ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ & ΠΛΗΡΟΦΟΡΙΚΗΣ

ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ
ΑΠΑΛΛΑΚΤΙΚΗ ΕΡΓΑΣΙΑ ΕΞΑΜΗΝΟΥ

Βοϊδονικόλα Μαρία ΑΜ: 19*****7
Γιαλαμάς Αθανάσιος ΑΜ: 71*****0
Αθήνα, 10/01/2024
Ακαδημαϊκό έτος 2023-2024

Περιεχόμενα

Ο κόσμος του προβλήματος

Ο κόσμος του προβλήματος, με βάση το τυπικό ορισμό, αποτελείται μόνο από τα αντικείμενα που υπάρχουν σε αυτόν, τις ιδιότητες των αντικειμένων και τις σχέσεις που τα συνδέουν. Κατά συνέπεια, στο πρόβλημα που μας παρουσιάστηκε, τα αντικείμενα είναι το ασανσέρ και οι όροφοι της πολυκατοικίας, ενώ οι ιδιότητες των ορόφων είναι ο αριθμός του έκαστου και οι εναπομείναντες ένοικοι ενώ για το ασανσέρ είναι ο όροφος στον οποίο βρίσκεται, οι διαθέσιμες θέσεις και οι ένοικοι που βρίσκονται σε αυτό. Τέλος, για τις σχέσεις που συνδέουν τα αντικείμενα μεταξύ τους, θεωρούμε πως το ασανσέρ βρίσκεται σε κάποιον όροφο σε κάθε κατάσταση και επίσης μπορεί να μεταφέρεται μεταξύ ορόφων και να μεταβάλλει τον αριθμό των εναπομεινάντων ενοίκων του ορόφου στον οποίο σταθμεύει.

Αρχική κατάσταση

Το ασανσέρ βρίσκεται άδειο στο ισόγειο. Ο πρώτος όροφος έχει όλους τους ενοίκους του, ο αριθμός των οποίων είναι 9. Το ίδιο συμβαίνει και στους υπόλοιπους αντίστοιχα, με τον δεύτερο να έχει 4 ενοίκους, τον τρίτο 12 και τέλος τον τέταρτο με 7 ενοίκους.

Με την κωδικοποίηση σε λίστα που θα χρησιμοποιηθεί στην παρούσα επίλυση του προβλήματος, η οποία και θα εξηγηθεί περαιτέρω στη συνέχεια, η αρχική κατάσταση μπορεί να περιγραφεί ως εξής:

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

Τελική κατάσταση

Όλοι οι όροφοι του κτιρίου είναι άδειοι. Το ασανσέρ βρίσκεται άδειο στη ταράτσα του κτιρίου.

Θεωρώντας τη ταράτσα ως τον 5ο όροφο του κτιρίου, η λίστα που θα περιγράφει την τελική κατάσταση θα έχει τη μορφή:

[5, 0, 0, 0, 0, 0]

Τελεστές μετάβασης

Ακολουθεί θεωρητική περιγραφή των τελεστών μετάβασης του προβλήματος. Για διευκόλυνση και μεγαλύτερη σαφήνεια στην παρακάτω αναπαράσταση, έχουν δοθεί επιπλέον ονομασίες που δεν έχουν χρησιμοποιηθεί στον κώδικα.

Για state[x] τον αριθμό των εναπομεινάντων ενοίκων σε έναν όροφο x και elev τον αριθμό των επιβατών του ασανσέρ στην τρέχουσα κατάσταση του προβλήματος, οι τελεστές μετάβασης του ορίζονται ως εξής:

Όνομα τελεστή: go_to_floor1
Ενέργεια: Μεταφορά του ασανσέρ με τους επιβαίνοντες του στον πρώτο όροφο
Προϋποθέσεις 1)Το ασανσέρ θα πρέπει να μην είναι γεμάτο. (0≤elev<8) 2)Να υπάρχει τουλάχιστον ένας ένοικος στον πρώτο όροφο. (state[1] > 0) 3)Το ασανσέρ να βρίσκεται σε άλλο όροφο. (x≠1)
Αποτελέσματα [1, state[1]', state[2], state[3], state[4], elev'] όπου: state[1]' = state[1]– cap , με cap = 8 – elev την χωρητικότητα του ασανσέρ και elev' = elev + state[1]– state[1]'

Όνομα τελεστή: go_to_floor2
Ενέργεια: Μεταφορά του ασανσέρ με τους επιβαίνοντες του στον δεύτερο όροφο
Προϋποθέσεις 1)Το ασανσέρ θα πρέπει να μην είναι γεμάτο. (0≤elev<8) 2)Να υπάρχει τουλάχιστον ένας ένοικος στο δεύτερο όροφο. (state[2]> 0) 3)Το ασανσέρ να βρίσκεται σε άλλο όροφο. (x≠2)
Αποτελέσματα [2, state[1], state[2]', state[3], state[4], elev'] όπου: state[2]' = state[2] – cap , με cap = 8 – elev την χωρητικότητα του ασανσέρ και elev' = elev + state[2] – state[2]'

Όνομα τελεστή: go_to_floor3
Ενέργεια: Μεταφορά του ασανσέρ με τους επιβαίνοντες του στο τρίτο όροφο
Προϋποθέσεις 1)Το ασανσέρ θα πρέπει να μην είναι γεμάτο. (0≤elev<8) 2)Να υπάρχει τουλάχιστον ένας ένοικος στο τρίτο όροφο. (state[3]> 0) 3)Το ασανσέρ να βρίσκεται σε άλλο όροφο. (x≠3)
Αποτελέσματα [3, state[1], state[2], state[3]', state[4], elev'] όπου: state[3]' = state[3] – cap , με cap = 8 – elev την χωρητικότητα του ασανσέρ και elev' = elev + state[3] – state[3]'

Όνομα τελεστή: go_to_floor4
Ενέργεια: Μεταφορά του ασανσέρ με τους επιβαίνοντες του στο τέταρτο όροφο
Προϋποθέσεις 1)Το ασανσέρ θα πρέπει να μην είναι γεμάτο. (0≤elev<8) 2)Να υπάρχει τουλάχιστον ένας ένοικος στο τέταρτο όροφο. (state[4]> 0) 3)Το ασανσέρ να βρίσκεται σε άλλο όροφο. (x≠4)
Αποτελέσματα [4, state[1], state[2], state[3], state[4]', elev'] όπου: state[4]' = state[4] – cap , με cap = 8 – elev την χωρητικότητα του ασανσέρ και elev' = elev + state[4] – state[4]'

Όνομα τελεστή: go_to_roof
Ενέργεια: Μεταφορά του ασανσέρ με τους επιβαίνοντες του στη ταράτσα
Προϋποθέσεις
Οι όροφοι να είναι άδειοι(state[1]=state[2]=state[3]=state[4]=0) ή το ασανσέρ γεμάτο (elev = 8)
Αποτελέσματα
[5, state[1], state[2], state[3], state[4], 0]

Χώρος καταστάσεων του προβλήματος

Με τον όρο χώρο των καταστάσεων εννοούμε το σύνολο εκείνο που περιέχει όλες τις δυνατές και αποδεκτές καταστάσεις.
Στην περίπτωση του προβλήματός μας, αποδεκτές θα είναι οι καταστάσεις που βγάζουν νόημα για το χώρο του προβλήματος. Θα πρέπει δηλαδή να τηρούνται πάντα οι προϋποθέσεις:

- ο όροφος του κτιρίου στον οποίο βρίσκεται ο ανελκυστήρας να έχει τιμές 0-5, όπου 0 είναι το ισόγειο και 5 είναι η ταράτσα
- Ο αριθμός των ενοίκων ανά όροφο να είναι πάντα μη αρνητικός και να μην ξεπερνά τον αρχικό αριθμό των ενοίκων του
- Το ασανσέρ να έχει πάντα από 0 έως 8 επιβαίνοντες

Με τη μορφή λίστας, η κάθε κατάσταση που περιέχεται στον χώρο καταστάσεων, περιγράφεται ως εξής:

[x, state[1], state[2], state[3], state[4], elev]

με $x \in [0,5]$, $state[1] \in [0,9]$, $state[2] \in [0,4]$, $state[3] \in [0,12]$, $state[4] \in [0,7]$ και $elev \in [0,8]$
Αντιπροσωπευτικά παραδείγματα του χώρου καταστάσεων σε μορφή λίστας με την κωδικοποίηση που επιλέχθηκε στην παρούσα επίλυση, είναι τα εξής:

- [0, 9, 4, 12, 7, 0], η αρχική κατάσταση
- [3, 2, 5, 8, 0, 3], ο ανελκυστήρας με 3 επιβάτες βρίσκεται στον 3ο όροφο
- [2, 0, 8, 0, 1, 0], ο ανελκυστήρας βρίσκεται άδειος στον 2ο όροφο

Ενώ μερικά χαρακτηριστικά παραδείγματα άκυρων και μη αποδεκτών καταστάσεων, που δεν θα ανήκουν στον χώρο καταστάσεων είναι:

- [7, 0, 0, 0, 0, 0], ο ανελκυστήρας βρίσκεται σε όροφο εκτός του κτιρίου

- [4, 1, 3, 5, 7, 10], ο ανελκυστήρας έχει παραπάνω άτομα από την χωρητικότητά του

Κωδικοποίηση του κόσμου του προβλήματος

Για την αναπαράσταση του κόσμου του προβλήματος, χρησιμοποιήθηκε μια λίστα 6 στοιχείων με όνομα state που περιγράφει την εκάστοτε κατάσταση του προβλήματος σε κάθε στιγμή. Η κωδικοποίηση της έγινε ως εξής:

[όροφος στον οποίο βρίσκεται το ασανσέρ, ένοικοι 1ου ορόφου, ένοικοι 2ου, ένοικοι 3ου, ένοικοι 4ου, αριθμός ατόμων στο ασανσέρ]

Κωδικοποίηση

Τελεστών μετάβασης του προβλήματος

Όλες οι συναρτήσεις που υλοποιούν τη μετακίνηση του ασανσέρ μεταξύ των ορόφων λαμβάνουν υπόψη τη κατάσταση του ανελκυστήρα και τους ενοίκους στους ορόφους.
Η συνάρτηση “go_to_roof” δέχεται μια κατάσταση *state,* η οποία αναπαριστά τη τρέχουσα κατάσταση του συστήματος με τους ενοίκους σε κάθε όροφο και τον αριθμό των ατόμων στο ασανσέρ. Με χρήση της ενσωματωμένης συνάρτησης *all* ελέγχει αν όλοι οι όροφοι είναι άδειοι, δηλαδή αν οι τιμές των στοιχείων της λίστας για τους όρους 1 έως 4 είναι μηδέν.
Στη συνέχεια, η συνάρτηση ελέγχει δύο περιπτώσεις. Αν ο ανελκυστήρας είναι γεμάτος (η τελευταία θέση της *state* είναι 8) ή όλοι οι όροφοι είναι άδειοι. Τότε επιστρέφεται μια νέα κατάσταση που αντιπροσωπεύει το ασανσέρ στη ταράτσα, με τους ενοίκους από τους υπόλοιπους ορόφους και τον αριθμό των ατόμων στον ανελκυστήρα να είναι μηδέν.
Στην αρχή, η συνάρτηση “go_to_floor1” ελέγχει αν ο ανελκυστήρας είναι γεμάτος (το τελευταίο στοιχείο της λίστας *state* είναι μικρότερο του 8) και αν υπάρχει τουλάχιστον ένα άτομο (το δεύτερο στοιχείο της *state[1]* είναι μεγαλύτερο από το 0). Αν και οι δύο συνθήκες είναι αληθείς, η συνάρτηση προχωρά σε δεύτερη σειρά ελέγχων. Αν το πλήθος των ενοίκων στο πρώτο όροφο (*state[1]*) είναι μεγαλύτερο από τη διαφορά του 8 με τον αριθμό των ατόμων που υπάρχουν ήδη στον ασανσέρ (*state[-1]*), τότε το νέο της κατάστασης *new_state* περιλαμβάνει έναν ανελκυστήρα με 8 άτομα και τους υπόλοιπους ενοίκους που υπήρχαν στο 1ο όροφο. Αν ο πρώτος έλεγχος δεν ικανοποιείται, ο κώδικας ελέγχει το δεύτερο. Σε αυτή τη περίπτωση, το *new_state* περιλαμβάνει τους ενοίκους του 1ου ορόφου.
Η συνάρτηση “go_to_floor2” υλοποιεί τη μετακίνηση του ανελκυστήρα στο δεύτερο όροφο. Ελέγχονται δύο συνθήκες για να καθοριστεί αν η μετακίνηση στον όροφο είναι δυνατή. Πρώτα ελέγχει αν το ασανσέρ δεν είναι γεμάτο (η τελευταία θέση της *state* είναι

μικρότερη του 8) και έπειτα αν ο αριθμός των ενοίκων στο δεύτερο όροφο είναι μεγαλύτερος του μηδενός.
Αν οι δύο αυτές συνθήκες ισχύουν, τότε υπολογίζεται η νέα κατάσταση *new_state* ανάλογα με τον αριθμό των ανθρώπων που πρόκειται να μετακινηθούν. Αν ο αριθμός αυτός είναι μεγαλύτερος από τη διαθέσιμη χωρητικότητα του ανελκυστήρα (*8-state[1]*), τότε μεταφέρει τους ενοίκους στο δεύτερο όροφο, και θέτει τον αριθμό των ατόμων στο ασανσέρ στο μέγιστο (8). Διαφορετικά, θέτει τον αριθμό των ενοίκων στο δεύτερο όροφο και στο ασανσέρ σε μηδέν. Τέλος, επιστρέφεται η νέα κατάσταση *new_state.*
Η συνάρτηση "go_to_floor3", που μετακινεί το ασανσέρ στο τρίτο όροφο, ελέγχει δύο προϋποθέσεις. Αρχικά, ελέγχει αν ο ανελκυστήρας δεν είναι γεμάτος (η τελευταία θέση της *state* είναι μικρότερη από 8) και αν ο τρίτος όροφος έχει τουλάχιστον έναν ενοίκο (η τρίτη θέση της *state* είναι μεγαλύτερη από 0).
Στη συνέχεια, ελέγχει δύο περιπτώσεις. Αν οι ένοικοι στον τρίτο όροφο είναι περισσότεροι από όσους μπορεί να χωρέσει ο ανελκυστήρας (δηλαδή ο τρίτος όροφος έχει περισσότερους από 8 ενοίκους μαζί με τους ενοίκους που είναι ήδη μέσα στον ανελκυστήρα), τότε η συνάρτηση δημιουργεί μια νέα κατάσταση *new_state* όπου μετακινούνται οι περισσότεροι ένοικοι από τον τρίτο όροφο στον ανελκυστήρα, υπολογίζοντας πόσοι ακριβώς μπορούν να χωρέσουν.
Στη δεύτερη περίπτωση, αν οι ένοικοι στον τρίτο όροφο δεν ξεπερνούν τη χωρητικότητα του ασανσέρ, η συνάρτηση μετακινεί όλους τους ενοίκους από τον τρίτο όροφο στον ανελκυστήρα, αφήνοντας τον τρίτο όροφο άδειο. Τέλος, η συνάρτηση επιστρέφει τη νέα κατάσταση *new_state*.
Και τέλος, η συνάρτηση "go_to_floor4" μετακινεί τον ανελκυστήρα στο τέταρτο όροφο. Στο πρώτο έλεγχο που διενεργεί η συνάρτηση ελέγχει αν η τελευταία θέση (*state[-1]*) της *state* είναι μικρότερη του 8, δηλαδή το ασανσέρ δεν είναι γεμάτο, και αν υπάρχει έστω και ένας ένοικος στο τέταρτο όροφο (*state[4]>0*).
Αν οι δύο αυτές συνθήκες ικανοποιούνται, ελέγχεται αν οι ένοικοι στο τέταρτο όροφο υπερβαίνουν τη χωρητικότητα του ανελκυστήρα οπότε, δημιουργείται η νέα κατάσταση *new_state* όπου μετακινούνται οι ένοικοι από το τέταρτο στον ανελκυστήρα βάση της επιτρεπόμενης χωρητικότητας. Σε περίπτωση όμως που ένοικοι του ορόφου δεν υπερβαίνουν τη χωρητικότητα του ασανσέρ, η συνθήκη εκτελείται αλλά όλοι οι ένοικοι μετακινούνται στον ανελκυστήρα και ο όροφος αδειάζει. Και τελικά, επιστρέφεται η *new_state.*

Συνάρτηση εύρεσης απογόνων (findchildren)

Η συνάρτηση find_children είναι υπεύθυνη για τη δημιουργία και επιστροφή μιας λίστας που περιέχει τους διάφορους δυνατούς απογόνους (παιδιά) μιας δεδομένης κατάστασης στο πρόβλημα. Κάθε παιδί αντιστοιχεί σε μια πιθανή επόμενη κατάσταση μετά από μια επιτυχημένη μετάβαση. Αρχικά, δημιουργούνται αντίγραφα της τρέχουσας κατάστασης (`state`) για κάθε δυνατό παιδί με τη μέθοδο deepcopy (πχ

`roof_state=copy.deepcopy(state)`, `floor1_state=copy.deepcopy(state)` κλπ). Για κάθε δυνατό παιδί, καλούνται οι συγκεκριμένες συναρτήσεις (*go_to_roof*, *go_to_floor1*, κλπ) που περιγράφουν τις επιτρεπτές μεταβάσεις.
Τα παιδιά που δεν επιστρέφουν *None*, που σημαίνει ότι είναι μια έγκυρη μετάβαση, προστίθενται στη λίστα *children*. Τέλος, η λίστα *children* επιστρέφεται ως αποτέλεσμα της συνάρτησης. Η συνάρτηση αυτή παίζει κρίσιμο ρόλο στην υλοποίηση του αλγορίθμου αναζήτησης, καθώς παρέχει τις δυνατότητες κίνησης (μεταβάσεις) από μια κατάσταση σε μια άλλη.

Περιγραφή μεθόδων αναζήτησης

Αναζήτηση κατά βάθος (Depth First Search - DFS)

Ο αλγόριθμος Depth First Search (DFS) εξερευνά συνεχώς τις βαθύτερες πιθανές καταστάσεις προτού επιστρέψει να εξερευνήσει άλλες πιθανότητες και αυτό δημιουργεί μια βαθιά αναζήτηση στο χώρο καταστάσεων. Στο συγκεκριμένο πρόβλημα, λειτούργησε ξεκινώντας από μια αρχική κατάσταση (initial_state: [0, 9, 4, 12, 7, 0]) που περιγράφει την αρχική τοποθεσία του ανελκυστήρα και την κατανομή των ενοίκων στους ορόφους της πολυκατοικίας. Ο στόχος (goal) ήταν να μεταφερθούν όλοι ένοικοι από τους ορόφους στη ταράτσα και περιγράφεται από την τελική κατάσταση ([5, 0, 0, 0, 0, 0]). Ο αλγόριθμος χρησιμοποιήσε μια ουρά (queue) για τη διαχείριση των μονοπατιών που δημιουργούνται από κόμβο σε κόμβο κατά τη διάρκεια της διαδικασίας της αναζήτησης, όπου αρχικά προστέθηκε η αρχική κατάσταση σε αυτή και έπειτα ακολουθεί το μονοπάτι που διασχίζει ο αλγόριθμος. Αποτελεί σημαντικό στοιχείο για την λειτουργία της μεθόδου DFS, αφού καταγράφεται το τρέχον κλαδί και δίνει τη δυνατότητα της επιστροφής προς τα πίσω σε περίπτωση που το τερματίσει χωρίς αποτέλεσμα και χρειαστεί να μεταβεί σε άλλο κλαδί.
Με την επέκταση του μετώπου (front) γίνεται εξέταση του πρώτου στοιχείου της ουράς. Για κάθε “παιδί” (ή πιθανή μετάβαση) της τρέχουσας κατάστασης, προστίθεται στη κορυφή της ουράς. Επίσης, χρησιμοποιήθηκε ένας έλεγχος επανάληψης ώστε να αποφευχθούν επαναλήψεις καταστάσεων κατά την εκτέλεση του αλγορίθμου. Τέλος, ο αλγόριθμος εξέτασε τις συνθήκες τερματισμού, δηλαδή είτε να έχει επιτευχθεί ο στόχος είτε η ουρά να είναι άδεια.

Αναζήτηση κατά πλάτος (Breadth First Search - BFS)

Ο αλγόριθμος Breadth First Search (BFS) λειτουργεί με την αναζήτηση κατά πλάτος του χώρου καταστάσεων και παρέχει εγγυημένα το βέλτιστο μονοπάτι στην εύρεση λύσης, καθώς εξερευνά όλες τις καταστάσεις σε ένα επίπεδο, πριν προχωρήσει στο επόμενο. Για το πρόβλημα του ανελκυστήρα, ξεκίνησε μια αρχική κατάσταση (initial_state) όπου περιγράφεται η αρχική τοποθεσία του ασανσέρ και ο αριθμός των ενοίκων σε κάθε

όροφο. Στόχος (goal), και για αυτό τον αλγόριθμο, είναι η μετακίνηση όλων των ενοίκων από τους ορόφους στη ταράτσα και περιγράφεται από την τελική κατάσταση ([5, 0, 0, 0, 0, 0]). Εδώ η ουρά (queue) χρησιμοποιείται για τη διαχείριση του μετώπου και την εξερεύνηση των καταστάσεων σε ένα επίπεδο πριν συνεχίσει στο επόμενο. Εισάγεται σε αυτή η αρχική κατάσταση κατά την έναρξη και έπειτα αποθηκεύονται εκεί τα μονοπάτια μεταξύ των κόμβων κάθε επιπέδου. Η ουρά είναι σημαντική δομή για την αναζήτηση σε πλάτος, καθώς η σειρά των μονοπατιών στην ουρά υποδεικνύει και τη σειρά κατά την αναζήτηση.
Η επέκταση του μετώπου (front) γίνεται με την εξέταση του πρώτου στοιχείου της ουράς. Για κάθε πιθανή μετάβαση της τρέχουσας κατάστασης, προστίθεται στο τέλος της ουράς. Χρησιμοποιήθηκε έλεγχος επανάληψης για την αποφυγή επανάληψης καταστάσεων κατά τη διάρκεια της εξερεύνησης, ενώ οι συνθήκες τερματισμού ήταν είτε να βρεθεί ο στόχος είτε η ουρά να είναι άδεια.

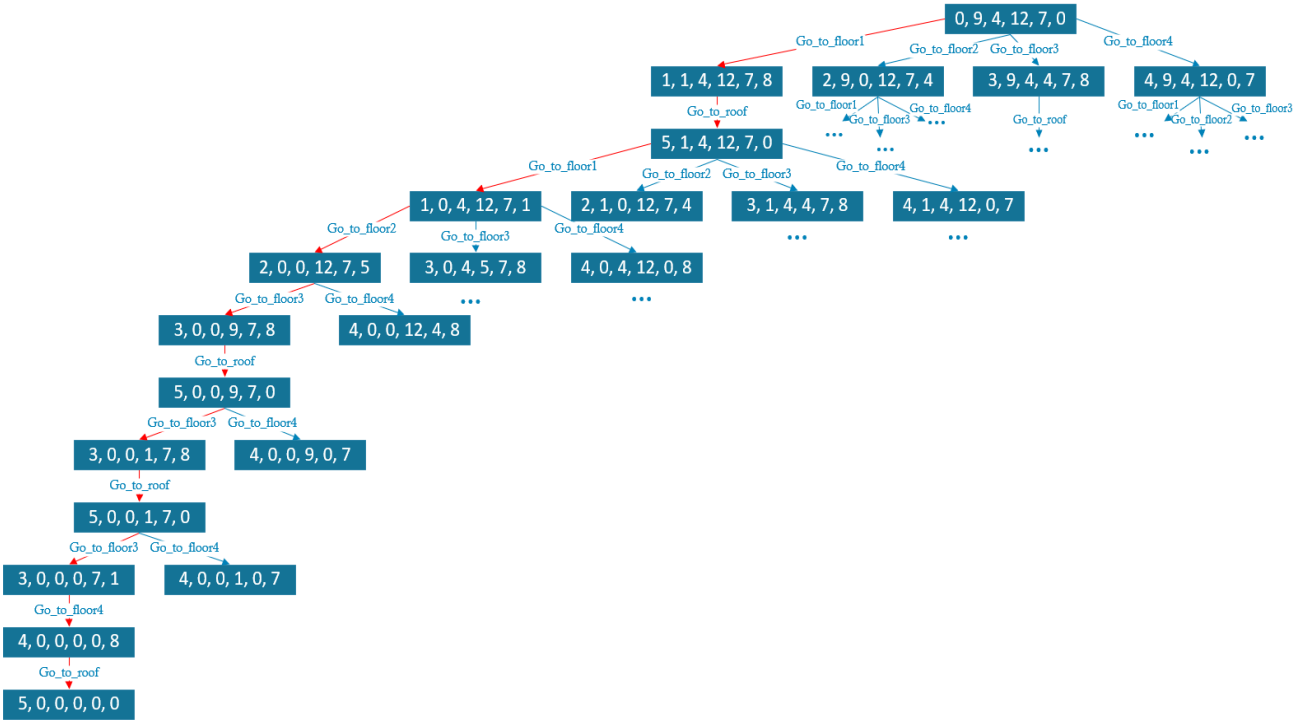
Ευριστική μέθοδος

Ο αλγόριθμος με όνομα ‘FEF’ (Fullest Elevator First) που δημιουργήθηκε στην προσπάθεια υλοποίησης μιας ευριστικής προσέγγισης για τη λύση του προβλήματος του ανελκυστήρα, βασίζεται στην προτεραιοποίηση των καταστάσεων βάσει του αριθμού των ατόμων στον ανελκυστήρα. Πιο συγκεκριμένα, διαλέγοντας τις καταστάσεις που έχουν το μέγιστο αριθμό ατόμων στον ανελκυστήρα, που επομένως καταφέρνουν να τον γεμίσουν και συντομότερα. Με αυτή την επιλογή των καταστάσεων, η μέθοδος προσπαθεί να προσφέρει μια βέλτιστη αναζήτηση θεωρώντας ότι η μείωση των κινήσεων και η βελτιστοποίηση της χωρητικότητας του ανελκυστήρα αποτελούν καλές τακτικές για την εύρεση της λύσης.
Όσον αφορά το ευριστικό κριτήριο, η μέθοδος αυτή βάζει σε προτεραιότητα τις καταστάσεις όπου ο ανελκυστήρας είναι πιο γεμάτος.
Σχετικά με το ευριστικό μηχανισμό που χρησιμοποιήθηκε, αρχικά η τρέχουσα κατάσταση προκύπτει από την πρώτη κατάσταση που βρίσκεται στη λίστα, εμπνεόμενη από τη DFS. Έπειτα, για την εύρεση των παιδιών, αφού κληθεί η find_children και δημιουργηθούν όλες οι πιθανές επόμενες καταστάσεις, ταξινομούνται βάσει του αριθμού των ατόμων του ανελκυστήρα που βρίσκεται στη τελευταία θέση της λίστας, έχοντας την κατάσταση με τον πιο γεμάτο στην αριστερότερη θέση. Η τοποθέτηση των καταστάσεων-παιδιών στο μέτωπο γίνεται ξανά στο μπροστινό μέρος του, θυμίζοντας πάλι τη μέθοδο DFS. Με παρόμοιο τρόπο λειτουργεί και η αποθήκευση στην ουρά. Όλα αυτά τα βήματα ακολουθούνται μέχρι την εύρεση της απαιτούμενης λύσης, που αποτελεί τη κατάσταση-στόχο.

Δέντρα αναζήτησης

Για αναζήτηση κατά βάθος (Depth First Search - DFS)

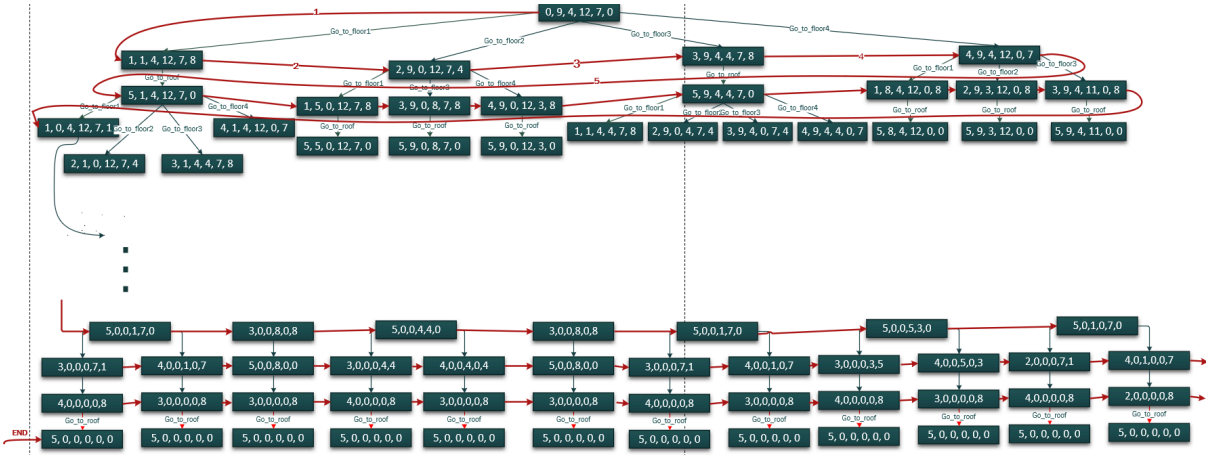
Για τη δημιουργία ενός δένδρου αναζήτησης, απαιτείται, ξεκινώντας από την αρχική κατάσταση, με την χρήση όλων των τελεστών μετάβασης, να δημιουργούνται όλες οι αποδεκτές επόμενες καταστάσεις που προκύπτουν. Στα παρακάτω διαγράμματα, δεν περιέχονται οι περιπτώσεις εφαρμογής των τελεστών που καταλήγουν σε μια κατάσταση που δε φέρει κάποιο αποτέλεσμα, όπως για παράδειγμα να πάμε στη ταράτσα με άδειο ασανσέρ. Επιπλέον, είναι προφανές ότι για όλες τις μεθόδους έχουμε το ίδιο δέντρο, αλλά οι κόμβοι επισκέπτονται με διαφορετική σειρά, η οποία καταφαίνεται σε κάθε ένα από αυτά. Για την απλούστευσή τους δεν περιέχουν όλους τους κόμβους που θα σχηματίζονταν, αλλά όσους απαιτούνται για να φανεί η λειτουργία των μεθόδων.
Στην αναζήτηση κατά βάθος (DFS) έχουμε το δέντρο:



Στο διάγραμμα, συμβολίζονται με μπλε πλαίσια οι καταστάσεις που προκύπτουν. Ξεκινώντας από την αρχική κατάσταση [0, 9, 4, 12, 7, 0], τοποθετούμε βέλη τα οποία αναγράφουν τον κάθε τελεστή μετάβασης που μπορεί να ακολουθήσει. Στο τέλος του βέλους βρίσκονται οι καταστάσεις-παιδιά που δημιουργήθηκαν από την εκτέλεση του κάθε τελεστή. Με τον τελεστή μετάβασης go_to_floor1 θα έχουμε ως αποτέλεσμα την κατάσταση [1, 1, 4, 12, 7, 8], με τον τελεστή μετάβασης go_to_floor2 θα έχουμε ως

αποτέλεσμα την κατάσταση [2, 9, 0, 12, 7, 2], κ.ο.κ. Ο τελεστής μετάβασης go_to_roof δεν θα έφερνε κάποιο διαφορετικό αποτέλεσμα, οπότε παραλείπεται.
Στη συνέχεια της αναζήτησης, η μέθοδος DFS επιλέγει κάθε φορά το αριστερότερο κλαδί για να πορευτεί, όπως φαίνεται με τα κόκκινα βέλη του διαγράμματος, συνεχίζοντας τη διαδικασία με τον ίδιο τρόπο με τον κόμβο που συναντάται εκεί.
Η αναζήτηση τερματίζει όταν συναντήσει κόμβο που περιέχει την τελική κατάσταση.

Για αναζήτηση κατά πλάτος (Breadth First Search - BFS)



Λόγω του υπερμεγέθους διαγράμματος που προκύπτει, στην παραπάνω εικόνα φαίνεται ενδεικτικά ένα πολύ μικρό μέρος του. Βλέπουμε ότι η αναζήτηση γίνεται κατά πλάτος, επιλέγοντας δηλαδή πρώτα όλες τις καταστάσεις που βρίσκονται στο ίδιο επίπεδο-βάθος τους δέντρου και έπειτα προχωρά στο επόμενο. Η διαδικασία επαναλαμβάνεται μέχρι να τελειώσει η αναζήτηση, να βρεθεί δηλαδή για πρώτη φορά κόμβος που να δείχνει την τελική κατάσταση, που αποτελεί το στόχο της αναζήτησης.

Περιπτώσεις εξαντλητικού ελέγχου και συμπεράσματα

Σε κάθε βήμα της εκπόνησης του κώδικα για την επίλυση του προβλήματος διεξήχθησαν εκτενείς δοκιμές και έλεγχοι με σκοπό την εξασφάλιση και επαλήθευση της ορθότητάς του τόσο σε συντακτικό όσο και σε λογικό επίπεδο. Καθώς τα αποτελέσματα των δοκιμών αυτών είναι πολυάριθμα, παρακάτω ακολουθούν τα αποτελέσματα για κάθε μέθοδο ξεχωριστά στην τελική μορφή του κώδικα, η οποία περιέχει την επέκταση του μετώπου και την παρακολούθηση της ουράς των μονοπατιών.

Depth-First Search

```
Choose searching method (DFS, BFS or FEF):
DFS
____BEGIN__SEARCHING__FOR__DFS____
Front:
```

```
[[0, 9, 4, 12, 7, 0]]
Queue:
[[[0, 9, 4, 12, 7, 0]]]
Front:
[[1, 1, 4, 12, 7, 8], [2, 9, 0, 12, 7, 4], [3, 9, 4, 4, 7, 8], [4,
9, 4, 12, 0, 7]]
Queue:
[[[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8]], [[0, 9, 4, 12, 7, 0],
[2, 9, 0, 12, 7, 4]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8]],
[[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7]]]
Front:
[[5, 1, 4, 12, 7, 0], [2, 9, 0, 12, 7, 4], [3, 9, 4, 4, 7, 8], [4,
9, 4, 12, 0, 7]]
Queue:
[[[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0]],
[[0, 9, 4, 12, 7, 0], [2, 9, 0, 12, 7, 4]], [[0, 9, 4, 12, 7, 0],
[3, 9, 4, 4, 7, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7]]]
Front:
[[1, 0, 4, 12, 7, 1], [2, 1, 0, 12, 7, 4], [3, 1, 4, 4, 7, 8], [4,
1, 4, 12, 0, 7], [2, 9, 0, 12, 7, 4], [3, 9, 4, 4, 7, 8], [4, 9, 4,
12, 0, 7]]
Queue:
[[[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1,
0, 4, 12, 7, 1]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1,
4, 12, 7, 0], [2, 1, 0, 12, 7, 4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4,
12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8]], [[0, 9, 4, 12,
7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [4, 1, 4, 12, 0,
7]], [[0, 9, 4, 12, 7, 0], [2, 9, 0, 12, 7, 4]], [[0, 9, 4, 12, 7,
0], [3, 9, 4, 4, 7, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7]]]
Front:
[[2, 0, 0, 12, 7, 5], [3, 0, 4, 5, 7, 8], [4, 0, 4, 12, 0, 8], [2,
1, 0, 12, 7, 4], [3, 1, 4, 4, 7, 8], [4, 1, 4, 12, 0, 7], [2, 9, 0,
12, 7, 4], [3, 9, 4, 4, 7, 8], [4, 9, 4, 12, 0, 7]]
Queue:
[[[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1,
0, 4, 12, 7, 1], [2, 0, 0, 12, 7, 5]], [[0, 9, 4, 12, 7, 0], [1, 1,
4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [3, 0, 4, 5,
7, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7,
0], [1, 0, 4, 12, 7, 1], [4, 0, 4, 12, 0, 8]], [[0, 9, 4, 12, 7, 0],
[1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [2, 1, 0, 12, 7, 4]], [[0,
9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4,
4, 7, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12,
7, 0], [4, 1, 4, 12, 0, 7]], [[0, 9, 4, 12, 7, 0], [2, 9, 0, 12, 7,
4]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8]], [[0, 9, 4, 12, 7,
0], [4, 9, 4, 12, 0, 7]]]
Front:
[[3, 0, 0, 9, 7, 8], [4, 0, 0, 12, 4, 8], [3, 0, 4, 5, 7, 8], [4, 0,
4, 12, 0, 8], [2, 1, 0, 12, 7, 4], [3, 1, 4, 4, 7, 8], [4, 1, 4, 12,
0, 7], [2, 9, 0, 12, 7, 4], [3, 9, 4, 4, 7, 8], [4, 9, 4, 12, 0, 7]]
```

Queue:
[[[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [2, 0, 0, 12, 7, 5], [3, 0, 0, 9, 7, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [2, 0, 0, 12, 7, 5], [4, 0, 0, 12, 4, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [3, 0, 4, 5, 7, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [4, 0, 4, 12, 0, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [2, 1, 0, 12, 7, 4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [4, 1, 4, 12, 0, 7]], [[0, 9, 4, 12, 7, 0], [2, 9, 0, 12, 7, 4]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7]]]
Front:
[[5, 0, 0, 9, 7, 0], [4, 0, 0, 12, 4, 8], [3, 0, 4, 5, 7, 8], [4, 0, 4, 12, 0, 8], [2, 1, 0, 12, 7, 4], [3, 1, 4, 4, 7, 8], [4, 1, 4, 12, 0, 7], [2, 9, 0, 12, 7, 4], [3, 9, 4, 4, 7, 8], [4, 9, 4, 12, 0, 7]]
Queue:
[[[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [2, 0, 0, 12, 7, 5], [3, 0, 0, 9, 7, 8], [5, 0, 0, 9, 7, 0]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [2, 0, 0, 12, 7, 5], [4, 0, 0, 12, 4, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [3, 0, 4, 5, 7, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [4, 0, 4, 12, 0, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [2, 1, 0, 12, 7, 4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [4, 1, 4, 12, 0, 7]], [[0, 9, 4, 12, 7, 0], [2, 9, 0, 12, 7, 4]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7]]]
Front:
[[3, 0, 0, 1, 7, 8], [4, 0, 0, 9, 0, 7], [4, 0, 0, 12, 4, 8], [3, 0, 4, 5, 7, 8], [4, 0, 4, 12, 0, 8], [2, 1, 0, 12, 7, 4], [3, 1, 4, 4, 7, 8], [4, 1, 4, 12, 0, 7], [2, 9, 0, 12, 7, 4], [3, 9, 4, 4, 7, 8], [4, 9, 4, 12, 0, 7]]
Queue:
[[[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [2, 0, 0, 12, 7, 5], [3, 0, 0, 9, 7, 8], [5, 0, 0, 9, 7, 0], [3, 0, 0, 1, 7, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [2, 0, 0, 12, 7, 5], [3, 0, 0, 9, 7, 8], [5, 0, 0, 9, 7, 0], [4, 0, 0, 9, 0, 7]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [2, 0, 0, 12, 7, 5], [4, 0, 0, 12, 4, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [3, 0, 4, 5, 7, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [4, 0, 4, 12, 0, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0],

[2, 1, 0, 12, 7, 4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [4, 1, 4, 12, 0, 7]], [[0, 9, 4, 12, 7, 0], [2, 9, 0, 12, 7, 4]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7]]]
Front:
[[5, 0, 0, 1, 7, 0], [4, 0, 0, 9, 0, 7], [4, 0, 0, 12, 4, 8], [3, 0, 4, 5, 7, 8], [4, 0, 4, 12, 0, 8], [2, 1, 0, 12, 7, 4], [3, 1, 4, 4, 7, 8], [4, 1, 4, 12, 0, 7], [2, 9, 0, 12, 7, 4], [3, 9, 4, 4, 7, 8], [4, 9, 4, 12, 0, 7]]
Queue:
[[[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [2, 0, 0, 12, 7, 5], [3, 0, 0, 9, 7, 8], [5, 0, 0, 9, 7, 0], [3, 0, 0, 1, 7, 8], [5, 0, 0, 1, 7, 0]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [2, 0, 0, 12, 7, 5], [3, 0, 0, 9, 7, 8], [5, 0, 0, 9, 7, 0], [4, 0, 0, 9, 0, 7]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [2, 0, 0, 12, 7, 5], [4, 0, 0, 12, 4, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [3, 0, 4, 5, 7, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [4, 0, 4, 12, 0, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [2, 1, 0, 12, 7, 4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [4, 1, 4, 12, 0, 7]], [[0, 9, 4, 12, 7, 0], [2, 9, 0, 12, 7, 4]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7]]]
Front:
[[3, 0, 0, 0, 7, 1], [4, 0, 0, 1, 0, 7], [4, 0, 0, 9, 0, 7], [4, 0, 0, 12, 4, 8], [3, 0, 4, 5, 7, 8], [4, 0, 4, 12, 0, 8], [2, 1, 0, 12, 7, 4], [3, 1, 4, 4, 7, 8], [4, 1, 4, 12, 0, 7], [2, 9, 0, 12, 7, 4], [3, 9, 4, 4, 7, 8], [4, 9, 4, 12, 0, 7]]
Queue:
[[[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [2, 0, 0, 12, 7, 5], [3, 0, 0, 9, 7, 8], [5, 0, 0, 9, 7, 0], [3, 0, 0, 1, 7, 8], [5, 0, 0, 1, 7, 0], [3, 0, 0, 0, 7, 1]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [2, 0, 0, 12, 7, 5], [3, 0, 0, 9, 7, 8], [5, 0, 0, 9, 7, 0], [3, 0, 0, 1, 7, 8], [5, 0, 0, 1, 7, 0], [4, 0, 0, 1, 0, 7]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [2, 0, 0, 12, 7, 5], [3, 0, 0, 9, 7, 8], [5, 0, 0, 9, 7, 0], [4, 0, 0, 9, 0, 7]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [2, 0, 0, 12, 7, 5], [4, 0, 0, 12, 4, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [3, 0, 4, 5, 7, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [4, 0, 4, 12, 0, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [2, 1, 0, 12, 7, 4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7,

```
8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0],
[4, 1, 4, 12, 0, 7]], [[0, 9, 4, 12, 7, 0], [2, 9, 0, 12, 7, 4]],
[[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8]], [[0, 9, 4, 12, 7, 0], [4,
9, 4, 12, 0, 7]]]
Front:
[[4, 0, 0, 0, 0, 8], [4, 0, 0, 1, 0, 7], [4, 0, 0, 9, 0, 7], [4, 0,
0, 12, 4, 8], [3, 0, 4, 5, 7, 8], [4, 0, 4, 12, 0, 8], [2, 1, 0, 12,
7, 4], [3, 1, 4, 4, 7, 8], [4, 1, 4, 12, 0, 7], [2, 9, 0, 12, 7, 4],
[3, 9, 4, 4, 7, 8], [4, 9, 4, 12, 0, 7]]
Queue:
[[[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1,
0, 4, 12, 7, 1], [2, 0, 0, 12, 7, 5], [3, 0, 0, 9, 7, 8], [5, 0, 0,
9, 7, 0], [3, 0, 0, 1, 7, 8], [5, 0, 0, 1, 7, 0], [3, 0, 0, 0, 7,
1], [4, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8],
[5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [2, 0, 0, 12, 7, 5], [3,
0, 0, 9, 7, 8], [5, 0, 0, 9, 7, 0], [3, 0, 0, 1, 7, 8], [5, 0, 0, 1,
7, 0], [4, 0, 0, 1, 0, 7]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7,
8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [2, 0, 0, 12, 7, 5],
[3, 0, 0, 9, 7, 8], [5, 0, 0, 9, 7, 0], [4, 0, 0, 9, 0, 7]], [[0, 9,
4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4,
12, 7, 1], [2, 0, 0, 12, 7, 5], [4, 0, 0, 12, 4, 8]], [[0, 9, 4, 12,
7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7,
1], [3, 0, 4, 5, 7, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8],
[5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1], [4, 0, 4, 12, 0, 8]], [[0,
9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [2, 1, 0,
12, 7, 4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12,
7, 0], [3, 1, 4, 4, 7, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7,
8], [5, 1, 4, 12, 7, 0], [4, 1, 4, 12, 0, 7]], [[0, 9, 4, 12, 7, 0],
[2, 9, 0, 12, 7, 4]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8]],
[[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7]]]
_GOAL_FOUND_
[5, 0, 0, 0, 0, 0]
[[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1,
0, 4, 12, 7, 1], [2, 0, 0, 12, 7, 5], [3, 0, 0, 9, 7, 8], [5, 0, 0,
9, 7, 0], [3, 0, 0, 1, 7, 8], [5, 0, 0, 1, 7, 0], [3, 0, 0, 0, 7,
1], [4, 0, 0, 0, 0, 8], [5, 0, 0, 0, 0, 0]]

Process finished with exit code 0
```

Τα αποτελέσματα εκτέλεσης του αλγορίθμου DFS παρουσιάζονται παραπάνω, όπου:

- Ο πίνακας Front παρουσιάζει τις καταστάσεις των κόμβων που εξετάζονται από τον αλγόριθμο. Κάθε γραμμή του Front αντιπροσωπεύει ένα κόμβο, για παράδειγμα η πρώτη γραμμή του Front είναι ο κόμβος [0, 9, 4, 12, 7, 0]. Κάθε κατάσταση στο Front εμφανίζεται μόνο μια φορά.
- Ο πίνακας Queue αναπαριστά την ουρά του αλγορίθμου. Κάθε στοιχείο της λίστας είναι μια λίστα που περιέχει δύο υπό-λίστες: την κατάσταση στο Front

και τη νέα κατάσταση που προκύπτει μετά την εξέταση του επόμενου βήματος. Για παράδειγμα, η πρώτη λίστα Queue είναι [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8]].

- Οι κόμβοι εξερευνώνται με βάση τη σειρά που προστίθενται στην ουρά και η εξερεύνηση τους εξαρτάται από την αρχική κατάσταση. Για παράδειγμα, στο τελευταίο βήμα, ο κόμβος [3, 0, 0, 0, 7, 1] προστίθεται στον Front και η κατάσταση του προστίθεται στην Queue.
- Κάθε κόμβος εξερευνάται για τους γείτονές του, και οι νέοι κόμβοι προστίθενται στο Front και στην Queue.
- Ο αλγόριθμος εξερευνά μια κατάσταση του δένδρου καταστάσεων (πλευρά) πριν κινηθεί προς τα κάτω στο ίδιο επίπεδο. Αυτό φαίνεται καθώς εξερευνά τις καταστάσεις [1, 1, 4, 12, 7, 8] και [2, 9, 0, 12, 7, 4] προτού κινηθεί στις καταστάσεις [5, 1, 4, 12, 7, 0] και [1, 0, 4, 12, 7, 1].
- Ο DFS συνεχίζει την εξερεύνηση μέχρι να φτάσει σε κατάσταση που δεν υπάρχουν άλλοι γείτονες προς εξέταση, και τότε επιστρέφει προς τα πίσω για να εξερευνήσει άλλες πλευρές.
- Ολοκληρώνεται όταν εξερευνήσει όλο το δένδρο ή φτάσει σε ένα σημείο που η αναζήτηση δεν μπορεί να συνεχίσει.
- Τα αποτελέσματα του κώδικα είναι ακριβώς όπως αναμέναμε και όπως καταφαίνεται και η διαδικασία αναζήτησης στο δέντρο.

Breadth-First Search

(Ενδεικτικά αποτελέσματα)

```
Choose searching method (DFS, BFS or FEF):
____BEGIN__SEARCHING__FOR__BFS____
Front:
[[0, 9, 4, 12, 7, 0]]
Queue:
[[[0, 9, 4, 12, 7, 0]]]
Front:
[[4, 9, 4, 12, 0, 7], [3, 9, 4, 4, 7, 8], [2, 9, 0, 12, 7, 4], [1,
1, 4, 12, 7, 8]]
Queue:
[[[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7]], [[0, 9, 4, 12, 7, 0],
[3, 9, 4, 4, 7, 8]], [[0, 9, 4, 12, 7, 0], [2, 9, 0, 12, 7, 4]],
[[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8]]]
Front:
[[3, 9, 4, 4, 7, 8], [2, 9, 0, 12, 7, 4], [1, 1, 4, 12, 7, 8], [3,
9, 4, 11, 0, 8], [2, 9, 3, 12, 0, 8], [1, 8, 4, 12, 0, 8]]
Queue:
[[[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8]], [[0, 9, 4, 12, 7, 0],
[2, 9, 0, 12, 7, 4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8]],
[[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [3, 9, 4, 11, 0, 8]],
[[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [2, 9, 3, 12, 0, 8]],
[[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [1, 8, 4, 12, 0, 8]]]
```

Front:
[[2, 9, 0, 12, 7, 4], [1, 1, 4, 12, 7, 8], [3, 9, 4, 11, 0, 8], [2, 9, 3, 12, 0, 8], [1, 8, 4, 12, 0, 8], [5, 9, 4, 4, 7, 0]]
Queue:
[[[0, 9, 4, 12, 7, 0], [2, 9, 0, 12, 7, 4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [3, 9, 4, 11, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [2, 9, 3, 12, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [1, 8, 4, 12, 0, 8]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8], [5, 9, 4, 4, 7, 0]]]

. . . . . .

. . . . . .

. . . . . .

Front:
[[3, 0, 0, 0, 7, 1], [4, 0, 0, 5, 0, 3], [3, 0, 0, 0, 3, 5], [1, 0, 0, 0, 0, 8], [2, 0, 0, 0, 0, 8], [1, 0, 0, 0, 0, 8], [3, 0, 0, 0, 0, 8], [1, 0, 0, 0, 0, 8], [3, 0, 0, 0, 0, 8], [1, 0, 0, 0, 0, 8], [2, 0, 0, 0, 0, 8], [1, 0, 0, 0, 0, 8], [3, 0, 0, 0, 0, 8], [2, 0, 0, 0, 0, 8], [3, 0, 0, 0, 0, 8], [5, 0, 0, 0, 0, 0], [5, 0, 0, 0, 0, 0], [5, 0, 0, 0, 0, 0], [1, 0, 0, 0, 0, 8], [4, 0, 0, 0, 0, 8], [2, 0, 0, 0, 0, 8], [4, 0, 0, 0, 0, 8], [5, 0, 0, 0, 0, 0], [4, 0, 0, 0, 0, 8], [2, 0, 0, 0, 0, 8], [4, 0, 0, 0, 0, 8], [3, 0, 0, 0, 0, 8], [4, 0, 0, 0, 0, 8], [4, 0, 0, 0, 0, 8], [3, 0, 0, 0, 0, 8]]
Queue:
[[[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8], [5, 9, 4, 4, 7, 0], [2, 9, 0, 4, 7, 4], [1, 5, 0, 4, 7, 8], [5, 5, 0, 4, 7, 0], [1, 0, 0, 4, 7, 5], [3, 0, 0, 1, 7, 8], [5, 0, 0, 1, 7, 0], [3, 0, 0, 0, 7, 1]], [[0, 9, 4, 12, 7, 0], [2, 9, 0, 12, 7, 4], [4, 9, 0, 12, 3, 8], [5, 9, 0, 12, 3, 0], [1, 1, 0, 12, 3, 8], [5, 1, 0, 12, 3, 0], [1, 0, 0, 12, 3, 1], [3, 0, 0, 5, 3, 8], [5, 0, 0, 5, 3, 0], [4, 0, 0, 5, 0, 3]], [[0, 9, 4, 12, 7, 0], [2, 9, 0, 12, 7, 4], [4, 9, 0, 12, 3, 8], [5, 9, 0, 12, 3, 0], [1, 1, 0, 12, 3, 8], [5, 1, 0, 12, 3, 0], [1, 0, 0, 12, 3, 1], [3, 0, 0, 5, 3, 8], [5, 0, 0, 5, 3, 0], [3, 0, 0, 0, 3, 5]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [3, 9, 4, 11, 0, 8], [5, 9, 4, 11, 0, 0], [3, 9, 4, 3, 0, 8], [5, 9, 4, 3, 0, 0], [3, 9, 4, 0, 0, 3], [1, 4, 4, 0, 0, 8], [5, 4, 4, 0, 0, 0], [2, 4, 0, 0, 0, 4], [1, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [3, 9, 4, 11, 0, 8], [5, 9, 4, 11, 0, 0], [3, 9, 4, 3, 0, 8], [5, 9, 4, 3, 0, 0], [3, 9, 4, 0, 0, 3], [1, 4, 4, 0, 0, 8], [5, 4, 4, 0, 0, 0], [1, 0, 4, 0, 0, 4], [2, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [3, 9, 4, 11, 0, 8], [5, 9, 4, 11, 0, 0], [3, 9, 4, 3, 0, 8], [5, 9, 4, 3, 0, 0], [2, 9, 0, 3, 0, 4],

[1, 5, 0, 3, 0, 8], [5, 5, 0, 3, 0, 0], [3, 5, 0, 0, 0, 3], [1, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [3, 9, 4, 11, 0, 8], [5, 9, 4, 11, 0, 0], [3, 9, 4, 3, 0, 8], [5, 9, 4, 3, 0, 0], [2, 9, 0, 3, 0, 4], [1, 5, 0, 3, 0, 8], [5, 5, 0, 3, 0, 0], [1, 0, 0, 3, 0, 5], [3, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [3, 9, 4, 11, 0, 8], [5, 9, 4, 11, 0, 0], [3, 9, 4, 3, 0, 8], [5, 9, 4, 3, 0, 0], [1, 1, 4, 3, 0, 8], [5, 1, 4, 3, 0, 0], [3, 1, 4, 0, 0, 3], [2, 1, 0, 0, 0, 7], [1, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [3, 9, 4, 11, 0, 8], [5, 9, 4, 11, 0, 0], [3, 9, 4, 3, 0, 8], [5, 9, 4, 3, 0, 0], [1, 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]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [3, 9, 4, 11, 0, 8], [5, 9, 4, 11, 0, 0], [3, 9, 4, 3, 0, 8], [5, 9, 4, 3, 0, 0], [1, 1, 4, 3, 0, 8], [5, 1, 4, 3, 0, 0], [1, 0, 4, 3, 0, 1], [2, 0, 0, 3, 0, 5], [3, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [3, 9, 4, 11, 0, 8], [5, 9, 4, 11, 0, 0], [2, 9, 0, 11, 0, 4], [3, 9, 0, 7, 0, 8], [5, 9, 0, 7, 0, 0], [1, 1, 0, 7, 0, 8], [5, 1, 0, 7, 0, 0], [1, 0, 0, 7, 0, 1], [3, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [2, 9, 3, 12, 0, 8], [5, 9, 3, 12, 0, 0], [3, 9, 3, 4, 0, 8], [5, 9, 3, 4, 0, 0], [3, 9, 3, 0, 0, 4], [1, 5, 3, 0, 0, 8], [5, 5, 3, 0, 0, 0], [2, 5, 0, 0, 0, 3], [1, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [2, 9, 3, 12, 0, 8], [5, 9, 3, 12, 0, 0], [3, 9, 3, 4, 0, 8], [5, 9, 3, 4, 0, 0], [3, 9, 3, 0, 0, 4], [1, 5, 3, 0, 0, 8], [5, 5, 3, 0, 0, 0], [1, 0, 3, 0, 0, 5], [2, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [2, 9, 3, 12, 0, 8], [5, 9, 3, 12, 0, 0], [3, 9, 3, 4, 0, 8], [5, 9, 3, 4, 0, 0], [2, 9, 0, 4, 0, 3], [1, 4, 0, 4, 0, 8], [5, 4, 0, 4, 0, 0], [3, 4, 0, 0, 0, 4], [1, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [2, 9, 3, 12, 0, 8], [5, 9, 3, 12, 0, 0], [3, 9, 3, 4, 0, 8], [5, 9, 3, 4, 0, 0], [2, 9, 0, 4, 0, 3], [1, 4, 0, 4, 0, 8], [5, 4, 0, 4, 0, 0], [1, 0, 0, 4, 0, 4], [3, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [2, 9, 3, 12, 0, 8], [5, 9, 3, 12, 0, 0], [3, 9, 3, 4, 0, 8], [5, 9, 3, 4, 0, 0], [1, 1, 3, 4, 0, 8], [5, 1, 3, 4, 0, 0], [1, 0, 3, 4, 0, 1], [3, 0, 3, 0, 0, 5], [2, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [2, 9, 3, 12, 0, 8], [5, 9, 3, 12, 0, 0], [1, 1, 3, 12, 0, 8], [5, 1, 3, 12, 0, 0], [1, 0, 3, 12, 0, 1], [3, 0, 3, 5, 0, 8], [5, 0, 3, 5, 0, 0], [2, 0, 0, 5, 0, 3], [3, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [1, 8, 4, 12, 0, 8], [5, 8, 4, 12, 0, 0], [3, 8, 4, 4, 0, 8], [5, 8, 4, 4, 0, 0], [3, 8, 4, 0, 0, 4], [2, 8, 0, 0, 0, 8], [5, 8, 0, 0, 0, 0], [1, 0, 0, 0, 0, 8], [5, 0, 0, 0, 0, 0]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [1, 8, 4, 12, 0, 8], [5, 8, 4, 12, 0, 0], [3, 8, 4, 4, 0, 8], [5, 8, 4, 4, 0, 0], [1, 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]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [1, 8, 4, 12, 0, 8], [5, 8, 4, 12, 0, 0], [3, 8, 4, 4, 0, 8], [5, 8, 4, 4, 0, 0], [1, 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]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8], [5, 9, 4, 4, 7, 0], [3, 9,

4, 0, 7, 4], [4, 9, 4, 0, 3, 8], [5, 9, 4, 0, 3, 0], [2, 9, 0, 0, 3, 4], [1, 5, 0, 0, 3, 8], [5, 5, 0, 0, 3, 0], [4, 5, 0, 0, 0, 3], [1, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8], [5, 9, 4, 4, 7, 0], [3, 9, 4, 0, 7, 4], [4, 9, 4, 0, 3, 8], [5, 9, 4, 0, 3, 0], [2, 9, 0, 0, 3, 4], [1, 5, 0, 0, 3, 8], [5, 5, 0, 0, 3, 0], [1, 0, 0, 0, 3, 5], [4, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8], [5, 9, 4, 4, 7, 0], [3, 9, 4, 0, 7, 4], [4, 9, 4, 0, 3, 8], [5, 9, 4, 0, 3, 0], [1, 1, 4, 0, 3, 8], [5, 1, 4, 0, 3, 0], [1, 0, 4, 0, 3, 1], [4, 0, 4, 0, 0, 4], [2, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8], [5, 9, 4, 4, 7, 0], [3, 9, 4, 0, 7, 4], [4, 9, 4, 0, 3, 8], [5, 9, 4, 0, 3, 0], [1, 1, 4, 0, 3, 8], [5, 1, 4, 0, 3, 0], [1, 0, 4, 0, 3, 1], [2, 0, 0, 0, 3, 5], [4, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8], [5, 9, 4, 4, 7, 0], [3, 9, 4, 0, 7, 4], [2, 9, 0, 0, 7, 8], [5, 9, 0, 0, 7, 0], [1, 1, 0, 0, 7, 8], [5, 1, 0, 0, 7, 0], [1, 0, 0, 0, 7, 1], [4, 0, 0, 0, 0, 8], [5, 0, 0, 0, 0, 0]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8], [5, 9, 4, 4, 7, 0], [3, 9, 4, 0, 7, 4], [1, 5, 4, 0, 7, 8], [5, 5, 4, 0, 7, 0], [1, 0, 4, 0, 7, 5], [4, 0, 4, 0, 4, 8], [5, 0, 4, 0, 4, 0], [2, 0, 0, 0, 4, 4], [4, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8], [5, 9, 4, 4, 7, 0], [3, 9, 4, 0, 7, 4], [1, 5, 4, 0, 7, 8], [5, 5, 4, 0, 7, 0], [1, 0, 4, 0, 7, 5], [2, 0, 1, 0, 7, 8], [5, 0, 1, 0, 7, 0], [4, 0, 1, 0, 0, 7], [2, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8], [5, 9, 4, 4, 7, 0], [3, 9, 4, 0, 7, 4], [1, 5, 4, 0, 7, 8], [5, 5, 4, 0, 7, 0], [1, 0, 4, 0, 7, 5], [2, 0, 1, 0, 7, 8], [5, 0, 1, 0, 7, 0], [2, 0, 0, 0, 7, 1], [4, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8], [5, 9, 4, 4, 7, 0], [2, 9, 0, 4, 7, 4], [4, 9, 0, 4, 3, 8], [5, 9, 0, 4, 3, 0], [1, 1, 0, 4, 3, 8], [5, 1, 0, 4, 3, 0], [1, 0, 0, 4, 3, 1], [4, 0, 0, 4, 0, 4], [3, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8], [5, 9, 4, 4, 7, 0], [2, 9, 0, 4, 7, 4], [4, 9, 0, 4, 3, 8], [5, 9, 0, 4, 3, 0], [1, 1, 0, 4, 3, 8], [5, 1, 0, 4, 3, 0], [1, 0, 0, 4, 3, 1], [3, 0, 0, 0, 3, 5], [4, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8], [5, 9, 4, 4, 7, 0], [2, 9, 0, 4, 7, 4], [1, 5, 0, 4, 7, 8], [5, 5, 0, 4, 7, 0], [1, 0, 0, 4, 7, 5], [4, 0, 0, 4, 4, 8], [5, 0, 0, 4, 4, 0], [3, 0, 0, 0, 4, 4], [4, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8], [5, 9, 4, 4, 7, 0], [2, 9, 0, 4, 7, 4], [1, 5, 0, 4, 7, 8], [5, 5, 0, 4, 7, 0], [1, 0, 0, 4, 7, 5], [3, 0, 0, 1, 7, 8], [5, 0, 0, 1, 7, 0], [4, 0, 0, 1, 0, 7], [3, 0, 0, 0, 0, 8]]]

Front:

[[4, 0, 0, 5, 0, 3], [3, 0, 0, 0, 3, 5], [1, 0, 0, 0, 0, 8], [2, 0, 0, 0, 0, 8], [1, 0, 0, 0, 0, 8], [3, 0, 0, 0, 0, 8], [1, 0, 0, 0, 0, 8], [1, 0, 0, 0, 0, 8], [3, 0, 0, 0, 0, 8], [3, 0, 0, 0, 0, 8], [1, 0, 0, 0, 0, 8], [2, 0, 0, 0, 0, 8], [1, 0, 0, 0, 0, 8], [3, 0, 0, 0, 0, 8], [2, 0, 0, 0, 0, 8], [3, 0, 0, 0, 0, 8], [5, 0, 0, 0, 0, 0], [5, 0, 0, 0, 0, 0], [5, 0, 0, 0, 0, 0], [1, 0, 0, 0, 0, 8], [4, 0, 0, 0, 0, 8], [2, 0, 0, 0, 0, 8], [4, 0, 0, 0, 0, 8], [5, 0, 0, 0, 0, 0], [4, 0, 0, 0, 0, 8], [2, 0, 0, 0, 0, 8], [4, 0, 0, 0, 0, 8], [3,

0, 0, 0, 0, 8], [4, 0, 0, 0, 0, 8], [4, 0, 0, 0, 0, 8], [3, 0, 0, 0, 0, 8], [4, 0, 0, 0, 0, 8]]
Queue:
[[[0, 9, 4, 12, 7, 0], [2, 9, 0, 12, 7, 4], [4, 9, 0, 12, 3, 8], [5, 9, 0, 12, 3, 0], [1, 1, 0, 12, 3, 8], [5, 1, 0, 12, 3, 0], [1, 0, 0, 12, 3, 1], [3, 0, 0, 5, 3, 8], [5, 0, 0, 5, 3, 0], [4, 0, 0, 5, 0, 3]], [[0, 9, 4, 12, 7, 0], [2, 9, 0, 12, 7, 4], [4, 9, 0, 12, 3, 8], [5, 9, 0, 12, 3, 0], [1, 1, 0, 12, 3, 8], [5, 1, 0, 12, 3, 0], [1, 0, 0, 12, 3, 1], [3, 0, 0, 5, 3, 8], [5, 0, 0, 5, 3, 0], [3, 0, 0, 0, 3, 5]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [3, 9, 4, 11, 0, 8], [5, 9, 4, 11, 0, 0], [3, 9, 4, 3, 0, 8], [5, 9, 4, 3, 0, 0], [3, 9, 4, 0, 0, 3], [1, 4, 4, 0, 0, 8], [5, 4, 4, 0, 0, 0], [2, 4, 0, 0, 0, 4], [1, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [3, 9, 4, 11, 0, 8], [5, 9, 4, 11, 0, 0], [3, 9, 4, 3, 0, 8], [5, 9, 4, 3, 0, 0], [3, 9, 4, 0, 0, 3], [1, 4, 4, 0, 0, 8], [5, 4, 4, 0, 0, 0], [1, 0, 4, 0, 0, 4], [2, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [3, 9, 4, 11, 0, 8], [5, 9, 4, 11, 0, 0], [3, 9, 4, 3, 0, 8], [5, 9, 4, 3, 0, 0], [2, 9, 0, 3, 0, 4], [1, 5, 0, 3, 0, 8], [5, 5, 0, 3, 0, 0], [3, 5, 0, 0, 0, 3], [1, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [3, 9, 4, 11, 0, 8], [5, 9, 4, 11, 0, 0], [3, 9, 4, 3, 0, 8], [5, 9, 4, 3, 0, 0], [2, 9, 0, 3, 0, 4], [1, 5, 0, 3, 0, 8], [5, 5, 0, 3, 0, 0], [1, 0, 0, 3, 0, 5], [3, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [3, 9, 4, 11, 0, 8], [5, 9, 4, 11, 0, 0], [3, 9, 4, 3, 0, 8], [5, 9, 4, 3, 0, 0], [1, 1, 4, 3, 0, 8], [5, 1, 4, 3, 0, 0], [3, 1, 4, 0, 0, 3], [2, 1, 0, 0, 0, 7], [1, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [3, 9, 4, 11, 0, 8], [5, 9, 4, 11, 0, 0], [3, 9, 4, 3, 0, 8], [5, 9, 4, 3, 0, 0], [1, 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]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [3, 9, 4, 11, 0, 8], [5, 9, 4, 11, 0, 0], [3, 9, 4, 3, 0, 8], [5, 9, 4, 3, 0, 0], [1, 1, 4, 3, 0, 8], [5, 1, 4, 3, 0, 0], [1, 0, 4, 3, 0, 1], [2, 0, 0, 3, 0, 5], [3, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [3, 9, 4, 11, 0, 8], [5, 9, 4, 11, 0, 0], [2, 9, 0, 11, 0, 4], [3, 9, 0, 7, 0, 8], [5, 9, 0, 7, 0, 0], [1, 1, 0, 7, 0, 8], [5, 1, 0, 7, 0, 0], [1, 0, 0, 7, 0, 1], [3, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [2, 9, 3, 12, 0, 8], [5, 9, 3, 12, 0, 0], [3, 9, 3, 4, 0, 8], [5, 9, 3, 4, 0, 0], [3, 9, 3, 0, 0, 4], [1, 5, 3, 0, 0, 8], [5, 5, 3, 0, 0, 0], [2, 5, 0, 0, 0, 3], [1, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [2, 9, 3, 12, 0, 8], [5, 9, 3, 12, 0, 0], [3, 9, 3, 4, 0, 8], [5, 9, 3, 4, 0, 0], [3, 9, 3, 0, 0, 4], [1, 5, 3, 0, 0, 8], [5, 5, 3, 0, 0, 0], [1, 0, 3, 0, 0, 5], [2, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [2, 9, 3, 12, 0, 8], [5, 9, 3, 12, 0, 0], [3, 9, 3, 4, 0, 8], [5, 9, 3, 4, 0, 0], [2, 9, 0, 4, 0, 3], [1, 4, 0, 4, 0, 8], [5, 4, 0, 4, 0, 0], [3, 4, 0, 0, 0, 4], [1, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [2, 9, 3, 12, 0, 8], [5, 9, 3, 12, 0, 0], [3, 9, 3, 4, 0, 8], [5, 9, 3, 4, 0, 0], [2, 9, 0, 4, 0, 3], [1, 4, 0, 4, 0, 8], [5, 4, 0, 4, 0, 0], [1, 0, 0, 4, 0,

4], [3, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [2, 9, 3, 12, 0, 8], [5, 9, 3, 12, 0, 0], [3, 9, 3, 4, 0, 8], [5, 9, 3, 4, 0, 0], [1, 1, 3, 4, 0, 8], [5, 1, 3, 4, 0, 0], [1, 0, 3, 4, 0, 1], [3, 0, 3, 0, 0, 5], [2, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [2, 9, 3, 12, 0, 8], [5, 9, 3, 12, 0, 0], [1, 1, 3, 12, 0, 8], [5, 1, 3, 12, 0, 0], [1, 0, 3, 12, 0, 1], [3, 0, 3, 5, 0, 8], [5, 0, 3, 5, 0, 0], [2, 0, 0, 5, 0, 3], [3, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [1, 8, 4, 12, 0, 8], [5, 8, 4, 12, 0, 0], [3, 8, 4, 4, 0, 8], [5, 8, 4, 4, 0, 0], [3, 8, 4, 0, 0, 4], [2, 8, 0, 0, 0, 8], [5, 8, 0, 0, 0, 0], [1, 0, 0, 0, 0, 8], [5, 0, 0, 0, 0, 0]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [1, 8, 4, 12, 0, 8], [5, 8, 4, 12, 0, 0], [3, 8, 4, 4, 0, 8], [5, 8, 4, 4, 0, 0], [1, 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]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [1, 8, 4, 12, 0, 8], [5, 8, 4, 12, 0, 0], [3, 8, 4, 4, 0, 8], [5, 8, 4, 4, 0, 0], [1, 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]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8], [5, 9, 4, 4, 7, 0], [3, 9, 4, 0, 7, 4], [4, 9, 4, 0, 3, 8], [5, 9, 4, 0, 3, 0], [2, 9, 0, 0, 3, 4], [1, 5, 0, 0, 3, 8], [5, 5, 0, 0, 3, 0], [4, 5, 0, 0, 0, 3], [1, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8], [5, 9, 4, 4, 7, 0], [3, 9, 4, 0, 7, 4], [4, 9, 4, 0, 3, 8], [5, 9, 4, 0, 3, 0], [2, 9, 0, 0, 3, 4], [1, 5, 0, 0, 3, 8], [5, 5, 0, 0, 3, 0], [1, 0, 0, 0, 3, 5], [4, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8], [5, 9, 4, 4, 7, 0], [3, 9, 4, 0, 7, 4], [4, 9, 4, 0, 3, 8], [5, 9, 4, 0, 3, 0], [1, 1, 4, 0, 3, 8], [5, 1, 4, 0, 3, 0], [1, 0, 4, 0, 3, 1], [4, 0, 4, 0, 0, 4], [2, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8], [5, 9, 4, 4, 7, 0], [3, 9, 4, 0, 7, 4], [4, 9, 4, 0, 3, 8], [5, 9, 4, 0, 3, 0], [1, 1, 4, 0, 3, 8], [5, 1, 4, 0, 3, 0], [1, 0, 4, 0, 3, 1], [2, 0, 0, 0, 3, 5], [4, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8], [5, 9, 4, 4, 7, 0], [3, 9, 4, 0, 7, 4], [2, 9, 0, 0, 7, 8], [5, 9, 0, 0, 7, 0], [1, 1, 0, 0, 7, 8], [5, 1, 0, 0, 7, 0], [1, 0, 0, 0, 7, 1], [4, 0, 0, 0, 0, 8], [5, 0, 0, 0, 0, 0]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8], [5, 9, 4, 4, 7, 0], [3, 9, 4, 0, 7, 4], [1, 5, 4, 0, 7, 8], [5, 5, 4, 0, 7, 0], [1, 0, 4, 0, 7, 5], [4, 0, 4, 0, 4, 8], [5, 0, 4, 0, 4, 0], [2, 0, 0, 0, 4, 4], [4, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8], [5, 9, 4, 4, 7, 0], [3, 9, 4, 0, 7, 4], [1, 5, 4, 0, 7, 8], [5, 5, 4, 0, 7, 0], [1, 0, 4, 0, 7, 5], [2, 0, 1, 0, 7, 8], [5, 0, 1, 0, 7, 0], [4, 0, 1, 0, 0, 7], [2, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8], [5, 9, 4, 4, 7, 0], [3, 9, 4, 0, 7, 4], [1, 5, 4, 0, 7, 8], [5, 5, 4, 0, 7, 0], [1, 0, 4, 0, 7, 5], [2, 0, 1, 0, 7, 8], [5, 0, 1, 0, 7, 0], [2, 0, 0, 0, 7, 1], [4, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8], [5, 9, 4, 4, 7, 0], [2, 9, 0, 4, 7, 4], [4, 9, 0, 4, 3, 8], [5, 9, 0, 4, 3, 0], [1, 1, 0, 4, 3, 8], [5, 1, 0, 4, 3, 0], [1, 0, 0, 4, 3, 1], [4, 0, 0, 4, 0, 4], [3, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8], [5, 9, 4, 4, 7, 0], [2, 9, 0, 4, 7, 4], [4, 9, 0, 4, 3, 8], [5, 9, 0, 4, 3, 0], [1, 1, 0, 4, 3, 8], [5, 1, 0, 4, 3, 0], [1,

```
0, 0, 4, 3, 1], [3, 0, 0, 0, 3, 5], [4, 0, 0, 0, 0, 8]], [[0, 9, 4,
12, 7, 0], [3, 9, 4, 4, 7, 8], [5, 9, 4, 4, 7, 0], [2, 9, 0, 4, 7,
4], [1, 5, 0, 4, 7, 8], [5, 5, 0, 4, 7, 0], [1, 0, 0, 4, 7, 5], [4,
0, 0, 4, 4, 8], [5, 0, 0, 4, 4, 0], [3, 0, 0, 0, 4, 4], [4, 0, 0, 0,
0, 8]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8], [5, 9, 4, 4, 7,
0], [2, 9, 0, 4, 7, 4], [1, 5, 0, 4, 7, 8], [5, 5, 0, 4, 7, 0], [1,
0, 0, 4, 7, 5], [3, 0, 0, 1, 7, 8], [5, 0, 0, 1, 7, 0], [4, 0, 0, 1,
0, 7], [3, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7,
8], [5, 9, 4, 4, 7, 0], [2, 9, 0, 4, 7, 4], [1, 5, 0, 4, 7, 8], [5,
5, 0, 4, 7, 0], [1, 0, 0, 4, 7, 5], [3, 0, 0, 1, 7, 8], [5, 0, 0, 1,
7, 0], [3, 0, 0, 0, 7, 1], [4, 0, 0, 0, 0, 8]]]
_GOAL_FOUND_
[5, 0, 0, 0, 0, 0]
[[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7], [1, 8, 4, 12, 0, 8], [5,
8, 4, 12, 0, 0], [3, 8, 4, 4, 0, 8], [5, 8, 4, 4, 0, 0], [3, 8, 4,
0, 0, 4], [2, 8, 0, 0, 0, 8], [5, 8, 0, 0, 0, 0], [1, 0, 0, 0, 0,
8], [5, 0, 0, 0, 0, 0]]
```

- Η μέθοδος BFS εξερευνά το χώρο καταστάσεων προχωρώντας από επίπεδο σε επίπεδο, αφού επισκέπτεται πρώτα όλους τους κόμβους του τρέχοντος επιπέδου πριν προχωρήσει στο επόμενο, κατι που βλέπουμε ότι έχει επιτευχθεί στον κώδικά μας.
- Η μόνη διαφορά στα αποτελέσματά μας από το αναμενόμενο τρόπο αναζήτησης που φαίνεται από το δένδρο, είναι ότι αντί να έχει στην αρχή του μετώπου αναζήτησης το αριστερότερο μέρος του δένδρου σε κάθε επίπεδο, το έχει στο τέλος. Πχ, ενώ στο προτελευταίο επίπεδο του δένδρου βλέπουμε την κατάσταση [4,0,0,0,0,8] αριστερά ως η πρώτη επιλαχούσα του επιπέδου, στα αποτελέσματα του κώδικα είναι στο τέλος της λίστας μετώπου. Ενώ δηλαδή περιμέναμε ότι σε κάθε φάση θα διαλέγει από το 1 προς 4, διαλέγει αντίστροφα. Όμως κατα τ άλλα η διαδικασία αναζήτησης λειτουργεί σωστά βάσει τη φιλοσοφία του αλγορίθμου της BFS.

Ευριστική μέθοδος

```
Choose searching method (DFS, BFS, FEF or all):
FEF
____BEGIN__SEARCHING__FOR__FEF____
Front:
[[0, 9, 4, 12, 7, 0]]
Queue:
[[[0, 9, 4, 12, 7, 0]]]
Front:
[[1, 1, 4, 12, 7, 8], [3, 9, 4, 4, 7, 8], [4, 9, 4, 12, 0, 7], [2,
9, 0, 12, 7, 4]]
Queue:
```

```
[[[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8]], [[0, 9, 4, 12, 7, 0],
[3, 9, 4, 4, 7, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7]],
[[0, 9, 4, 12, 7, 0], [2, 9, 0, 12, 7, 4]]]
Front:
[[5, 1, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8], [4, 9, 4, 12, 0, 7], [2,
9, 0, 12, 7, 4]]
Queue:
[[[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0]],
[[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8]], [[0, 9, 4, 12, 7, 0], [4,
9, 4, 12, 0, 7]], [[0, 9, 4, 12, 7, 0], [2, 9, 0, 12, 7, 4]]]
Front:
[[3, 1, 4, 4, 7, 8], [4, 1, 4, 12, 0, 7], [2, 1, 0, 12, 7, 4], [1,
0, 4, 12, 7, 1], [3, 9, 4, 4, 7, 8], [4, 9, 4, 12, 0, 7], [2, 9, 0,
12, 7, 4]]
Queue:
[[[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3,
1, 4, 4, 7, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1,
4, 12, 7, 0], [4, 1, 4, 12, 0, 7]], [[0, 9, 4, 12, 7, 0], [1, 1, 4,
12, 7, 8], [5, 1, 4, 12, 7, 0], [2, 1, 0, 12, 7, 4]], [[0, 9, 4, 12,
7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7,
1]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8]], [[0, 9, 4, 12, 7,
0], [4, 9, 4, 12, 0, 7]], [[0, 9, 4, 12, 7, 0], [2, 9, 0, 12, 7,
4]]]
Front:
[[5, 1, 4, 4, 7, 0], [4, 1, 4, 12, 0, 7], [2, 1, 0, 12, 7, 4], [1,
0, 4, 12, 7, 1], [3, 9, 4, 4, 7, 8], [4, 9, 4, 12, 0, 7], [2, 9, 0,
12, 7, 4]]
Queue:
[[[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3,
1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0]], [[0, 9, 4, 12, 7, 0], [1, 1, 4,
12, 7, 8], [5, 1, 4, 12, 7, 0], [4, 1, 4, 12, 0, 7]], [[0, 9, 4, 12,
7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [2, 1, 0, 12, 7,
4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0],
[1, 0, 4, 12, 7, 1]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8]],
[[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7]], [[0, 9, 4, 12, 7, 0],
[2, 9, 0, 12, 7, 4]]]
Front:
[[4, 1, 4, 4, 0, 7], [2, 1, 0, 4, 7, 4], [3, 1, 4, 0, 7, 4], [1, 0,
4, 4, 7, 1], [4, 1, 4, 12, 0, 7], [2, 1, 0, 12, 7, 4], [1, 0, 4, 12,
7, 1], [3, 9, 4, 4, 7, 8], [4, 9, 4, 12, 0, 7], [2, 9, 0, 12, 7, 4]]
Queue:
[[[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3,
1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [4, 1, 4, 4, 0, 7]], [[0, 9, 4,
12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7,
8], [5, 1, 4, 4, 7, 0], [2, 1, 0, 4, 7, 4]], [[0, 9, 4, 12, 7, 0],
[1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1,
4, 4, 7, 0], [3, 1, 4, 0, 7, 4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4,
12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7,
0], [1, 0, 4, 4, 7, 1]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8],
```

[5, 1, 4, 12, 7, 0], [4, 1, 4, 12, 0, 7]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [2, 1, 0, 12, 7, 4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7]], [[0, 9, 4, 12, 7, 0], [2, 9, 0, 12, 7, 4]]]
Front:
[[1, 0, 4, 4, 0, 8], [2, 1, 3, 4, 0, 8], [3, 1, 4, 3, 0, 8], [2, 1, 0, 4, 7, 4], [3, 1, 4, 0, 7, 4], [1, 0, 4, 4, 7, 1], [4, 1, 4, 12, 0, 7], [2, 1, 0, 12, 7, 4], [1, 0, 4, 12, 7, 1], [3, 9, 4, 4, 7, 8], [4, 9, 4, 12, 0, 7], [2, 9, 0, 12, 7, 4]]
Queue:
[[[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [4, 1, 4, 4, 0, 7], [1, 0, 4, 4, 0, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [4, 1, 4, 4, 0, 7], [2, 1, 3, 4, 0, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [4, 1, 4, 4, 0, 7], [3, 1, 4, 3, 0, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [2, 1, 0, 4, 7, 4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [3, 1, 4, 0, 7, 4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [1, 0, 4, 4, 7, 1]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [4, 1, 4, 12, 0, 7]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [2, 1, 0, 12, 7, 4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7]], [[0, 9, 4, 12, 7, 0], [2, 9, 0, 12, 7, 4]]]
Front:
[[5, 0, 4, 4, 0, 0], [2, 1, 3, 4, 0, 8], [3, 1, 4, 3, 0, 8], [2, 1, 0, 4, 7, 4], [3, 1, 4, 0, 7, 4], [1, 0, 4, 4, 7, 1], [4, 1, 4, 12, 0, 7], [2, 1, 0, 12, 7, 4], [1, 0, 4, 12, 7, 1], [3, 9, 4, 4, 7, 8], [4, 9, 4, 12, 0, 7], [2, 9, 0, 12, 7, 4]]
Queue:
[[[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [4, 1, 4, 4, 0, 7], [1, 0, 4, 4, 0, 8], [5, 0, 4, 4, 0, 0]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [4, 1, 4, 4, 0, 7], [2, 1, 3, 4, 0, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [4, 1, 4, 4, 0, 7], [3, 1, 4, 3, 0, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [2, 1, 0, 4, 7, 4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [3, 1, 4, 0, 7, 4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [1, 0, 4, 4, 7, 1]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [4, 1, 4, 12, 0, 7]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [2, 1, 0, 12, 7, 4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7]], [[0, 9, 4, 12, 7, 0], [2, 9, 0, 12, 7, 4]]]
Front:
[[0, 4, 4, 0, 8], [2, 1, 3, 4, 0, 8], [3, 1, 4, 3, 0, 8], [2, 1, 0, 4, 7, 4], [3, 1, 4, 0, 7, 4], [1, 0, 4, 4, 7, 1], [4, 1, 4, 12, 0, 7], [2, 1, 0, 12, 7, 4], [1, 0, 4, 12, 7, 1], [3, 9, 4, 4, 7, 8], [4, 9, 4, 12, 0, 7], [2, 9, 0, 12, 7, 4]]
Queue:
[[[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [4, 1, 4, 4, 0, 7], [1, 0, 4, 4, 0, 8], [5, 0, 4, 4, 0, 0]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [4, 1, 4, 4, 0, 7], [2, 1, 3, 4, 0, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [4, 1, 4, 4, 0, 7], [3, 1, 4, 3, 0, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [2, 1, 0, 4, 7, 4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [3, 1, 4, 0, 7, 4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [1, 0, 4, 4, 7, 1]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4,

12, 7, 0], [4, 1, 4, 12, 0, 7]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [2, 1, 0, 12, 7, 4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7]], [[0, 9, 4, 12, 7, 0], [2, 9, 0, 12, 7, 4]]]
Front:
[[2, 0, 0, 4, 0, 4], [3, 0, 4, 0, 0, 4], [2, 1, 3, 4, 0, 8], [3, 1, 4, 3, 0, 8], [2, 1, 0, 4, 7, 4], [3, 1, 4, 0, 7, 4], [1, 0, 4, 4, 7, 1], [4, 1, 4, 12, 0, 7], [2, 1, 0, 12, 7, 4], [1, 0, 4, 12, 7, 1], [3, 9, 4, 4, 7, 8], [4, 9, 4, 12, 0, 7], [2, 9, 0, 12, 7, 4]]
Queue:
[[[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [4, 1, 4, 4, 0, 7], [1, 0, 4, 4, 0, 8], [5, 0, 4, 4, 0, 0], [2, 0, 0, 4, 0, 4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [4, 1, 4, 4, 0, 7], [1, 0, 4, 4, 0, 8], [5, 0, 4, 4, 0, 0], [3, 0, 4, 0, 0, 4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [4, 1, 4, 4, 0, 7], [2, 1, 3, 4, 0, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [4, 1, 4, 4, 0, 7], [3, 1, 4, 3, 0, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [2, 1, 0, 4, 7, 4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [3, 1, 4, 0, 7, 4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [1, 0, 4, 4, 7, 1]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [4, 1, 4, 12, 0, 7]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [2, 1, 0, 12, 7, 4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [1, 0, 4, 12, 7, 1]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8]], [[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7]], [[0, 9, 4, 12, 7, 0], [2, 9, 0, 12, 7, 4]]]
Front:
[[3, 0, 0, 0, 0, 8], [3, 0, 4, 0, 0, 4], [2, 1, 3, 4, 0, 8], [3, 1, 4, 3, 0, 8], [2, 1, 0, 4, 7, 4], [3, 1, 4, 0, 7, 4], [1, 0, 4, 4, 7, 1], [4, 1, 4, 12, 0, 7], [2, 1, 0, 12, 7, 4], [1, 0, 4, 12, 7, 1], [3, 9, 4, 4, 7, 8], [4, 9, 4, 12, 0, 7], [2, 9, 0, 12, 7, 4]]
Queue:
[[[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [4, 1, 4, 4, 0, 7], [1, 0, 4, 4, 0, 8], [5, 0, 4, 4, 0, 0], [2, 0, 0, 4, 0, 4], [3, 0, 0, 0, 0, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [4, 1, 4, 4, 0, 7], [1, 0, 4, 4, 0, 8], [5, 0, 4, 4, 0, 0], [3, 0, 4, 0, 0, 4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [4, 1, 4, 4, 0, 7], [2, 1, 3, 4, 0, 8]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [4, 1, 4, 4, 0, 7], [3, 1, 4, 3, 0, 8]],

```
[[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3,
1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [2, 1, 0, 4, 7, 4]], [[0, 9, 4,
12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7,
8], [5, 1, 4, 4, 7, 0], [3, 1, 4, 0, 7, 4]], [[0, 9, 4, 12, 7, 0],
[1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3, 1, 4, 4, 7, 8], [5, 1,
4, 4, 7, 0], [1, 0, 4, 4, 7, 1]], [[0, 9, 4, 12, 7, 0], [1, 1, 4,
12, 7, 8], [5, 1, 4, 12, 7, 0], [4, 1, 4, 12, 0, 7]], [[0, 9, 4, 12,
7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [2, 1, 0, 12, 7,
4]], [[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0],
[1, 0, 4, 12, 7, 1]], [[0, 9, 4, 12, 7, 0], [3, 9, 4, 4, 7, 8]],
[[0, 9, 4, 12, 7, 0], [4, 9, 4, 12, 0, 7]], [[0, 9, 4, 12, 7, 0],
[2, 9, 0, 12, 7, 4]]]
_GOAL_FOUND_
[5, 0, 0, 0, 0, 0]
[[0, 9, 4, 12, 7, 0], [1, 1, 4, 12, 7, 8], [5, 1, 4, 12, 7, 0], [3,
1, 4, 4, 7, 8], [5, 1, 4, 4, 7, 0], [4, 1, 4, 4, 0, 7], [1, 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]]


Process finished with exit code 0
```

- Βλέπουμε πως σε κάθε περίπτωση το μέτωπο είναι ορθά ταξινομημένο και γίνεται η αναμενόμενη επιλογή της κατάστασης με τον πιο γεμάτο ανελκυστήρα.
- Παρόμοια με τη μέθοδο DFS, η ουρά αποθηκεύει σωστά υπο-λίστες που υποδεικνύουν τα μονοπάτια από κάθε προηγούμενη κατάσταση στις πιθανές επόμενες.

Σύγκριση των τριών μεθόδων

Λόγω των πολλαπλών ομοιοτήτων μεταξύ της ευριστικής μεθόδου του δημιουργήθηκε στα πλαίσια της επίλυσης του προβλήματος και της μεθόδου DFS, παρατηρούμε ότι και τα αποτελέσματά τους δεν διαφέρουν ιδιαίτερα ως προς την έκταση τους, σε αντίθεση με τη μέθοδο BFS η οποία φάνηκε εκτενής και χρονοβόρα για το πρόβλημά μας.
Επομένως, βλέπουμε πως οι δύο πρώτες υλοποιήσεις απαιτούν λιγότερο χρόνο και κυρίως λιγότερο χώρο. Σε άλλες περιπτώσεις προβλημάτων όμως που η χρήση της BFS είναι κατάλληλη, είναι γνωστό ότι προσφέρει συντομότερα μονοπάτια από αυτά της DFS. Γι’ αυτό και είναι πιο χρήσιμη σε περιπτώσεις όπου η λύση βρίσκεται κοντά στον κόμβο-πηγή, ενώ από την άλλη η DFS όταν βρίσκεται σε μεγάλη απόσταση.