Στο συγκεκριμένο project μελετάται το πρόβλημα του pacman στον δισδιάστατο κόσμο. Συγκεκριμένα, παρατηρούμε την κίνηση του pacman, τα εμπόδια που αντιμετωπίζει και τους περιορισμούς που υφίστανται με στόχο την κατανάλωση όλων των φρούτων που υπάρχουν στον κόσμο του προβλήματος. Κύριο μέρος της εργασίας αποτελούν οι αλγόριθμοι αναζήτησης Πρώτα σε Βάθος(DFS) και Πρώτα σε Πλάτος(BFS). Oι DFS και BFS μελετώνται με παρακολούθηση μετώπου και ουράς αναζήτησης για εύρεση του παραπάνω στόχου.

Πίνακας περιεχομένων

ΕΙΣΑΓΩΓΗ…………………………………………………………..1

ΠΕΡΙΕΧΟΜΕΝΑ…………………………………………………….2

1. ΚΟΣΜΟΣ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ, ΑΡΧΙΚΗ-ΤΕΛΙΚΗ ΚΑΤΑΣΤΑΣΗ………………………………………………………..3-5

2. ΤΕΛΕΣΤΕΣ ΜΕΤΑΒΑΣΗΣ………………………………………..5-6

3. ΧΩΡΟΣ ΚΑΤΑΣΤΑΣΕΩΝ………………………………………....7-13

4. ΣΧΟΛΙΑΣΜΕΝΗ ΚΩΔΙΚΟΠΟΙΗΣΗ(ΤΟΥ ΚΟΣΜΟΥ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ,ΤΩΝ ΤΕΛΕΣΤΩΝ ΜΕΤΑΒΑΣΗΣ ΚΑΙ ΤΗΣ ΣΥΝΑΡΤΗΣΗΣ ΕΥΡΕΣΗΣ ΑΠΟΓΟΝΩΝ)…………......................................................13-18

5. ΑΠΟΤΕΛΕΣΜΑΤΑ ΑΛΓΟΡΙΘΜΟΥ BFS ΜΕ ΠΑΡΑΚΟΛΟΥΘΗΣΗ ΟΥΡΑΣ ΜΟΝΟΠΑΤΙΩΝ……….……………………………………………….18-22

ΣΗΜΕΙΩΣΗ(ΓΙΑ ΤΟΝ ΚΩΔΙΚΑ ΤΗΣ ΕΡΓΑΣΙΑΣ)……………………………………………………………...23

1. Στο συγκεκριμένο σημείο θα περιγραφεί ο κόσμος του προβλήματος και εν συνεχεία η αρχική και η τελική του κατάσταση.

Ως κόσμος του προβλήματος ορίζεται ένα υποσύνολο του πραγματικού κόσμου. Αναλυτικότερα, περιέχει τα αντικείμενα του προβλήματος, τις ιδιότητες τους και τις σχέσεις που τα συνδέουν.

Συγκεκριμένα, τα αντικείμενα του προβλήματος του pacman είναι τα εξής:

Α) pacman

Β) φρούτα

Γ) τοίχος

Δ) 2d λίστα

Ε) τοιχώματα της λίστας.

Παράλληλα, ως κελί ορίζεται κάθε λίστα 2 θέσεων από τις 16 που υπάρχουν στην 2d λίστα του προβλήματος.

[[[' ', ' '], ['w', 'w'], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', 'f'], ['', ' '], ['', ' '], [' ', ' ']],

[[' ', ' '], ['', ' '], ['p', ''], [' ', ' ']]]

Μπλε: κελί της λίστας

(τυχαίο στιγμιότυπο από την εξέλιξη του προβλήματος)

Τα παραπάνω αντικείμενα έχουν ορισμένες ιδιότητες και συνδέονται μέσω κάποιων σχέσεων. Όσον αφορά τις ιδιότητες των αντικειμένων, ο pacman τρώει το φρούτο αν βρεθούν στο ίδιο κελί. Τα φρούτα, από την άλλη πλευρά, είναι ακίνητα. Ωστόσο, είναι πολύ σημαντικά για την εξέλιξη του προβλήματος, καθώς αποτελούν στόχο για την κίνηση του pacman. Ο μοναδικός τοίχος της λίστας είναι και εκείνος ακίνητος, μη επιτρέποντας στον pacman να τον διαπεράσει , καθυστερώντας τον και αυξάνοντας την πολυπλοκότητα του εκάστοτε αλγορίθμου αναζήτησης. Επίσης, η πολυαναφερφείσα μέχρι στιγμής λίστα διαθέτει χώρο 16 θέσεων για τον pacman, χώρο 16 θέσεων για τα φρούτα και μίας θέσης(ένα ολόκληρο κελί) για τον μοναδικό τοίχο. Τέλος, τα τοιχώματα της λίστας αποτελούν το διαχωριστικό μεταξύ της λίστας και του πραγματικού κόσμου. Εμποδίζουν την κίνηση του pacman, αφού έχει τη δυνατότητα να τα προσεγγίσει, χωρίς όμως να τα υπερβεί.

Εστιάζοντας στις σχέσεις που διέπουν τα αντικείμενα του προβλήματος, ο pacman βρίσκεται(και μπορεί να κινηθεί) στην αριστερή θέση ενός από τα 16 κελιά της λίστας. Ακόμη, τα φρούτα βρίσκονται στη δεξιά θέση ενός ή και παραπάνω κελιών της λίστας(ανάλογα με τον αρχικό αριθμό τους). Ο τοίχος καταλαμβάνει και τις 2 θέσεις ενός κελιού. Τα τοιχώματα, είναι ευκόλως κατανοητό πως βρίσκονται περιμετρικά της λίστας.

*Αρχική κατάσταση του προβλήματος*:

Αξίζει να τονιστεί το γεγονός ότι στο πρόβλημα(μέσω του κώδικα) δύναται να υπάρχει αυστηρά 1 pacman. Ακόμη, υπάρχουν 1 έως και 5 φρούτα, τυχαίος αριθμός με χρήση της βιβλιοθήκης random και της συνάρτησης της randrange, και μοναδικός τοίχος. Καταρχήν , ο τοίχος τοποθετείται πρώτος στη λίστα καταλαμβάνοντας 1 ολόκληρο κελί , δηλαδή τόσο την αριστερή , όσο και τη δεξιά του θέση. Στη συνέχεια, τοποθετείται ο pacman σε τυχαία θέση(στο αριστερό της <<υποκελί>> , δηλαδή τη θέση 0: in\_state[random\_position1][random\_position2][0] = **'p'**

,όπουin\_state η λίστα, random\_position1 τυχαία γραμμή της λίστας και random\_position2 τυχαία στήλη της) αφού πρώτα γίνει έλεγχος τέτοιος ώστε να μην υπάρχει ο τοίχος στη συγκεκριμένη θέση. Έπειτα, τοποθετούνται τα φρούτα σε τυχαίες θέσεις(στη θέση 1 ενός από τα 16 κελιά της λίστας το καθένα ,

in\_state[random\_position1][random\_position2][0] = **'f'**) , τσεκάροντας νωρίτερα την μη ύπαρξη τοίχου σε ένα από τα διαθέσιμα για την τοποθέτηση φρούτων κελιά. Δεν υπάρχει περιορισμός στην αρχική τοποθέτηση pacman και φρούτου στο ίδιο κελί. Ακόμη, τα φρούτα , ο pacman και ο τοίχος μπορούν να βρεθούν αρχικά και σε διαχωριστικό της λίστας. Γίνονται τόσοι έλεγχοι όσα και τα φρούτα που υπάρχουν στην αρχική κατάσταση του προβλήματος.

*Τελική κατάσταση του προβλήματος*: Ως τελική κατάσταση του προβλήματος χαρακτηρίζεται η κατάσταση κατά την οποία έχει εκπληρωθεί ο σκοπός του αλγορίθμου αναζήτησης. Δηλαδή , έχουν φαγωθεί όλα τα φρούτα από τον pacman(έχουν βρεθεί στο ίδιο κελί σε μια συγκεκριμένη στιγμή της εξέλιξης του κόσμου του προβλήματος) και ο pacman βρίσκεται στο κελί(στην αριστερή πάντα θέση) που βρισκόταν το τελευταίο φρούτο που υπήρξε στη λίστα. Ο τοίχος, από την άλλη πλευρά, βρίσκεται ακριβώς στην αρχική του θέση. Εν κατακλείδι, δεν είναι δυνατή η εκ των προτέρων γνώση της τελικής κατάστασης του προβλήματος , δηλαδή της τελευταίας θέσης που θα βρεθεί ο pacman. Το παραπάνω συμβαίνει εξαιτίας του γεγονότος ότι η αρχική κατάσταση του προβλήματος προκύπτει μετά από τυχαία τοποθέτηση των οντοτήτων πάνω στη λίστα.

Παρακάτω απεικονίζεται ένα παράδειγμα αρχικής και τελικής κατάστασης του προβλήματος , που επιλύθηκε με χρήση DFS αλγορίθμου αναζήτησης με επέκταση του μετώπου(front) αναζήτησης.

Αρχική κατάσταση του προβλήματος:

[[[[' ', ' '], ['**w**', '**w**'], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', **'f**'], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], ['p', '**f**'], [' ', ' ']]]

Τελική κατάσταση του προβλήματος:

[[[' ', ' '], ['**w**', '**w**'], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[['p', ''], ['', ' '], ['', ' '], [' ', ' ']],

[['', ' '], ['', ' '], ['', ''], [' ', ' ']]]

1. Ιδιαίτερα σημαντικό κομμάτι του προβλήματος του pacman και γενικότερα των προβλημάτων στα οποία επεμβαίνουμε με αλγόριθμους αναζήτησης αποτελούν οι τελεστές μετάβασης. Αφορούν τις ενέργειες που λαμβάνουν χώρα κατά τη διάρκεια της επίλυσης του προβλήματος , ώστε να οδηγηθούμε από την εκάστοτε κατάσταση στην αμέσως επόμενη της. Για κάθε τελεστή μετάβασης διενεργούμε ελέγχους , με σκοπό να εφαρμοστεί το μοντέλο μετάβασης λαμβάνοντας υπόψιν του ο αλγόριθμους όλες τις προϋποθέσεις που οδηγούν στην ομαλή μετάβαση στην επόμενη κατάσταση.

Όσον αφορά το πρόβλημα του pacman,και ιδιαίτερα το ίδιο το pacman, στο δισδιάστατο χώρο ορίζονται 5 τελεστές μετάβασης:

1) Κατανάλωση φρούτου

2) Κίνηση προς τα δεξιά

3) Κίνηση προς τα αριστερά

4) Κίνηση προς τα πάνω

5) Κίνηση προς τα κάτω

Αναλυτικότερα, αν pacman και φρούτο βρεθούν στο ίδιο κελί(ο μοναδικός έλεγχος που πραγματοποιείται), τότε ο pacman θα << φάει >> το φρούτο. Εν συνεχεία, η επόμενη κατάσταση θα βρει τον pacman στην ίδια θέση

([**'p'** , **'f'**] **->** [**'p'** , **''**]).

Ενώ, το φρούτο θα έχει εξαφανιστεί , μειώνοντας κατά ένα τον αρχικό αριθμό των υπάρχοντών φρούτων και θα έχει γίνει έτσι ένα σημαντικό βήμα προς την επίλυση του προβλήματος. Δεν υπάρχει κάποια συγκεκριμένη προϋπόθεση που οφείλει να καλύπτεται ώστε να πραγματοποιηθεί ο πρώτος τελεστής μετάβασης.

Από την άλλη πλευρά , όσον αφορά τους τελεστές μετάβασης που σηματοδοτούν κίνηση σε κάθε από τις 4 κατευθύνσεις του δισδιάστατου επιπέδου υπάρχουν συγκεκριμένες προϋποθέσεις για να οδηγηθούμε στην επόμενη κατάσταση.

Χαρακτηριστικά, **απαγορεύεται** η κίνηση προς τα δεξιά όταν ανιχνευτεί κατά τη διάσχιση της λίστας ότι:  
 α) Ο pacman βρίσκεται στην τελευταία(δεξιότερη) στήλη της λίστας.  
 β) Υπάρχει τοίχος ακριβώς μία στήλη δεξιότερα από την τρέχουσα θέση του pacman.

Αν δεν απορριφθεί η κίνηση προς τα δεξιά, περνώντας από τον έλεγχο, το κελί που βρισκόταν ο pacman αδειάζει με τη φυγή του. Ο ίδιος, μεταφέρεται στο αμέσως δεξιότερο κελί της λίστας(πάντα στην αριστερά θέση από τις δύο του κελιού).

Επιπροσθέτως , **απαγορεύεται** η κίνηση προς τα αριστερά όταν ανιχνευτεί κατά τη διάσχιση της λίστας ότι:  
 α) Ο pacman βρίσκεται στην πρώτη(αριστερότερη) στήλη της λίστας(σε οποιαδήποτε από τις γραμμές τις λίστας, καθώς δε επηρεάζεται η κίνηση στον άξονα x).  
 β) Υπάρχει τοίχος ακριβώς μία στήλη αριστερότερα από την τρέχουσα θέση του pacman.

Αν δεν απορριφθεί η κίνηση προς τα αριστερά, περνώντας από τον έλεγχο, το κελί που βρισκόταν ο pacman αδειάζει με τη φυγή του. Ο ίδιος, μεταφέρεται στο αμέσως αριστερότερο κελί της λίστας(πάντα στην αριστερά θέση από τις δύο του κελιού).

Ακόμη , **απαγορεύεται** η κίνηση προς τα πάνω όταν ανιχνευτεί κατά τη διάσχιση της λίστας ότι:  
 α) Ο pacman βρίσκεται στην πρώτη(υψηλότερη,[0]) γραμμή της λίστας(σε οποιαδήποτε από τις στήλες τις λίστας, καθώς δε επηρεάζεται η κίνηση στον άξονα y).  
 β) Υπάρχει τοίχος ακριβώς μία γραμμή πιο πάνω από την τρέχουσα θέση του pacman.

Αν δεν απορριφθεί η κίνηση προς τα πάνω, περνώντας από τον έλεγχο, το κελί που βρισκόταν ο pacman αδειάζει με τη φυγή του. Ο ίδιος, μεταφέρεται στο αμέσως υψηλότερο κελί της λίστας(πάντα στην αριστερά θέση από τις δύο του κελιού).

Ακόμη , **απαγορεύεται** η κίνηση προς τα κάτω όταν ανιχνευτεί κατά τη διάσχιση της λίστας ότι:  
 α) Ο pacman βρίσκεται στην τελευταία(χαμηλότερη) γραμμή της λίστας.

β) Υπάρχει τοίχος ακριβώς μία γραμμή πιο κάτω από την τρέχουσα θέση του pacman.

Αν δεν απορριφθεί η κίνηση προς τα κάτω, περνώντας από τον έλεγχο, το κελί που βρισκόταν ο pacman αδειάζει με τη φυγή του. Ο ίδιος, μεταφέρεται στο αμέσως χαμηλότερο κελί της λίστας(πάντα στην αριστερά θέση από τις δύο του κελιού).

1. Ο χώρος καταστάσεων του προβλήματος περιγράφει όλες τις πιθανές διατάξεις του pacman, των φρούτων και του τοίχου μέσα στη 2d λίστα. Αποτελείται από την Αρχική Κατάσταση του προβλήματος, τις Ενέργειες που πραγματοποιούνται και το Μοντέλο Μετάβασης. Όπως είναι κατανοητό, πρέπει να ληφθούν σοβαρά οι αναφερόμενες παραπάνω απαγορεύσεις που σχετίζονται με την κίνηση του pacman. Δεδομένης μιας κατάστασης και μιας ενέργειας άλλωστε προκύπτει η επόμενη κατάσταση. Στο σύνολο των καταστάσεων ο τοίχος παραμένει στην ίδια θέση. Το ίδιο ισχύει και για τα φρούτα μέχρι και τη στιγμή που θα φαγωθούν, οπότε ένα-ένα εξαφανίζονται σε επόμενες καταστάσεις. Ο pacman διαρκώς κινείται , διαμορφώνοντας νέες καταστάσεις και αυξάνοντας το χώρο καταστάσεων(σε αρκετά μεγάλο βαθμό σε ορισμένα παραδείγματα με συγκεκριμένο αλγόριθμο αναζήτησης). Κατά τη διάρκεια εκτέλεσης των αλγορίθμων αναζήτησης αναπτύσσεται το δένδρο αναζήτησης. Η λύση, τέλος, αποτελεί ακολουθία καταστάσεων που ανήκουν στο χώρο καταστάσεων στις οποίες μπορεί να βρεθεί διαδοχικά το πρόβλημα, από την τελική έως και την τελική κατάσταση.

Για την περιγραφή του χώρου καταστάσεων του προβλήματος επιλέχθηκαν 2 παραδείγματα σε μορφή λιστών. Στο πρώτο παράδειγμα έχει χρησιμοποιηθεί ο αλγόριθμος *Πρώτα σε Βάθος(DFS) με ανάπτυξη μετώπου*. Ενώ, στο δεύτερο παράδειγμα έχει χρησιμοποιηθεί ο αλγόριθμος *Πρώτα σε Βάθος(DFS) με ανάπτυξη μετώπου και παράλληλη παρακολούθηση της ουράς των μονοπατιών*. Και τα δύο αποτελούν τρέξιμο του αντίστοιχου κώδικα που αναπτύχθηκε για την Εργασία και μπορούν να βρεθούν σε ξεχωριστά αρχεία .txt στο .zip αρχείο(γίνονται κατανοητά σε κάθε βήμα του αλγορίθμου το μέτωπο, η ουρά και το κλειστό σύνολο).

Παράδειγμα 1: αρχείο *DFS\_front\_run.txt*

Παράδειγμα 2: αρχείο *DFS\_front\_queue\_run.txt*

***Παράδειγμα 1***

**Αρχική κατάσταση:**

[[[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], ['p', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', 'f'], [' ', ' ']],

[[' ', ' '], [' ', ' '], ['w', 'w'], [' ', ' ']]]]

**1ο επίπεδο χώρου καταστάσεων:**

[[[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], ['', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], ['p', 'f'], [' ', ' ']],

[[' ', ' '], [' ', ' '], ['w', 'w'], [' ', ' ']]]

[[[' ', ' '], [' ', ' '], ['p', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], ['', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', 'f'], [' ', ' ']],

[[' ', ' '], [' ', ' '], ['w', 'w'], [' ', ' ']]]

[[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], ['p', ' '], ['', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', 'f'], [' ', ' ']],

[[' ', ' '], [' ', ' '], ['w', 'w'], [' ', ' ']]]

[[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], ['', ' '], ['p', ' ']],

[[' ', ' '], [' ', ' '], [' ', 'f'], [' ', ' ']],

[[' ', ' '], [' ', ' '], ['w', 'w'], [' ', ' ']]]]

**2ο επίπεδο χώρου καταστάσεων-τελική κατάσταση:**

[[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], ['', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], ['p', ''], [' ', ' ']],

[[' ', ' '], [' ', ' '], ['w', 'w'], [' ', ' ']]]

***Παράδειγμα 2***

**Αρχική κατάσταση:**

[[[[' ', ' '], [' ', ' '], [' ', ' '], ['p', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', 'f']]]]

**1ο επίπεδο χώρου καταστάσεων:**

[[[[' ', ' '], [' ', ' '], [' ', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['p', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', 'f']]]

[[[' ', ' '], [' ', ' '], ['p', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', 'f']]]]

**2ο επίπεδο χώρου καταστάσεων:**

[[[[' ', ' '], [' ', ' '], [' ', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['p', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', 'f']]]

[[[' ', ' '], [' ', ' '], [' ', ' '], ['p', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', 'f']]]

[[[' ', ' '], [' ', ' '], [' ', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], ['p', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', 'f']]]

[[[' ', ' '], [' ', ' '], ['p', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', 'f']]]]

**3ο επίπεδο χώρου καταστάσεων:**

[[[[' ', ' '], [' ', ' '], [' ', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], ['p', 'f']]]

[[[' ', ' '], [' ', ' '], [' ', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['p', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', 'f']]]

[[[' ', ' '], [' ', ' '], [' ', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], ['p', ' '], ['', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', 'f']]]

[[[' ', ' '], [' ', ' '], [' ', ' '], ['p', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', 'f']]]

[[[' ', ' '], [' ', ' '], [' ', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], ['p', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', 'f']]]

[[[' ', ' '], [' ', ' '], ['p', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', 'f']]]]

**4ο επίπεδο χώρου καταστάσεων-τελική κατάσταση:**

[[[[' ', ' '], [' ', ' '], [' ', ' '], ['p', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', 'f']]]

[[[' ', ' '], [' ', ' '], [' ', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['p', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', 'f']]]

[[[' ', ' '], [' ', ' '], [' ', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['p', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', 'f']]]

[[[' ', ' '], [' ', ' '], [' ', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], ['p', 'f']]]

[[[' ', ' '], [' ', ' '], [' ', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], ['p', '']]]]

Επιπροσθέτως ισχύει ότι,

**row** 0,1,2,3 }

**column** 0,1,2,3 }

**element** 0,1 }

Χώρος Καταστάσεων:

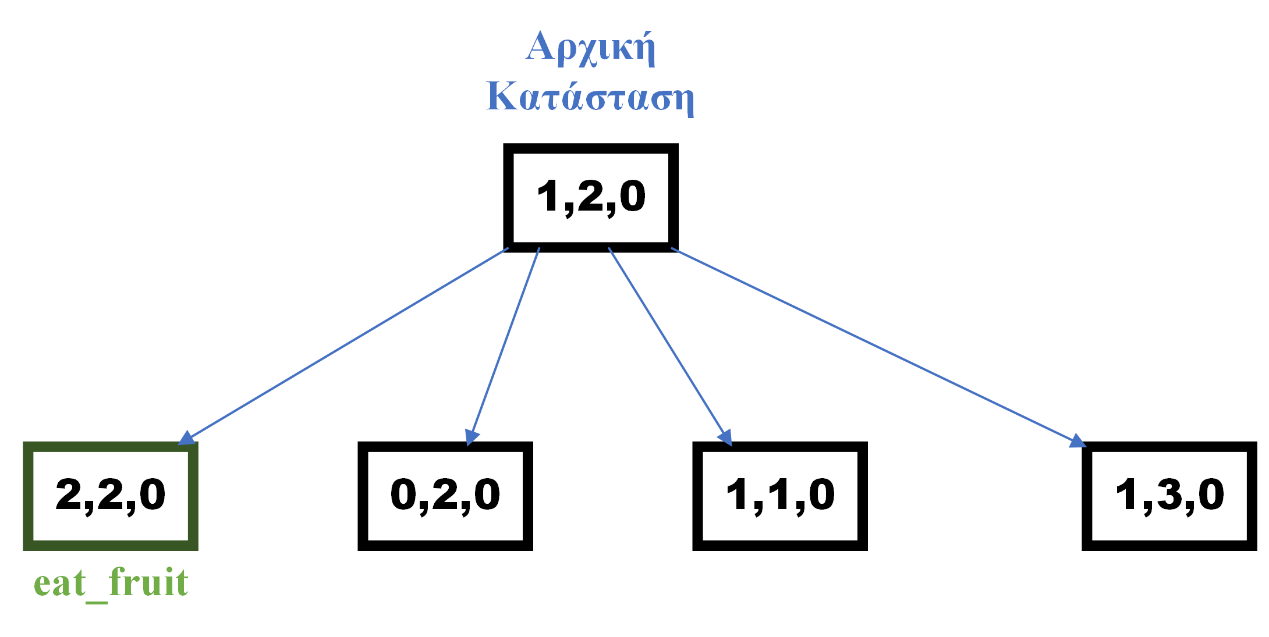
**wall** = (row, column, element)}

**pacman** = (row, column, element) : element=0) ≠ wall}

**fruit** = ((row, column, element) : element=1) ≠ wall }

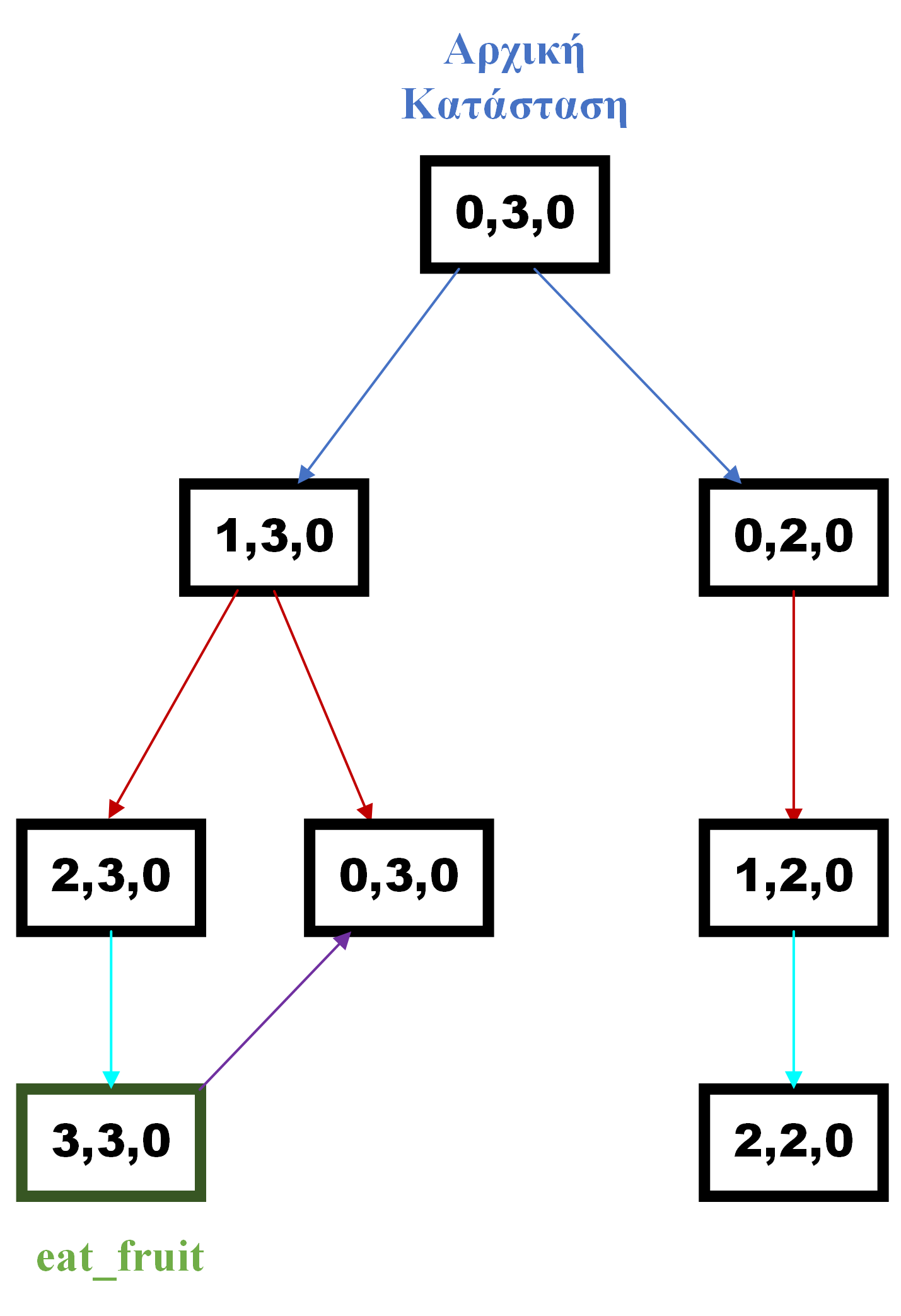
Ανάλογα με την Αρχική κατάσταση του προβλήματος(δημιουργείται με τυχαίο τρόπο) διαμορφώνεται ο Γράφος του χώρου καταστάσεων:

Όσον αφορά το **Παράδειγμα 1 – Χώρος καταστάσεων του pacman**:



Όταν ο pacman φάει το τελευταίο φρούτο, μένει μόνος του στο κελί και έτσι έχει παραχθεί η τελική κατάσταση του προβλήματος.

Όσον αφορά το **Παράδειγμα 2 - Χώρος καταστάσεων του pacman**:



1. Σε αυτό το σημείο θα αναλυθεί ο κώδικας που δημιουργήθηκε σε python για τον **κόσμο του προβλήματος**, τους **τελεστές μετάβασης** και τη **συνάρτηση εύρεσης απογόνων**. Προτιμήθηκε ο χωρισμός της ανάλυσης σε 3 διαφορετικές ενότητες. Αναλυτικότερα, θα αναλυθούν τα βήματα Β) , Γ) και Δ) όπως ακριβώς παρουσιάζονται στην εκφώνηση της Εργασίας. Δηλαδή, πρώτα θα αναλυθεί ο κώδικας(περισσότερο για τα σημεία που αναφέρθηκαν αρχικά) για τον αλγόριθμο αναζήτησης Πρώτα σε Βάθος(DFS) με παρακολούθηση μετώπου. Έπειτα, ο αλγόριθμος αλγόριθμο αναζήτησης Πρώτα σε Βάθος(DFS) με παρακολούθηση μετώπου και ουράς αναζήτησης. Τέλος, ο αλγόριθμος αλγόριθμο αναζήτησης Πρώτα σε Πλάτος(BFS) με παρακολούθηση μετώπου και ουράς αναζήτησης.

**Βήμα Β) - DFS με παρακολούθηση μετώπου**

Καταρχάς, για την ανάλυση θα χρησιμοποιηθεί σειρά συναρτήσεων με παρόμοιο τρόπο με τον οποίο καλούνται. Αρχικά , η συνάρτηση main() καλεί την find\_solution(make\_front(initial\_state), [], method). Η συνάρτηση δεν περιέχει το όρισμα goal ,καθώς χρησιμοποιείται παραλλαγμένη, λόγω της ύπαρξης τυχαίας αρχικής κατάστασης. Δεν υπάρχει δηλαδή η εκ των προτέρων εύρεση της κατάστασης-στόχου.

Καλώντας την find\_solution οδηγούμαστε σε 4 περιπτώσεις:

1) Δεν υπάρχει μέτωπο αναζήτησης. Οδηγούμαστε σε αποτυχία- τέλος του προγράμματος.

2) is\_goal\_state(front[0]).Η συγκεκριμένη συνάρτηση επιστρέφει 1(επιτυχία) όταν κατά τη διάσχιση της λίστας δεν εντοπίζονται φρούτα , δηλαδή έχουν φαγωθεί από τον pacman.Παίρνει ως όρισμα τη γονική κατάσταση του μετώπου αναζήτησης. Οδηγούμαστε σε επιτυχία- τέλος του προγράμματος.

3) front[0] in closed. Σε αυτή την περίπτωση δημιουργούμε ,μέσω της deepcopy, αντίγραφο του μετώπου. Ακολουθεί αναδρομική κλήση της find\_solution με όρισμα το νέο μέτωπο, χωρίς να περιέχει δηλαδή τη γονική κατάσταση του(αντίγραφου του) μετώπου αναζήτησης.

4) Βάζουμε την κατάσταση-γονέα στο κλειστό σύνολο(2ο όρισμα της find\_solution). Εκεί περιέχονται οι εκάστοτε καταστάσεις γονείς. Έπειτα, δημιουργούμε αντίγραφο του μετώπου και οδηγούμαστε στη συνάρτηση expand\_front. Μετά από τον απαραίτητο έλεγχο του είδους της τυφλής μεθόδου αναζήτησης που θα χρησιμοποιήσουμε(DFS), με τη συνάρτηση expand\_front που παίρνει ως παραμέτρους το μέτωπο και το πρώτο στοιχείο του μετώπου (node) φτιάχνουμε παιδιά – επεκτείνουμε το μέτωπο και τα τοποθετούμε στην αρχή του μετώπου(insert).

**for** child **in find\_children**(node):  
 front.insert(0, child)

Δέχεται ως όρισμα ένα state.Χρησιμοποιεί τη λίστα children.Δημιουργεί και εξετάζει 5 αντίγραφα του state-node ,ένα για κάθε **τελεστή μετάβασης**, μέσω της copy.deepcopy(state):

Α) right\_state: Τοποθετείται ως όρισμα της function move\_right.Στη συγκριμένη γίνεται έλεγχος(**can\_move\_right**) σχετικός με τη θέση του state. Εάν βρισκόμαστε στη δεξιότερη στήλη της εκάστοτε γραμμής ή υπάρχει στην αμέσως δεξιότερη στήλη της ίδιας γραμμής τοίχος(‘w’), δε μπορούμε να κινηθούμε δεξιά και επιστρέφεται None. Αντίθετα, μέσω μιας επανάληψης που διατρέχει όλες τις θέσεις , ο pacman(‘p’) κινείται δεξιότερα(move\_right), δηλαδή το state[i][j][0] = **'p'**

μετατρέπεται σε state[i][j + 1][0] = **'p'**. Γίνεται return το state από τη move\_right.

Τα παραπάνω αποθηκεύονται στην child\_right:

child\_right = move\_right(right\_state) στη **συνάρτηση εύρεσης απογόνων**.

Εάν το child\_right δεν είναι κενό(is not None) το τοποθετούμε στο τέλος(append) της προαναφερθείσας λίστας.

Τα παραπάνω ισχύουν και για τα υπόλοιπα 4 αντίγραφα που εξετάζονται:

Β) left\_state

Γ) up\_state

Δ) down\_state

Ε) eat\_state

Στο τέλος της **find\_children** επιστρέφουμε τη λίστα **children.**

Σε αυτό το σημείο έχουμε επεκτείνει το μέτωπο αναζήτησης. Επιπροσθέτως, δημιουργούμε αντίγραφο του κλειστού συνόλου(closed\_copy). Στη συνέχεια, ξεκινάει η αναδρομική κλήση της find\_solution.

Δέχεται τα εξής ορίσματα:

1. front\_children, ώπου front\_children = expand\_front(front\_copy, method) με την expand\_front να αποτελεί διαδικασία που αναλύσαμε παραπάνω.

2. closed\_copy

3. method , στη συγκεκριμένη περίπτωση η τυφλή μέθοδος αναζήτησης DFS. Με λίγα λόγια, επεκτείνεται πάντα ο βαθύτερος κόμβος. Οι κόμβοι που έχουν ελεγχθεί αφαιρούνται από το μέτωπο. Δίνεται η δυνατότητα της χρονικής οπισθοδρόμησης, με περίπτωση που έχει ελεγχθεί όλο το βάθος του δένδρου αναζήτησης.

Εν κατακλείδι, εάν το μέτωπο γεμίσει με παιδιά τα οποία έχουμε επισκεφθεί , τότε το αδειάζουμε και οδηγούμαστε στο τέλος του προβλήματος του pacman.

**Βήμα Γ) - DFS με παρακολούθηση ουράς μονοπατιών**

Διαφοροποιήσεις- προσθήκες στον κώδικα DFS με παρακολούθηση μετώπου , ώστε να υποστηρίζεται και η παρακολούθηση της ουράς των μονοπατιών που δημιουργεί ο DFS κατά την επίλυση:

1) συνάρτηση make\_queue , για αρχικοποίηση της ουράς.

2) συνάρτηση extend\_queue , για επέκταση της ουράς.

Σε αυτό το σημείο έχουμε φτιάξει την ουρά μέσω της make\_queue επιστρέφοντας το [[state]]. Αρχικά, αφαιρούμε από την ουρά το queue[0] και το εκχωρούμε στη λίστα node.

Παίρνουμε το queue[0] από τον τελευταίο(κάτω-κάτω) κόμβο και το εκχωρούμε ως όρισμα στη συνάρτηση **find\_children**.

Έπειτα σαρώνουμε τη λίστα **children** που δημιουργεί η **find\_children**, η οποία περιέχει τα παιδιά που δημιουργούνται με βάση τις επιτρεπόμενες κινήσεις που γίνονται σε κάθε κατάσταση του προβλήματος. Φτιάχνουμε αντίγραφο της λίστας node και προσαρτούμε κάθε παιδί(δηλαδή την τρέχουσα κατάσταση του node) στο τέλος της διαδρομής.

Μετά κάνουμε insert το path στην queue. Έτσι, γράφοντας όλες τις διαδρομές από τη ρίζα(παρελθόντες κόμβοι) στο τέλος της ουράς σε κάθε στάδιο.

3) find\_solution(make\_front(initial\_state), make\_queue(initial\_state) , [], method)

Όπως στο βήμα Β) έτσι και στο Γ) υπάρχουν 3 υποπεριπτώσεις της συνάρτησης find\_solution:

1. Αποτυχία(Δεν υπάρχει μέτωπο αναζήτησης)

2. Επιτυχία(is\_goal\_state(front[0])). Τυπώνεται το queue[0], μέσω της pprint, που παρουσιάζει τη λίστα με όμορφο και εύκολα αναγνώσιμο τρόπο. Το queue[0] αποτελεί το μονοπάτι από τη ρίζα – αρχική κατάσταση έως και την κατάσταση – στόχο. Αφού χρησιμοποιείται κλειστό σύνολο στον αλγόριθμο και ο χώρος αναζήτησης είναι πεπερασμένος , τότε θα βρεθεί λύση(πλήρης αλγόριθμος).

3. Υπάρχει η αρχή του μετώπου στο κλειστό σύνολο, στο οποί αποθηκεύονται οι καταστάσεις - γονείς. Δημιουργούμε αντίγραφα του μετώπου και της ουράς αναζήτησης(new\_front , new\_queue). Και από τα 2 αντίγραφα αφαιρώ(pop(0)) τη γονική κατάσταση. Έπειτα καλούμε(αναδρομική κλήση) τη find\_solution:

find\_solution(new\_front , new\_queue , closed, method)

Μετά τη νέα κληση της συνάρτησης κλειδί για το πρόβλημα ο αλγόριθμος συνεχίζει να ψάχνει τη λύση και ο pacman να τρώει τα φρούτα , αποφεύγοντας τον τοίχο και τα περιθώρια των 4 κατευθύνσεων.

4. Τέλος, εάν δε βρισκόμαστε σε καμία από τις 3 προαναφερθείσες καταστάσεις, βάζουμε την κατάσταση-γονέα του μετώπου στο κλειστό σύνολο. Επιπροσθέτως, δημιουργούμε αντίγραφα του front και της queue , μέσω της deepcopy. Έπειτα, καλούμε τις συναρτήσεις expand\_front και extend\_queue , με σκοπό την επέκταση του μετώπου και της ουράς αναζήτησης.

Όσον αφορά το μέτωπο, φτιάχνουμε παιδιά και τα τοποθετουμε στην αρχή του μετώπου. Σε αυτό το σημείο χρησιμοποιούμε τη συνάρτηση **find\_children**. Δημιουργούμε τους απαραίτητους ελέγχους και στη συνέχεια της κινήσεις που φανερώνουν οι 5 τελεστές μετάβασης(δημιουργία 5 αντιγράφων όπως αναλύθηκε σε προηγούμενο στάδιο). Επιστρέφεται η λίστα **children** και τα νέα παιδιά τοποθετούνται στην αρχή του μετώπου. Με αυτό τον τρόπο λειτουργεί η συνάρτηση expand\_front. Όσον αφορά στην ουρά αναζήτησης , αφαιρούμε και από εκείνη την κατάσταση-γονέα και την τοποθετούμε στον node. Η **find\_children** λειτουργεί όπως και παραπάνω. Στη συνέχεια δημιουργούμε αντίγραφο του node(path) και το αποθηκεύουμε στο τέλος του αντιγράφου του κάθε όρου που περιέχεται στη λίστα **children**(append). Επιπροσθέτως, τοποθετούμε στην αρχή του αντιγράφου της ουράς αναζήτησης(queue\_copy) το path. Στο τέλος της συνάρτησης extend\_queue γίνεται return το queue\_copy. Έτσι, σε αυτό το σημείο έχει επεκταθεί η ουρά αναζήτησης του DFS.

Λίγο πριν το τέλος της τέταρτης περίπτωσης της find\_solution δημιουργούμε αντίγραφο και για το κλειστό σύνολο. Τέλος καλούμε αναδρομικά τη find\_solution:

find\_solution(front\_children, queue\_children, closed\_copy, method)

,όπου front\_children = expand\_front(front\_copy, method) ,

closed\_copy = copy.deepcopy(closed) και   
 queue\_children = extend\_queue(queue\_copy, method).

Σημείωση: Οι συναρτήσεις που δεν αναφέρθηκαν παραμένουν ίδιες με τον κώδικα **DFS με παρακολούθηση μετώπου** , και χρησιμοποιούνται κανονικά στον κώδικα του **DFS με παρακολούθηση ουράς μονοπατιών**.

**Βήμα Δ) - BFS με παρακολούθηση ουράς μονοπατιών**

Κατά την Πρώτα σε Πλάτος αναζήτηση(BFS) επεκτείνεται αρχικά ο ριζικός κόμβος, έπειτα οι διάδοχοι του και στη συνέχεια οι διάδοχοι των διαδόχων. Βρίσκει την καλύτερη λύση, είναι πλήρης. Ωστόσο, το μέτωπο της αναζήτησης μεγαλώνει πολύ σε μέγεθος.

Όσον αφορά τη λειτουργία της BFS, δημιουργεί καταστάσεις-παιδιά και τα τοποθετεί στο τέλος του μετώπου αναζήτησης(append). Με αυτό τον τρόπο και έχοντας αφαιρέσει την κατάσταση-γονέα από το παλαιό μέτωπο, δημιουργείται το νέο μέτωπο αναζήτησης. Η κατάσταση-γονέας τοποθετείται στο κελιστό σύνολο.

Επιπροσθέτως, η δεύτερη διαφορά της από τη DFS είναι η διαχείριση της ουράς αναζήτησης. Χαρακτηριστικά, μέχρι να βρεθεί η κατάσταση-στόχος , το εκάστοτε μονοπάτι για επέκταση τοποθετείται στο τέλος της ουράς αναζήτησης.

Αξίζει να τονιστεί το γεγονός ότι ο κώδικας της BFS διαφοροποιείται από τον κώδικα της DFS σε δύο σημεία. Αναλυτικά, οι συναρτήσεις που χρησιμοποιούνται για επέκταση του μετώπου(expand\_front) και της ουράς αναζήτησης(extend\_queue) δέχθηκα ορισμένες διαφοροποιήσεις.

Στην expand\_front:

Το front.insert(0, child) μετατρέπεται σε front.append(child).

Στην extend\_queue:

Το queue\_copy.insert(0, path) μετατρέπεται σε queue\_copy.append(path)

.

Οι παραπάνω αλλαγές οδηγούν τα νέα παιδιά στο τέλος του μετώπου και της ουράς αναζήτησης , όπως ορίζει ο αλγόριθμος αναζήτησης BFS.

1. Τέλος, θα παρουσιαστούν τα αποτελέσματα του Βήματος Δ της Εργασίας. Δηλαδή θα εκτεθούν τα μονοπάτια που οδηγούν σε λύση από την αρχική κατάσταση μέχρι και την τελική κατάσταση, με χρήση του αλγορίθμου Πρώτα σε Πλάτος αναζήτησης(BFS) με παρακολούθηση της ουράς μονοπατιών για δύο διαφορετικές αρχικές καταστάσεις.

Παράδειγμα 1: αρχείο *BFS\_front\_queue\_run\_1.txt*

Παράδειγμα 2: αρχείο *BFS\_front\_queue\_run\_2.txt*

Στα αρχεία υπάρχουν αναλυτικά τα αποτελέσματα, με χρήση μετώπου, ουράς αναζήτησης και του κλειστού συνόλου.

**Αρχική κατάσταση 1:**

[[[' ', ' '], [' ', 'f'], [' ', ' '], ['p', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']]]

**Μονοπάτι:**

[[[[' ', ' '], [' ', 'f'], [' ', ' '], ['p', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']]]

[[[' ', ' '], [' ', 'f'], ['p', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']]]

[[[' ', ' '], [' ', 'f'], [' ', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['p', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']]]

[[[' ', ' '], [' ', 'f'], ['', ' '], ['p', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']]]

[[[' ', ' '], ['p', 'f'], ['', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']]]

[[[' ', ' '], [' ', 'f'], ['', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], ['p', ' '], [' ', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']]]

[[[' ', ' '], [' ', 'f'], [' ', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], ['p', ' '], ['', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']]]

[[[' ', ' '], [' ', 'f'], [' ', ' '], ['p', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']]]

[[[' ', ' '], [' ', 'f'], [' ', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], ['p', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']]]

[[[' ', ' '], [' ', 'f'], ['', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['p', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']]]

[[[' ', ' '], ['', 'f'], ['p', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']]]

[[['p', ' '], ['', 'f'], ['', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']]],

[[[' ', ' '], ['', 'f'], ['', ' '], ['', ' ']],

[[' ', ' '], ['p', ' '], [' ', ' '], [' ', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']]]]

[[[[' ', ' '], [' ', 'f'], [' ', ' '], ['p', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']]]

[[[' ', ' '], [' ', 'f'], ['p', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']]]

[[[' ', ' '], ['p', 'f'], ['', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']]]

[[[' ', ' '], ['p', ''], ['', ' '], ['', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[['w', 'w'], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']]]]

**Αρχική κατάσταση 2:**

[[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['p', 'f']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], ['w', 'w'], [' ', ' ']]]

**Μονοπάτι:**

[[[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['p', 'f']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], ['w', 'w'], [' ', ' ']]]

[[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], ['p', ' '], ['', 'f']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], ['w', 'w'], [' ', ' ']]]

[[[' ', ' '], [' ', ' '], [' ', ' '], ['p', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['', 'f']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], ['w', 'w'], [' ', ' ']]]

[[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['', 'f']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['p', ' ']],

[[' ', ' '], [' ', ' '], ['w', 'w'], [' ', ' ']]]]

[[[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['p', 'f']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], ['w', 'w'], [' ', ' ']]]

[[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], [' ', ' '], ['p', '']],

[[' ', ' '], [' ', ' '], [' ', ' '], [' ', ' ']],

[[' ', ' '], [' ', ' '], ['w', 'w'], [' ', ' ']]]]

***Σημείωση:***

Κώδικας αλγορίθμου Πρώτα σε Βάθος αναζήτησης(DFS) με παρακολούθηση του μετώπου αναζήτησης:

Αρχείο *DFS\_with\_front.py*

Κώδικας αλγορίθμου Πρώτα σε Βάθος αναζήτησης(DFS) με παρακολούθηση του μετώπου και της ουράς αναζήτησης:

Αρχείο *DFS\_with\_front\_and\_queue.py*

Κώδικας αλγορίθμου Πρώτα σε Πλάτος αναζήτησης(BFS) με παρακολούθηση της ουράς μονοπατιών :

Αρχείο *BFS\_with\_front\_and\_queue.py*