2 6 4) (2 5 2 1 4) (0 0 2 1 4) (2 1 2 0 4) (3 5 2 0 0) (0 0 2 0 0) (1 2 0 0 0) (0 0 0 0 0))
```

Εκτέλεση με χρήση της trace για τον αλγόριθμο BestFS.

```
CL-USER 3 > (SEARCHPROBLEM '(0 0 2 6 4) '(0 0 0 0 0) 'BestFS)

0 SEARCHPROBLEM > ...

  >> START-STATE : (0 0 2 6 4)
```

>> GOAL : (0 0 0 0 0)

>> METHOD : BESTFS

____BEGIN_SEARCHING_____

"This is the solution: "

0 SEARCHPROBLEM < ...

<< VALUE-0 : ((0 0 2 6 4) (2 5 2 1 4) (0 0 2 1 4) (2 1 2 0 4) (3 5 2 0 0) (0 0 2 0 0) (1 2 0 0 0) (0 0 0 0 0))

((0 0 2 6 4) (2 5 2 1 4) (0 0 2 1 4) (2 1 2 0 4) (3 5 2 0 0) (0 0 2 0 0) (1 2 0 0 0) (0 0 0 0 0))

ΕΝΔΕΙΚΤΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΧΡΗΣΗ ΤΗΣ TRACE

CL-USER 1 > (SEARCHPROBLEM '(0 0 2 6 4) '(1 2 0 6 4) 'DFS)

____BEGIN_SEARCHING_____

"This is the solution: "

((0 0 2 6 4) (1 2 0 6 4))

CL-USER 2 > (SEARCHPROBLEM '(0 0 2 6 4) '(2 5 0 3 4) 'DFS)

____BEGIN_SEARCHING_____

"This is the solution: "

((0 0 2 6 4) (1 2 0 6 4) (2 5 0 3 4))

CL-USER 3 > (SEARCHPROBLEM '(0 0 2 6 4) '(0 0 0 3 4) 'DFS)

____BEGIN_SEARCHING_____

"This is the solution: "

((0 0 2 6 4) (1 2 0 6 4) (2 5 0 3 4) (0 0 0 3 4))

CL-USER 4 > (SEARCHPROBLEM '(0 0 2 6 4) '(2 3 0 0 4) 'DFS)

BEGIN SEARCHING

"This is the solution: "

((0 0 2 6 4) (1 2 0 6 4) (2 5 0 3 4) (0 0 0 3 4) (2 3 0 0 4))

CL-USER 5 > (SEARCHPROBLEM '(0 0 2 6 4) '(2 5 0 0 2) 'DFS)

BEGIN SEARCHING

NO SOLUTION

CL-USER 6 > (SEARCHPROBLEM '(0 0 2 6 4) '(0 0 0 0 0) 'DFS)

BEGIN SEARCHING

"This is the solution: "

((0 0 2 6 4) (1 2 0 6 4) (2 5 0 3 4) (0 0 0 3 4) (2 3 0 0 4) (3 5 0 0 2) (0 0 0 0 2) (3 2 0 0 0) (0 0 0 0 0))

CL-USER 7 > (SEARCHPROBLEM '(0 0 2 6 4) '(3 5 0 0 2) 'DFS)

BEGIN SEARCHING

"This is the solution: "

((0 0 2 6 4) (1 2 0 6 4) (2 5 0 3 4) (0 0 0 3 4) (2 3 0 0 4) (3 5 0 0 2))

CL-USER 12 : 1 > (SEARCHPROBLEM '(0 0 2 6 4) '(0 0 0 0 0) 'BFS)

BEGIN SEARCHING

"This is the solution: "

((0 0 2 6 4) (2 5 2 1 4) (0 0 2 1 4) (2 1 2 0 4) (3 5 2 0 0) (0 0 2 0 0) (1 2 0 0 0) (0 0 0 0 0))

CL-USER 15 : 1 > (SEARCHPROBLEM '(0 0 2 6 4) '(1 2 0 6 4) 'BFS)

BEGIN_SEARCHING

"This is the solution: "

((0 0 2 6 4) (1 2 0 6 4))

CL-USER 16 : 1 > (SEARCHPROBLEM '(0 0 2 6 4) '(2 5 2 1 4) 'BFS)

BEGIN_SEARCHING

"This is the solution: "

((0 0 2 6 4) (2 5 2 1 4))

CL-USER 17 : 1 > (SEARCHPROBLEM '(0 0 2 6 4) '(0 0 0 3 4) 'BFS)

BEGIN_SEARCHING

"This is the solution: "

((0 0 2 6 4) (1 2 0 6 4) (2 5 0 3 4) (0 0 0 3 4))

CL-USER 18 : 2 > (SEARCHPROBLEM '(0 0 2 6 4) '(0 0 0 3 4) 'BestFS)

BEGIN_SEARCHING

"This is the solution: "

((0 0 2 6 4) (1 2 0 6 4) (2 5 0 3 4) (0 0 0 3 4))

CL-USER 19 : 2 > (SEARCHPROBLEM '(0 0 2 6 4) '(2 5 2 1 0) 'BestFS)

BEGIN_SEARCHING

NO SOLUTION

CL-USER 20 : 2 > (SEARCHPROBLEM '(0 0 2 6 4) '(3 5 0 0 2) 'BestFS)

BEGIN SEARCHING

"This is the solution: "

((0 0 2 6 4) (1 2 0 6 4) (2 5 0 3 4) (0 0 0 3 4) (2 3 0 0 4) (3 5 0 0 2))

ΥΓ Για να τρέξουμε τις εντολές πήγαμε Works -> Expressions -> Trace -> Trace στο πρόγραμμα LispWorks. Τα αποτελέσματα ΔΕΝ είναι τα ίδια μ αυτά που μας δείξατε στο τελευταίο εργαστήριο, εμφανισιακά.

ΥΓ2 Ο κώδικας της εργασίας αρχικά είχε γίνει μόνο από ένα άτομο, έπειτα όμως από δική σας υπόδειξη, ο κώδικας γράφτηκε και από τους τρείς μας.