ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
UNIVERSITY OF WEST ATTICA

Σχολή Μηχανικών

Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Τεχνητή Νοημοσύνη

Απαλλακτική Εργασία Εξαμήνου

«Το πρόβλημα της εκκένωσης κτηρίου προς την ταράτσα»

ΕΥΦΡΟΣΥΝΗ ΒΑΡΣΟΥ 21390021

ΑΓΓΕΛΟΣ ΚΩΝΣΤΑΝΤΙΝΟΣ ΜΕΝΤΖΕΛΟΣ 21390132

Τεκμηρίωση Εργασίας

1. *Να οριστούν και να αναπαρασταθούν κατάλληλα:*

   i. *Ο κόσμος του προβλήματος*

Αντικείμενα	Ιδιότητες	Σχέσεις
Όροφος 1	Παρουσία ενοίκων για εκκένωση από το ασανσέρ.	9 ένοικοι
Όροφος 2		4 ένοικοι
Όροφος 3		12 ένοικοι
Όροφος 4		7 ένοικοι
Ταράτσα	Παρουσία ενοίκων από κάποιον όροφο που εκκενώθηκαν.	
Ισόγειο	Παρουσία Ασανσέρ	
Ασανσέρ	Χωρητικότητα ατόμων. Αριθμός ενοίκων που βρίσκονται μέσα. Σε ποιον όροφο βρίσκεται.	
Ένοικοι	Βρίσκονται σε κάποιον όροφο ή στο ασανσέρ.	

Πίνακας 1: Ο κόσμος του προβλήματος

   ii. *Η αρχική κατάσταση και η τελική κατάσταση του προβλήματος*

Ως αρχική κατάσταση έχουμε ότι το ασανσερ βρίσκεται στο ισόγειο (Όροφος 0), όλοι οι ένοικοι βρίσκονται στους αντίστοιχους ορόφους, η ταράτσα δεν έχει κανέναν ένοικο και το ασανσερ είναι άδειο.

Ως τελική κατάσταση έχουμε ότι το ασανσέρ βρίσκεται στην ταράτσα (Όροφος 5), όλοι οι όροφοι είναι άδειοι, η ταράτσα έχει όλους τους ενοίκους και το ασανσερ είναι άδειο.

Η αναπαραστασή τους σε λίστες αντίστοιχα είναι η εξής:

Αρχική Κατάσταση: `[0,9,4,12,7,0]`
Τελική Κατάσταση: `[5,0,0,0,0,0]`

2. *Να περιγραφούν οι τελεστές μετάβασης.*

Οι τελεστές μετάβασης του προβλήματος του ασανσέρ είναι οι ενέργειες που μπορούν να γίνουν, δηλαδή σε ποιον όροφο να πάει το ασανσέρ, για να αλλάξει η τρέχουσα κατάσταση n σε μια νέα κατάσταση n+1 του κόσμου του προβλήματος.

Όνομα Τελεστή Μετάβασης:	go_to_floor1
Περιγραφή/Ενέργεια:	Το ασανσέρ πηγαίνει στον πρώτο όροφο και παίρνει όσους ενοίκους χωράνε με σκοπό την εκκένωσή τους.
Προϋποθέσεις:	1. Τα άτομα που βρίσκονται στο ασανσέρ είναι λιγότερα της χωρητικότητας του (8 άτομα). 2. Στον πρώτο όροφο να υπάρχει τουλάχιστον ένας ένοικος. 3. Το ασανσέρ να μην βρίσκεται ήδη στον πρώτο όροφο.
Αποτέλεσμα/Μοντέλο Μετάβασης:	Το ασανσέρ βρίσκεται στον πρώτο όροφο. Το ασανσέρ είτε γεμίζει (8 άτομα) είτε αυξάνεται κατά τον αριθμό των ατόμων του πρώτου ορόφου. Ο αριθμός των ατόμων του πρώτου ορόφου είτε μειώνεται κατά την υπόλοιπη χωρητικότητα του ασανσέρ είτε μένει στάσιμος. (π.χ. [1,1,4,12,7,8])

Πίνακας 2: Τελεστής Μετάβασης go_to_floor1

Όνομα Τελεστή Μετάβασης:	go_to_floor2
Περιγραφή/Ενέργεια:	Το ασανσέρ πηγαίνει στον δεύτερο όροφο και παίρνει όσους ενοίκους χωράνε με σκοπό την εκκένωσή τους.
Προϋποθέσεις:	1. Τα άτομα που βρίσκονται στο ασανσέρ είναι λιγότερα της χωρητικότητας του (8 άτομα). 2. Στον δεύτερο όροφο να υπάρχει τουλάχιστον ένας ένοικος. 3. Το ασανσέρ να μην βρίσκεται ήδη στον δεύτερο όροφο.
Αποτέλεσμα/Μοντέλο Μετάβασης:	Το ασανσέρ βρίσκεται στον δεύτερο όροφο. Το ασανσέρ είτε γεμίζει (8 άτομα) είτε αυξάνεται κατά τον αριθμό των ατόμων του δεύτερου ορόφου. Ο αριθμός των ατόμων του δεύτερου ορόφου είτε μειώνεται κατά την υπόλοιπη χωρητικότητα του ασανσέρ είτε μένει στάσιμος. (π.χ. [2,9,0,12,7,4])

Πίνακας 3: Τελεστής Μετάβασης go_to_floor2

Όνομα Τελεστή Μετάβασης:	go_to_floor3
Περιγραφή/Ενέργεια:	Το ασανσέρ πηγαίνει στον τρίτο όροφο και παίρνει όσους ενοίκους χωράνε με σκοπό την εκκένωσή τους.
Προϋποθέσεις:	1. Τα άτομα που βρίσκονται στο ασανσέρ είναι λιγότερα της χωρητικότητας του (8 άτομα). 2. Στον τρίτο όροφο να υπάρχει τουλάχιστον ένας ένοικος. 3. Το ασανσέρ να μην βρίσκεται ήδη στον τρίτο όροφο.
Αποτέλεσμα/Μοντέλο Μετάβασης:	Το ασανσέρ βρίσκεται στον τρίτο όροφο. Το ασανσέρ είτε γεμίζει (8 άτομα) είτε αυξάνεται κατά τον αριθμό των ατόμων του τρίτου ορόφου. Ο αριθμός των ατόμων του τρίτου ορόφου είτε μειώνεται κατά την υπόλοιπη χωρητικότητα του ασανσέρ είτε μένει στάσιμος. (π.χ. [3,9,4,4,7,8])

Πίνακας 4: Τελεστής Μετάβασης go_to_floor3

Όνομα Τελεστή Μετάβασης:	go_to_floor4
Περιγραφή/Ενέργεια:	Το ασανσέρ πηγαίνει στον τέταρτο όροφο και παίρνει όσους ενοίκους χωράνε με σκοπό την εκκένωσή τους.
Προϋποθέσεις:	1. Τα άτομα που βρίσκονται στο ασανσέρ είναι λιγότερα της χωρητικότητας του (8 άτομα). 2. Στον τέταρτο όροφο να υπάρχει τουλάχιστον ένας ένοικος. 3. Το ασανσέρ να μην βρίσκεται ήδη στον τέταρτο όροφο.
Αποτέλεσμα/*Μοντέλο Μετάβασης:*	Το ασανσέρ βρίσκεται στον τρίτο όροφο. Το ασανσέρ είτε γεμίζει (8 άτομα) είτε αυξάνεται κατά τον αριθμό των ατόμων του τρίτου ορόφου. Ο αριθμός των ατόμων του τρίτου ορόφου είτε μειώνεται κατά την υπόλοιπη χωρητικότητα του ασανσέρ είτε μένει στάσιμος. (π.χ. [4,9,4,12,0,7])

Πίνακας 5: Τελεστής Μετάβασης go_to_floor4

Όνομα Τελεστή Μετάβασης:	go_to_top
Περιγραφή/Ενέργεια:	Το ασανσέρ φτάνει στην ταράτσα γεμάτο και εκκενώνει τους ενοίκους.
Προϋποθέσεις:	1. Το ασανσέρ να είναι πλήρης ή να μην υπάρχουν άλλοι ένοικοι σε κάποιον όροφο. 2. Τα άτομα του ασανσέρ να είναι λιγότερα η ίσα της χωρητικότητας του. 3. Το ασανσέρ να μην βρίσκεται ήδη στην ταράτσα.
Αποτέλεσμα/*Μοντέλο Μετάβασης:*	Το ασανσέρ βρίσκεται στην ταράτσα. Τα άτομα του ασανσέρ αδειάζουν. (π.χ. [5,1,4,12,7,0])

Πίνακας 6: Τελεστής Μετάβασης go_to_top

3. Να προσδιοριστεί ο χώρος καταστάσεων του προβλήματος (περιγραφικά και με λίστες), παραθέτοντας αντιπροσωπευτικά παραδείγματα.

Ο χώρος καταστάσεων αναφέρεται στο σύνολο των πιθανών καταστάσεων που μπορεί να υπάρξει το πρόβλημα. Για να είναι εφικτή μια κατάσταση πρέπει να πληρούνται οι εξής προϋποθέσεις:

1. Οι όροφοι του ασανσέρ θα πρέπει να κυμαίνονται από 0 έως 5.
2. Ο πρώτος όροφος να έχει από 0 έως 9 άτομα.
3. Ο δεύτερος όροφος να έχει από 0 έως 4 άτομα.
4. Ο τρίτος όροφος να έχει από 0 έως 12 άτομα.
5. Ο τέταρτος όροφος να έχει από 0 έως 7 άτομα.
6. Το ασανσέρ να έχει από 0 έως 8 άτομα.
7. Όταν το ασανσέρ βρίσκεται στο ισόγειο ή στην ταράτσα, να είναι άδειο.
8. Αν το ασανσέρ βρίσκεται στην ταράτσα, να έχει μειωθεί ο αριθμός των ατόμων σε κάποιον όροφο.

Εφικτοί χώροι καταστάσεων:

`[1,1,4,12,7,8]`

`[2,9,0,12,7,4]`

`[3,9,4,4,7,8]`

`[4,9,4,12,0,7]`

`[3,9,4,0,7,4]`

`[1,0,4,12,7,1]`

`[5,1,4,12,7,8]`

`[3,1,4,4,7,8]`

Μη-εφικτοί χώροι καταστάσεων:

`[2,9,6,3,7,0]`: Διότι υπάρχουν 9 άτομα στον δεύτερο όροφο ενώ οι ένοικοι είναι 4.

`[2,9,4,3,7,9]`: Διότι υπάρχουν 9 άτομα στο ασανσέρ ενώ η χωρητικότητα του είναι μέχρι 8.

`[6,1,4,12,7,8]`: Διότι όροφος του ασανσέρ είναι 6 ενώ δεν υπάρχει έκτος όροφος.

`[0,0,4,12,7,1]` : Διότι το ασανσέρ βρίσκεται στο ισόγειο και έχει άτομα.

`[5,9,4,12,7,0]` : Διότι βρίσκεται στην ταράτσα ενώ όλοι οι ένοικοι βρίσκονται στους ορόφους.

4. Να παρουσιαστεί σχολιασμένη η κωδικοποίηση του κόσμου του προβλήματος.

Τα αντικείμενα του κόσμου του προβλήματος και συνεπώς την κάθε κατάσταση του προβλήματος θα τα αναπαραστήσουμε μέσω μιας λίστας. Στους δείκτες θα αναπαριστώνται οι όροφοι και σαν τιμές τους θα είναι οι ένοικοι καθώς και σε δύο δείκτες οι ιδιότητες του ασανσέρ. Η ταράτσα και το ισόγειο θα αναπαριστώνται μέσω που βρίσκεται το ασανσέρ γιατί δεν χρειάζεται να ξέρουμε τον αριθμό των ατόμων τους. Πιο συγκεκριμένα, κάθε δείκτης αναπαριστά τα ακόλουθα:

[0]: Σε ποιον όροφο βρίσκεται το ασανσερ. (Ισόγειο, Όροφος 1-4, Ταράτσα)

[1]: Πόσα άτομα βρίσκονται στον πρώτο όροφο.

[2]: Πόσα άτομα βρίσκονται στον δεύτερο όροφο.

[3]: Πόσα άτομα βρίσκονται στον τρίτο όροφο.

[4]: Πόσα άτομα βρίσκονται στον τέταρτο όροφο.

[5]: Πόσα άτομα βρίσκονται στο ασανσέρ.

Για παράδειγμα, η λίστα `[1,1,4,12,7,8]` περιγράφει ότι το ασανσέρ είναι στον πρώτο όροφο και έχει 8 άτομα, ο πρώτος όροφος έχει 1 άτομο, ο δεύτερος όροφος έχει 4 άτομα, ο τρίτος όροφος έχει 12 άτομα και ο τέταρτος όροφος έχει 7 άτομα.

5. Να παρουσιαστούν σχολιασμένα η κωδικοποίηση των τελεστών μετάβασης του προβλήματος και η συνάρτηση εύρεσης απογόνων (findchildren).

Τις προϋποθέσεις των τελεστών θα τις υλοποιήσουμε μέσω μια δομής επιλογής και αναλόγως θα υπάρξουν τα ανάλογα αποτελέσματα.

Για παράδειγμα, ο πρώτος τελεστής μετάβασης go_to_floor1 έχει ως προϋποθέσεις τα άτομα του ασανσέρ να είναι λιγότερα του 8 (χωρητικότητα) και να υπάρχει τουλάχιστον ένας ένοικος στον πρώτο όροφο.

Συνεπώς, ο δείκτης [5] ή [-1] (-1 αναφέρεται στον τελευταίο δείκτη της λίστας) που περιγράφει πόσα άτομα βρίσκονται στο ασανσέρ της κατάστασης πρέπει να είναι < 8.

Επίσης, ο δείκτης [1] που αναφέρεται στα άτομα που βρίσκονται στον πρώτο όροφο πρέπει να είναι > 0.

Σαν αποτέλεσμα του τελεστή έχουμε δύο περιπτώσεις. Αν τα άτομα που βρίσκονται στον πρώτο όροφο είναι περισσότερα από τα άτομα που χωράνε στο ασανσέρ τότε το ασανσέρ πάει στον πρώτο όροφο και γεμίζει (8 άτομα) και τα άτομα του πρώτου ορόφου μειώνονται κατά πόσα μπήκανε στο ασανσέρ. Αλλιώς, αν τα άτομα που βρίσκονται στον πρώτο όροφο είναι λιγότερα από τα άτομα που χωράνε στο ασανσέρ τότε το ασανσέρ πάει στον πρώτο όροφο και αυξάνεται κατά τα άτομα του ορόφου και ο όροφος αδειάζει.

Επομένως, στην κωδικοποίηση αν `state[1] > 8 – state[-1]` (άτομα 1ου ορόφου > χωρητικότητα – άτομα ασανσέρ) στη νέα κατάσταση αλλάζει ο όροφος του ασανσέρ σε [1], τα άτομα του πρώτου ορόφου σε `[state[1] + state[-1] – 8]` και τα άτομα του ασανσέρ γίνονται `[8]` ενώ όλοι οι άλλοι όροφοι μένουν το ίδιο. Αλλιώς, στη νέα κατάσταση αλλάζει ο όροφος του ασανσέρ σε [1], τα άτομα του πρώτου ορόφου σε [0], και τα άτομα του ασανσέρ γίνονται `[state[1] + state[-1]]` ενώ όλοι οι άλλοι όροφοι μένουν το ίδιο.

Σύμφωνα με τα παραπάνω, υλοποιούνται αναλόγως και οι υπόλοιποι τελεστές μετάβασης go_to_floor2, go_to_floor3 και go_to_floor4 με μόνη αλλαγή σε ποιον όροφο γίνονται οι αλλαγές.

```
def go_to_floor1(state):
    if state[-1] < 8 and state[1] > 0:
        if state[1] > 8 - state[-1]:
            new_state = [1] + [state[1] + state[-1] - 8] + [state[2]] + [state[3]] + [state[4]] + [8]
        else:
            new_state = [1] + [0] + [state[2]] + [state[3]] + [state[4]] + [state[1] + state[-1]]
        return new_state

def go_to_floor2(state):
      if state[-1] < 8 and state[2] > 0:
        if state[2] > 8 - state[-1]:
            new_state = [2] + [state[1]] + [state[2] + state[-1] - 8] + [state[3]] + [state[4]] + [8]
        else:
            new_state = [2] + [state[1]] + [0] + [state[3]] + [state[4]] + [state[2] + state[-1]]
        return new_state

def go_to_floor3(state):
    if state[-1] < 8 and state[3] > 0:
        if state[3] > 8 - state[-1]:
            new_state = [3] + [state[1]] + [state[2]] + [state[3] + state[-1] - 8] + [state[4]] + [8]
        else:
            new_state = [3] + [state[1]] + [state[2]] + [0] + [state[4]] + [state[3] + state[-1]]
        return new_state

def go_to_floor4(state):
    if state[-1] < 8 and state[4] > 0:
        if state[4] > 8 - state[-1]:
            new_state = [4] + [state[1]] + [state[2]] + [state[3]] + [state[4] + state[-1] - 8] + [8]
        else:
            new_state = [4] + [state[1]] + [state[2]] + [state[3]] + [0] + [state[4] + state[-1]]
        return new_state
```

Τώρα, ο τελεστής μετάβασης go_to_top έχει ως προϋποθέσεις τα άτομα στο ασανσέρ να είναι 8 (γεμάτο) ή να μην υπάρχει κανένας άλλος ένοικος σε κάποιον όροφο, δηλαδή όλοι οι όροφοι να είναι άδειοι. Αν ισχύει τότε αλλάζει ο όροφος του ασανσέρ σε [5], όλοι οι όροφοι παραμένουν το ίδιο και το ασανσέρ αδειάζει.

```
def go_to_top(state):
    if state[-1] == 8 or (state[1] == 0 and state[2] == 0 and state[3] == 0 and state[4] == 0):
        new_state = [5] + [state[1]] + [state[2]] + [state[3]] + [state[4]] + [0]
        return new_state
```

Η συνάρτηση διαδόχων καλεί όλους τους τελεστές μετάβασης σε μια κατάσταση και παράγει/επιστρέφει όλες τις διάδοχες καταστάσεις (children). Συνεπώς, στην κωδικοποίηση αντιγράφουμε το state που δίνεται ως όρισμα για κάθε τελεστή μετάβασης και τους καλούμε και αν φέρουν αποτέλεσμα τα εισάγουμε σε μια λίστα children που περιέχει όλους τους διαδόχους.

```
def find_children(state):

    children=[]

    floor1_state=copy.deepcopy(state)
    floor1_child=go_to_floor1(floor1_state)

    floor2_state=copy.deepcopy(state)
    floor2_child=go_to_floor2(floor2_state)

    floor3_state=copy.deepcopy(state)
    floor3_child=go_to_floor3(floor3_state)

    floor4_state=copy.deepcopy(state)
    floor4_child=go_to_floor4(floor4_state)

    top_state=copy.deepcopy(state)
    top_child=go_to_top(top_state)

    if top_child != None:
        children.append(top_child)
    if floor4_child != None:
        children.append(floor4_child)
    if floor3_child != None:
        children.append(floor3_child)
    if floor2_child != None:
        children.append(floor2_child)
    if floor1_child != None:
        children.append(floor1_child)

    return children
```

6. Να περιγραφούν οι μέθοδοι αναζήτησης που υποστηρίζει η εργασία σας. Για την/τις ευριστικές μεθόδους αναζήτησης που θα χρησιμοποιηθούν να παρουσιαστεί το ευριστικό κριτήριο και ο ευριστικός μηχανισμός.

Τυφλές μέθοδοι αναζήτησης:

Οι τυφλές μέθοδοι αναζήτησης που υποστηρίζονται στο πρόβλημα είναι του Depth-First-Search (DFS) και του Breadth-First-Search (BFS). Ο DFS γενικά επισκέπτεται πρώτα το αριστερό υποδένδρο και μετά το δεξιό μιας δεδομένης αρχικής κατάστασης ενώ ο BFS πρώτα επισκέπτεται τα παιδία της αρχικής κατάστασης που βρίσκονται στο ίδιο επίπεδο και μετά προχωρά στα επόμενα παιδιά. Επομένως, ο DFS επισκέπτεται πιο βαθιά γιατί επισκέπτεται το πρώτο παιδί της κάθε κατάστασης, ενώ ο BFS πρώτα επισκέπτεται όλα τα παιδιά της κάθε κατάστασης και μετά προχωρά στην επόμενη.

Πιο συγκεκριμένα στο πρόβλημα μας, για τον DFS εφαρμόζεται στην τρέχουσα κατάσταση (node) η συνάρτηση εύρεσης απογόνων (findchildren) και προστίθενται τα παιδιά της στην αρχή του μετώπου (front και queue), με σκοπό στην συνέχεια όταν ξανά εφαρμοστεί η findchildren τα παιδία της τρέχουσας κατάστασης να επεκταθούν πρώτα. Ομοίως, για τον BFS εφαρμόζεται στην τρέχουσα κατάσταση (node) η συνάρτηση εύρεσης απογόνων (findchildren) και προστίθενται τα παιδιά της στο τέλος του μετώπου (front και queue), με σκοπό στην συνέχεια όταν ξανά εφαρμοστεί η findchildren να συνεχίσουν να επεκτείνονται τα παιδιά της προηγούμενης κατάστασης.

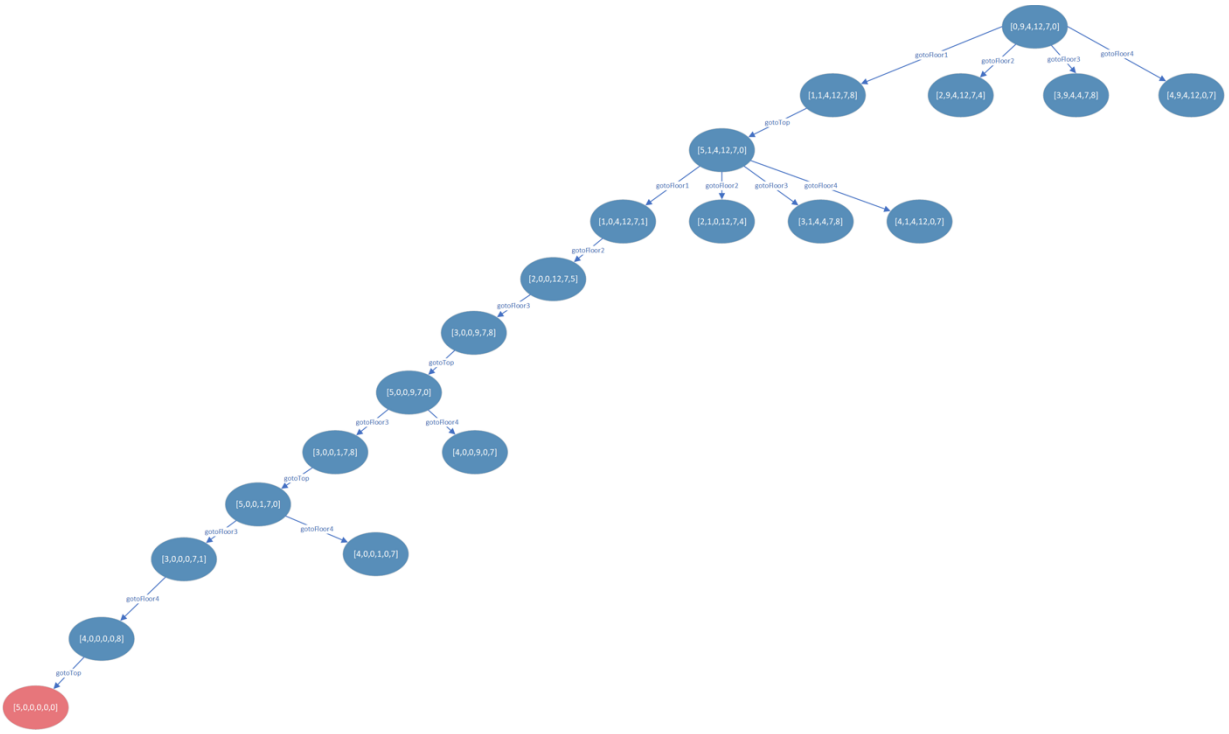
Ευριστικές μέθοδοι αναζήτησης:

Οι ευριστικές μέθοδοι αναζήτησης που υποστηρίζονται στο πρόβλημα είναι του Best-First-Search και του Hill Climbing. Και οι δύο ευριστικές μέθοδοι χρησιμοποιούν ως ευριστικό κριτήριο την κατάσταση με τον ελάχιστο αριθμό ατόμων στους ορόφους. Ο ευριστικός μηχανισμός χρησιμοποιεί την ευριστική συνάρτηση sum() που επιστρέφει ως ευριστική τιμή το άθροισμα των ατόμων που βρίσκονται στους ορόφους για την κάθε κατάσταση.

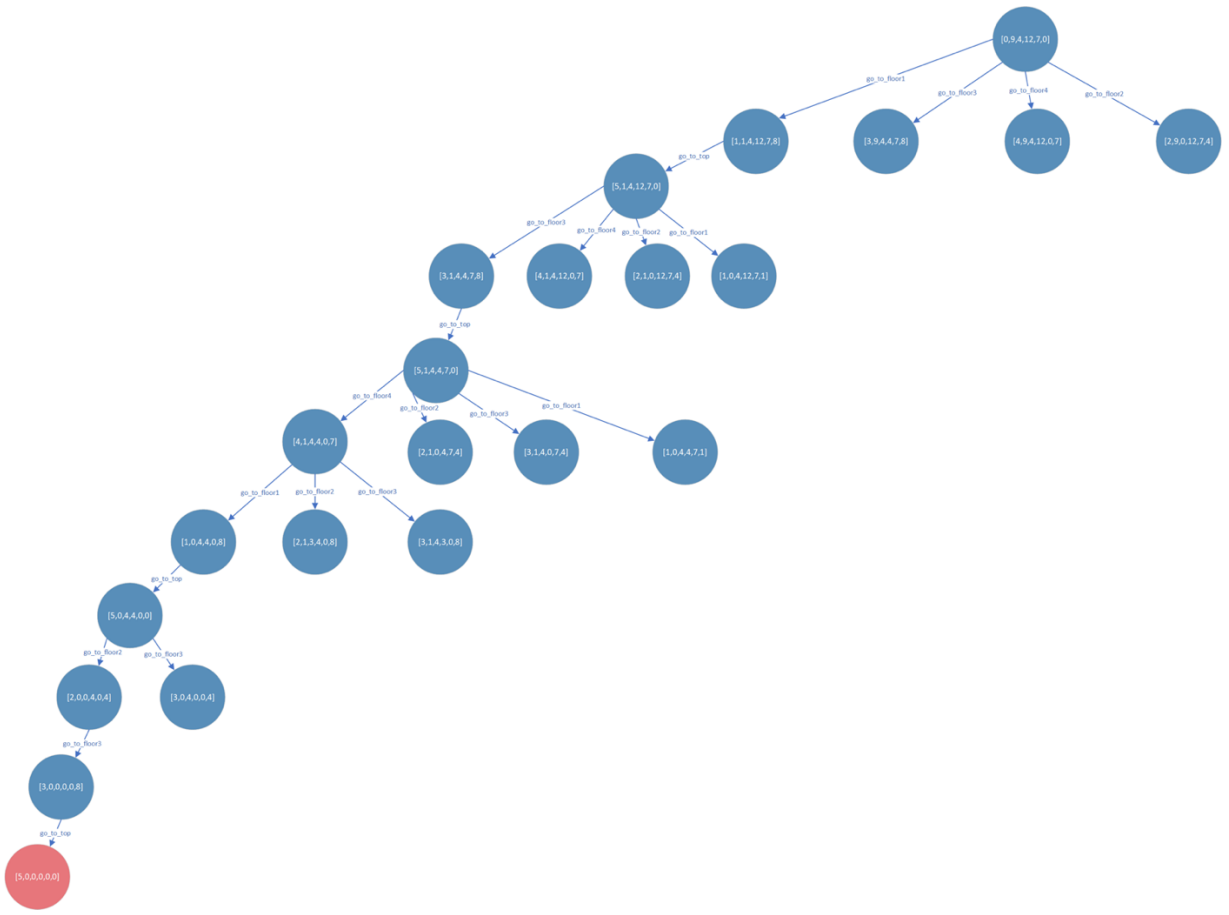
Ο Best-First-Search ταξινομεί τις καταστάσεις παιδιά βάζοντας πρώτα αυτή με το «χαμηλότερο» κόστος με βάση το ευριστικό κριτήριο. Πιο συγκεκριμένα στο πρόβλημα μας, εφαρμόζεται στην τρέχουσα κατάσταση (node) η συνάρτηση εύρεσης απογόνων (findchildren), προστίθενται τα παιδιά στο μέτωπο (front και queue) και μετά ταξινομείται κατά αύξουσα σειρά το μέτωπο σύμφωνα με την ευριστική τιμή που επιστρέφει η ευριστική συνάρτηση.

Ο Hill-Climbing παρομοίως ταξινομεί τις καταστάσεις παιδιά σύμφωνα με το «χαμηλότερο» κόστος αλλά αφαιρεί τις άλλες καταστάσεις παιδιά που δεν θεωρούνται καλές, άρα κρατάει στο μέτωπο μόνο την «καλύτερη» κατάσταση. Πιο συγκεκριμένα στο πρόβλημα μας, εφαρμόζεται στην τρέχουσα κατάσταση (node) η συνάρτηση εύρεσης απογόνων (findchildren), υπολογίζεται το κόστος κάθε παιδιού μέσω της ευριστικής τιμής, βρίσκεται το χαμηλότερο και προστίθεται αυτή στο μέτωπο (front και queue) ,αρα ουσιαστικά προσθέτουμε μόνο την κατάσταση με το «χαμηλότερο» κόστος και είναι σαν να αφαιρούμε τις υπόλοιπες.

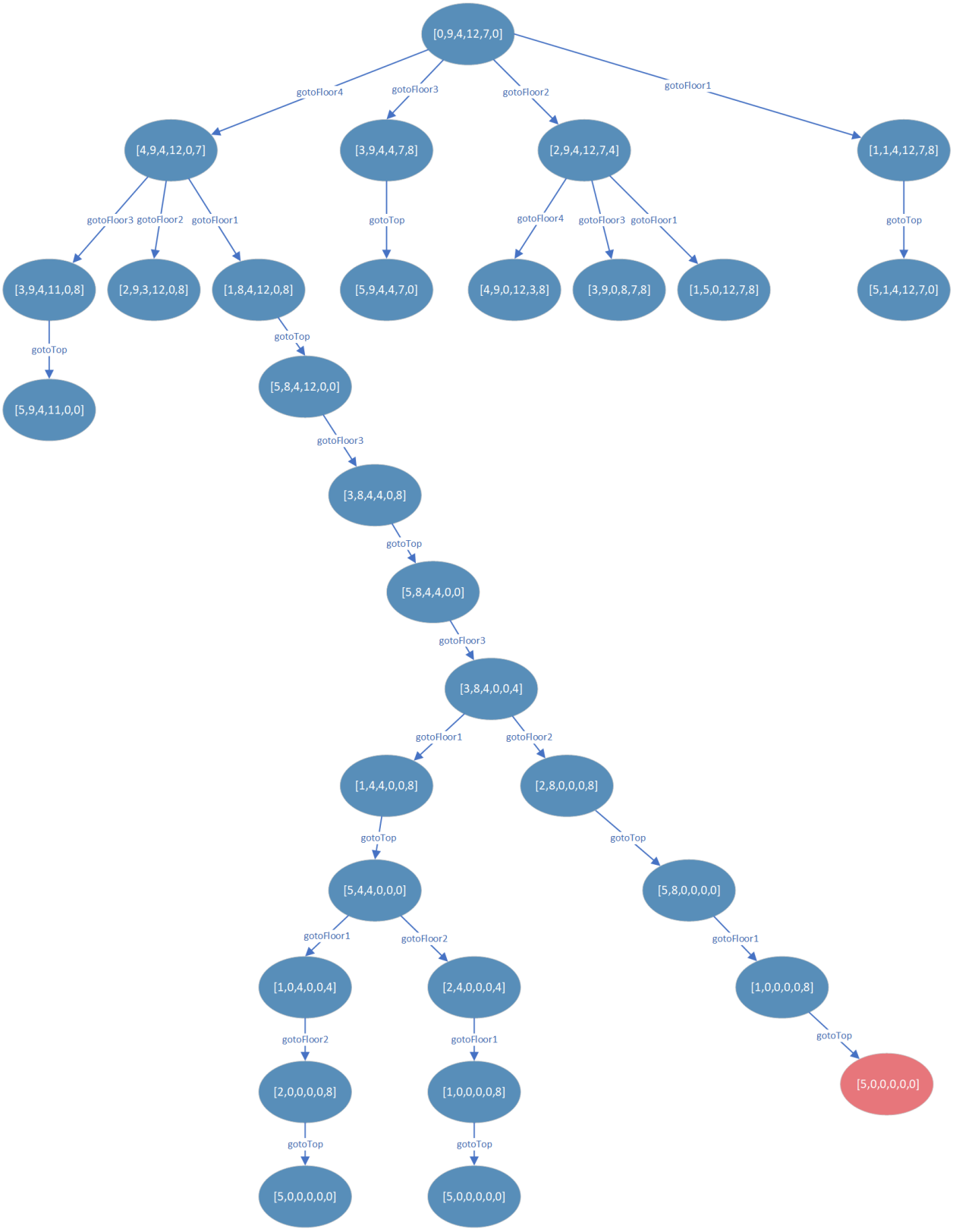
7. Για κάθε μέθοδο αναζήτησης να σχεδιαστεί το δένδρο αναζήτησης (όχι εξαντλητικό) για την απλή εκδοχή του προβλήματος που παρουσιάστηκε παραπάνω. Αν το δένδρο προχωρά σε πολύ μεγάλο βάθος περιορίστε το στα πρώτα 4 βήματα και στα 3 τελευταία.



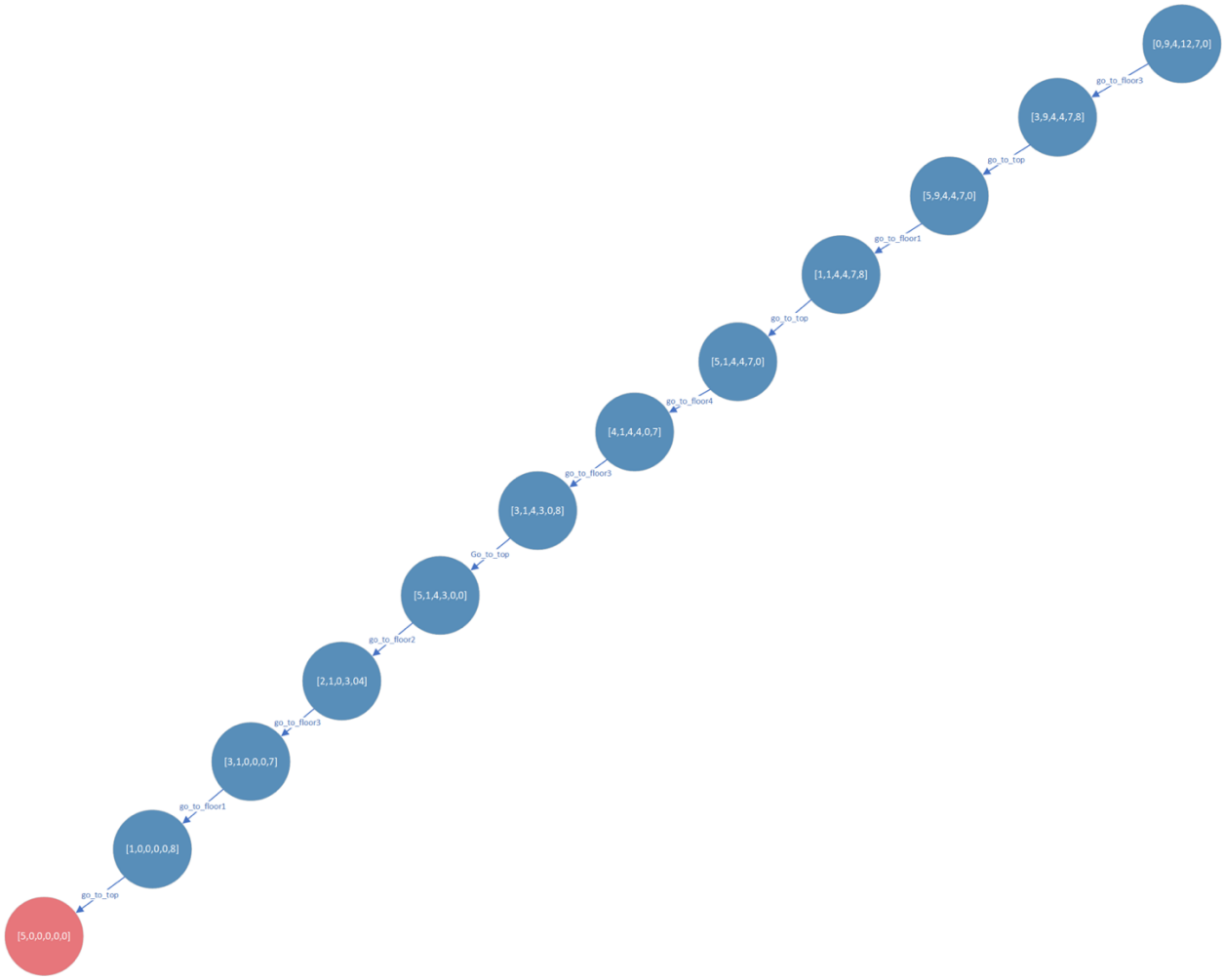
Εικόνα 1: Δέντρο Αναζήτησης αλγόριθμου Depth-First-Search



Εικόνα 2: Δέντρο Αναζήτησης αλγόριθμου Best-First-Search



Εικόνα 3: Δέντρο Αναζήτησης αλγόριθμου Breadth-First-Search



Εικόνα 4: Δέντρο Αναζήτησης αλγόριθμου Hill-Climbing

8. Να παρουσιαστούν και να σχολιαστούν οι περιπτώσεις εξαντλητικού ελέγχου που θα χρησιμοποιήσετε για κάθε μέθοδο αναζήτησης και τα συμπεράσματα που θα βγάλετε από τη συγκριτική μελέτη των αποτελεσμάτων των δοκιμών σας μεταξύ διαφορετικών μεθόδων αναζήτησης.

Σύγκριση αποτελεσμάτων με αρχική κατάσταση [0, 9, 4, 12, 7, 0]:

Ο DFS βρίσκει την λύση του προβλήματος εξερευνώντας το πρώτο παιδί της αρχικής κατάστασης επομένως το μέτωπο αναζήτησης δεν μεγαλώνει τόσο πολύ και ουσιαστικά δεν εγγυάται ότι η λύση που βρήκε είναι η βέλτιστη. Αυτό επιβεβαιώνεται στο πρόβλημα μας, γιατί βρίσκει το goal σε 12 βήματα ενώ ο BFS για παράδειγμα βρίσκει μονοπάτι με 11 βήματα, δηλαδή εκκενώνει τους ενοίκους με λιγότερα «δρομολόγια». Όμως, αν γενικά ένα πρόβλημα σαν το δικό μας έχει κλειστό σύνολο, δηλαδή ο χώρος καταστάσεων είναι πεπερασμένος τότε ο DFS σίγουρα θα βρει λύση.

Ο BFS επειδή εξερευνάει κάθε επίπεδο του δέντρου βρίσκει πάντα την μικρότερη σε μήκος μονοπάτι (καλύτερη λύση) αλλά το μέτωπο αναζήτησης του είναι πολύ μεγάλο και γενικά είναι ένας αργός αλγόριθμος. Αυτό επιβεβαιώνεται στο πρόβλημα μας, που συγκριτικά με τον DFS εκτελείται για περισσότερη ώρα και το front/queue έχει πιο πολλά παιδιά/μονοπάτια αλλά όπως αναφέρθηκε ο BFS έχει αποτελεσματικότερη λύση στο πρόβλημα.

Ο BestFS δίνει μια γρήγορη λύση στο πρόβλημα αλλά δεν σημαίνει ότι η λύση που έδωσε είναι και η βέλτιστη, γιατί εξαρτάται από τον ευριστικό μηχανισμό που χρησιμοποιήθηκε αν οδηγεί στην βέλτιστη λύση. Στο πρόβλημα μας, χρησιμοποιώντας ως ευριστικό μηχανισμό τον μικρότερο αριθμό ατόμων σε κάθε όροφο ο BestFS οδηγεί στην λύση σε 11 βήματα όπως ο BFS αλλά σε πιο γρήγορο χρόνο και το μέτωπο αναζήτησης του BestFS δεν μεγαλώνει συγκριτικά σε τόσο μεγάλο βαθμό.

Ο Hill Climbing γενικά χρησιμοποιείται σε προβλήματα που πρέπει να βρεθεί μια λύση πολύ γρήγορα και ας μην είναι η βέλτιστη, επειδή είναι πολύ αποδοτικός στον χρόνο και στην μνήμη. Στο πρόβλημα μας, ο Hill Climbing δεν βρίσκει τόση καλή λύση όσο ο BFS και ο BestFS αλλά βρίσκει μονοπάτι στο goal με 12 βήματα όπως ο DFS. Επιπλέον, σε σύγκριση με τους άλλους είναι πολύ πιο γρήγορος και δεν κρατάει τόση μνήμη στο front/queue για τα παιδιά/μονοπάτια εφόσον ακολουθεί μόνο το μονοπάτι που είναι «καλύτερο» σύμφωνα με τον ευριστικό μηχανισμό.

Σύγκριση αποτελεσμάτων με άλλες αρχικές καταστάσεις (εξαντλητικός έλεγχος):

1. Αρχική κατάσταση [2, 8, 4, 0, 0, 0]

   - Μονοπάτι DFS:

     ```
     [2, 8, 4, 0, 0, 0] [1, 0, 4, 0, 0, 8] [5, 0, 4, 0, 0, 0]
     [2, 0, 0, 0, 0, 4] [5, 0, 0, 0, 0, 0]
     ```

   - Μονοπάτι BFS:

     ```
     [2, 8, 4, 0, 0, 0] [1, 0, 4, 0, 0, 8] [5, 0, 4, 0, 0, 0]
     [2, 0, 0, 0, 0, 4] [5, 0, 0, 0, 0, 0]
     ```

   - Μονοπάτι BestFS:

     ```
     [2, 8, 4, 0, 0, 0] [1, 0, 4, 0, 0, 8] [5, 0, 4, 0, 0, 0]
     [2, 0, 0, 0, 0, 4] [5, 0, 0, 0, 0, 0]
     ```

   - Μονοπάτι Hill Climbing:

     ```
     [2, 8, 4, 0, 0, 0] [1, 0, 4, 0, 0, 8] [5, 0, 4, 0, 0, 0]
     [2, 0, 0, 0, 0, 4] [5, 0, 0, 0, 0, 0]
     ```

   Βλέπουμε ότι το μονοπάτι όλων των αλγορίθμων είναι μικρότερο και ίδιο σε όλους τους αλγορίθμους γιατί η κατάσταση αυτή είναι κοντά στον στόχο και οδηγείται με ένα μονοπάτι εκεί.

2. Αρχική κατάσταση [0,15,22,3,24,0]

   - Μονοπάτι DFS:

     ```
     [0, 15, 22, 3, 24, 0] [1, 7, 22, 3, 24, 8] [5, 7, 22, 3, 24, 0]
     [1, 0, 22, 3, 24, 7]  [2, 0, 21, 3, 24, 8] [5, 0, 21, 3, 24, 0]
     [2, 0, 13, 3, 24, 8]  [5, 0, 13, 3, 24, 0] [2, 0, 5, 3, 24, 8]
     [5, 0, 5, 3, 24, 0]   [2, 0, 0, 3, 24, 5]  [3, 0, 0, 0, 24, 8]
     [5, 0, 0, 0, 24, 0]   [4, 0, 0, 0, 16, 8]  [5, 0, 0, 0, 16, 0]
     [4, 0, 0, 0, 8, 8]    [5, 0, 0, 0, 8, 0]   [4, 0, 0, 0, 0, 8]
     [5, 0, 0, 0, 0, 0]
     ```

   - Μονοπάτι BFS:

     ```
     [0, 15, 22, 3, 24, 0] [4, 15, 22, 3, 16, 8] [5, 15, 22, 3, 16, 0]
     [4, 15, 22, 3, 8, 8]  [5, 15, 22, 3, 8, 0]  [4, 15, 22, 3, 0, 8]
     [5, 15, 22, 3, 0, 0]  [3, 15, 22, 0, 0, 3]  [2, 15, 17, 0, 0, 8]
     [5, 15, 17, 0, 0, 0]  [2, 15, 9, 0, 0, 8]   [5, 15, 9, 0, 0, 0]
     [2, 15, 1, 0, 0, 8]   [5, 15, 1, 0, 0, 0]   [2, 15, 0, 0, 0, 1]
     [1, 8, 0, 0, 0, 8]    [5, 8, 0, 0, 0, 0]    [1, 0, 0, 0, 0, 8]
     [5, 0, 0, 0, 0, 0]
     ```

- Μονοπάτι BestFS:

```
[0, 15, 22, 3, 24, 0] [1, 7, 22, 3, 24, 8] [5, 7, 22, 3, 24, 0]
[2, 7, 14, 3, 24, 8]  [5, 7, 14, 3, 24, 0] [2, 7, 6, 3, 24, 8]
[5, 7, 6, 3, 24, 0]   [4, 7, 6, 3, 16, 8]  [5, 7, 6, 3, 16, 0]
[4, 7, 6, 3, 8, 8]    [5, 7, 6, 3, 8, 0]   [4, 7, 6, 3, 0, 8]
[5, 7, 6, 3, 0, 0]    [1, 0, 6, 3, 0, 7]   [2, 0, 5, 3, 0, 8]
[5, 0, 5, 3, 0, 0]    [2, 0, 0, 3, 0, 5]   [3, 0, 0, 0, 0, 8]
[5, 0, 0, 0, 0, 0]
```

- Μονοπάτι Hill Climbing:

```
[0, 15, 22, 3, 24, 0] [4, 15, 22, 3, 16, 8] [5, 15, 22, 3, 16, 0]
[4, 15, 22, 3, 8, 8]  [5, 15, 22, 3, 8, 0]  [4, 15, 22, 3, 0, 8]
[5, 15, 22, 3, 0, 0]  [2, 15, 14, 3, 0, 8]  [5, 15, 14, 3, 0, 0]
[2, 15, 6, 3, 0, 8]   [5, 15, 6, 3, 0, 0]   [1, 7, 6, 3, 0, 8]
[5, 7, 6, 3, 0, 0]    [1, 0, 6, 3, 0, 7]    [3, 0, 6, 2, 0, 8]
[5, 0, 6, 2, 0, 0]    [2, 0, 0, 2, 0, 6]    [3, 0, 0, 0, 0, 8]
[5, 0, 0, 0, 0, 0]
```

Ως αρχική κατάσταση δίνουμε μια μη εφικτή στον χώρο καταστάσεων (πιο πολλοί ένοικοι στους ορόφους) αλλά οι αλγόριθμοι δίνουν λύση γιατί δεν το έχουμε θέσει ως προϋπόθεση στο πρόβλημα μας. Βλέπουμε, δηλαδή ότι όλοι οι αλγόριθμοι φτάνουν με τον ίδιο αριθμό βημάτων στον στόχο αλλά με άλλα μονοπάτια και προφανώς επειδή είναι περισσότεροι οι ένοικοι τα μονοπάτια είναι μεγαλύτερα.

3. Αρχική κατάσταση [3, 10, 27, 13, 8, 4]

- Μονοπάτι DFS:

```
[3, 10, 27, 13, 8, 4] [1, 6, 27, 13, 8, 8] [5, 6, 27, 13, 8, 0]
[1, 0, 27, 13, 8, 6]  [2, 0, 25, 13, 8, 8] [5, 0, 25, 13, 8, 0]
[2, 0, 17, 13, 8, 8]  [5, 0, 17, 13, 8, 0] [2, 0, 9, 13, 8, 8]
[5, 0, 9, 13, 8, 0]   [2, 0, 1, 13, 8, 8]  [5, 0, 1, 13, 8, 0]
[2, 0, 0, 13, 8, 1]   [3, 0, 0, 6, 8, 8]   [5, 0, 0, 6, 8, 0]
[3, 0, 0, 0, 8, 6]    [4, 0, 0, 0, 6, 8]   [5, 0, 0, 0, 6, 0]
[4, 0, 0, 0, 0, 6]    [5, 0, 0, 0, 0, 0]
```

- Μονοπάτι BFS:

  Δεν υπάρχει μονοπάτι, λογικά είναι τόσο μεγάλο το δέντρο του προβλήματος που λόγω του recursion limit δεν δημιουργείται το μονοπάτι στο μέτωπο.

- Μονοπάτι BestFS:

```
[3, 10, 27, 13, 8, 4] [1, 6, 27, 13, 8, 8] [5, 6, 27, 13, 8, 0]
[2, 6, 19, 13, 8, 8]  [5, 6, 19, 13, 8, 0] [2, 6, 11, 13, 8, 8]
[5, 6, 11, 13, 8, 0]  [2, 6, 3, 13, 8, 8]  [5, 6, 3, 13, 8, 0]
[3, 6, 3, 5, 8, 8]    [5, 6, 3, 5, 8, 0]   [4, 6, 3, 5, 0, 8]
[5, 6, 3, 5, 0, 0]    [1, 0, 3, 5, 0, 6]   [2, 0, 1, 5, 0, 8]
[5, 0, 1, 5, 0, 0]    [3, 0, 1, 0, 0, 5]   [2, 0, 0, 0, 0, 6]
[5, 0, 0, 0, 0, 0]
```

- Μονοπάτι Hill Climbing:

```
[3, 10, 27, 13, 8, 4] [4, 10, 27, 13, 4, 8] [5, 10, 27, 13, 4, 0]
[3, 10, 27, 5, 4, 8]  [5, 10, 27, 5, 4, 0]  [2, 10, 19, 5, 4, 8]
[5, 10, 19, 5, 4, 0]  [2, 10, 11, 5, 4, 8]  [5, 10, 11, 5, 4, 0]
[2, 10, 3, 5, 4, 8]   [5, 10, 3, 5, 4, 0]   [1, 2, 3, 5, 4, 8]
[5, 2, 3, 5, 4, 0]    [3, 2, 3, 0, 4, 5]    [4, 2, 3, 0, 1, 8]
[5, 2, 3, 0, 1, 0]    [2, 2, 0, 0, 1, 3]    [1, 0, 0, 0, 1, 5]
[4, 0, 0, 0, 0, 6]    [5, 0, 0, 0, 0, 0]
```

Ομοίως, έχουμε μια μη εφικτή κατάσταση στον χώρο καταστάσεων αλλά ο αριθμός των ενοίκων είναι πιο μεγάλος και για αυτό τα μονοπάτια είναι πιο μεγάλα. Επιπλέον, στον BFS δεν δίνεται λύση γιατί επεκτείνεται το μέτωπο αναζήτησης επίπεδο ανά επίπεδο και για αυτό μεγαλώνει σε πιο υπερβολικό βαθμό σε σύγκριση με τους υπόλοιπους.

4. Αρχική κατάσταση [4, 1, 4, 12, 0, 7]

- Μονοπάτι DFS:

```
[4, 1, 4, 12, 0, 7] [1, 0, 4, 12, 0, 8] [5, 0, 4, 12, 0, 0]
[2, 0, 0, 12, 0, 4] [3, 0, 0, 8, 0, 8]  [5, 0, 0, 8, 0, 0]
[3, 0, 0, 0, 0, 8]  [5, 0, 0, 0, 0, 0]
```

- Μονοπάτι BFS:

```
[4, 1, 4, 12, 0, 7] [1, 0, 4, 12, 0, 8] [5, 0, 4, 12, 0, 0]
[3, 0, 4, 4, 0, 8]  [5, 0, 4, 4, 0, 0]  [3, 0, 4, 0, 0, 4]
[2, 0, 0, 0, 0, 8]  [5, 0, 0, 0, 0, 0]
```

- Μονοπάτι BestFS:

```
[4, 1, 4, 12, 0, 7] [1, 0, 4, 12, 0, 8] [5, 0, 4, 12, 0, 0]
[3, 0, 4, 4, 0, 8]  [5, 0, 4, 4, 0, 0]  [2, 0, 0, 4, 0, 4]
[3, 0, 0, 0, 0, 8]  [5, 0, 0, 0, 0, 0]
```

- Μονοπάτι Hill Climbing:

```
[4, 1, 4, 12, 0, 7] [3, 1, 4, 11, 0, 8] [5, 1, 4, 11, 0, 0]
[3, 1, 4, 3, 0, 8]  [5, 1, 4, 3, 0, 0]  [2, 1, 0, 3, 0, 4]
[3, 1, 0, 0, 0, 7]  [1, 0, 0, 0, 0, 8]  [5, 0, 0, 0, 0, 0]
```

Τέλος, βλέπουμε ότι τα μονοπάτια είναι και εδώ σχετικά μικρότερα, γιατί η αρχική κατάσταση βρίσκεται αρκετά κοντά στον στόχο, αλλά οδηγείται με διαφορετικά μονοπάτια ανάλογα με τον αλγόριθμο αναζήτησης. Μόνο ο Hill Climbing οδηγείται με περισσότερα βήματα στον στόχο ενώ όλοι οι άλλοι με τον ίδιο αριθμό βημάτων.