ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

«Τεχνητή Νοημοσύνη και Έμπειρα Συστήματα»

Ονοματεπώνυμο: Δημητρέλλος Παναγιώτης

ΑΜ: Π17026

Ημερομηνία: Ιούνιος 2020

*Assignment* AI&EE 2020

*Θέμα*: «Αναπτύξτε πρόγραμμα υπολογισμού της διαδρομής εξόδου στον παρακάτω λαβύρινθο (Ι: είσοδος, G: έξοδος), χρησιμοποιώντας τον αλγόριθμο branch and bound.»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | K | L | F |  |
|  | J |  | E | G |
|  | H |  | D |  |
| Ι | A | B | C |  |
|  |  |  |  |  |

*Α. Branch & Bound / Επέκταση & Οριοθέτηση*

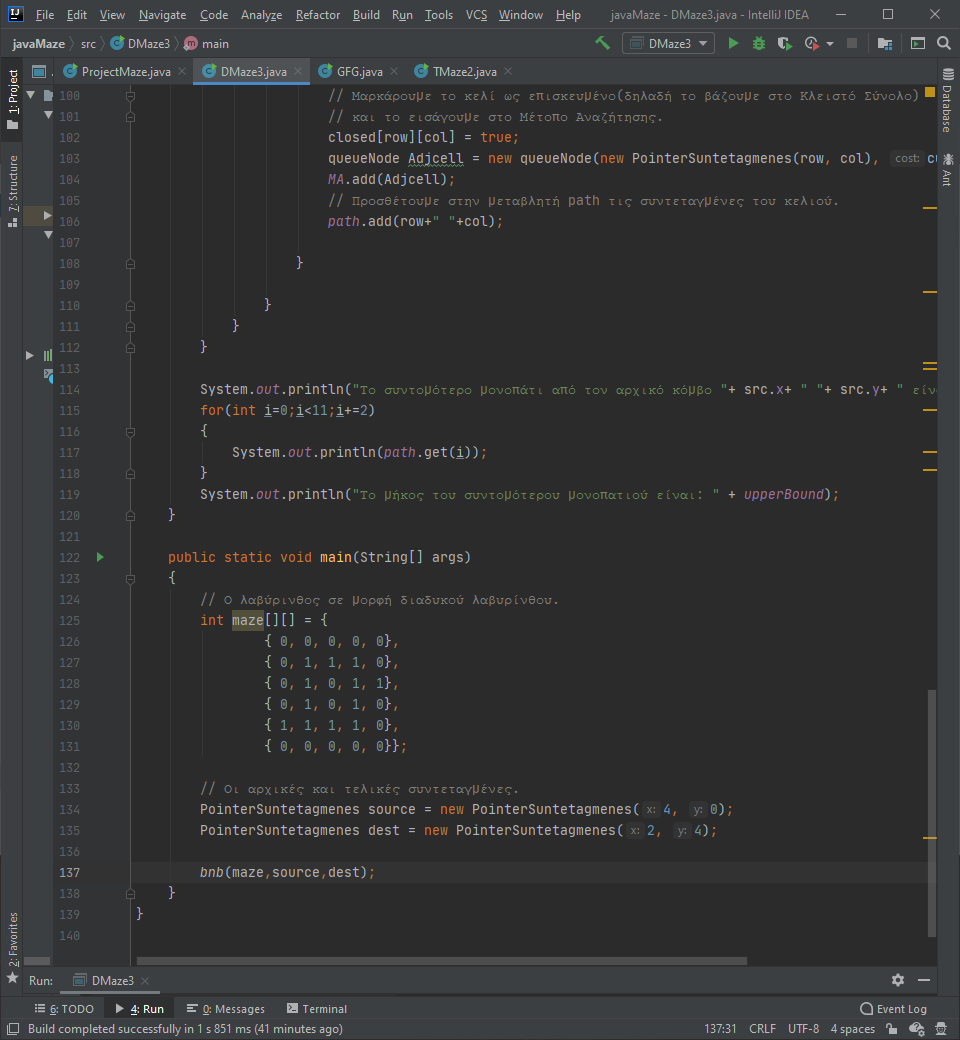
Ο αλγόριθμος «Branch & Bound» («Επέκταση & Οριοθέτηση») ανήκει στους τυφλούς αλγορίθμους αναζήτησης. Οι συγκεκριμένοι χρησιμοποιούνται όταν δεν υπάρχει πληροφορία για την αξιολόγηση των καταστάσεων, αλλά υπάρχει μία πραγματική εκτίμηση του κόστους. Σε αυτούς έχει σημασία η χρονική σειρά με την οποία παράγονται οι καταστάσεις από τον μηχανισμό επέκτασης. Συγκεκριμένα, ο «Branch & Bound» εφαρμόζεται όταν έχουμε προβλήματα βελτιστοποίησης. Στηρίζεται στη μέθοδο «Κλάδεμα/Pruning», στην απόρριψη δηλαδή κάποιων καταστάσεων και κατά επέκταση των υποδένδρων/παιδιών αυτών, βάσει ορισμένων προϋποθέσεων/ορίων. Αυτό έχει ως αποτέλεσμα τη μείωση του χώρου του μετώπου αναζήτησης αφού απορρίπτονται και δεν εξετάζονται οι κατάστασεις που δεν έχουν προοπτική για βελτίωση της λύσης, με άμεση απόρροια τη μικρότερη απαίτηση σε μνήμη. Όμως, με το κλάδεμα των καταστάσεων υπάρχει ο κίνδυνος να μην βρεθεί τελικά λύση εάν ο αλγόριθμος οδηγηθεί σε αδιέξοδο. Σε αυτή την περίπτωση δεν είναι εφικτό να γυρίσει πίσω στις προηγούμενες καταστάσεις που κλάδεψε. Συνοπτικά, η προσέγγιση που ακολουθεί ο «Branch & Bound» είναι να εξετάζει και να επεκτείνει μία μία της καταστάσεις που υπάρχουν στο μέτωπο αναζήτητης και βάσει κάποιου ορίου/προϋπόθεσης να απορρίπτει τις καταστάσεις – παιδιά που δεν υπακούουν σε αυτό.

*B. Υπολογισμός Διαδρομής Λαβυρίνθου με τoν Branch & Bound*

1. Γενικά Χαρακτηριστικά Προγράμματος

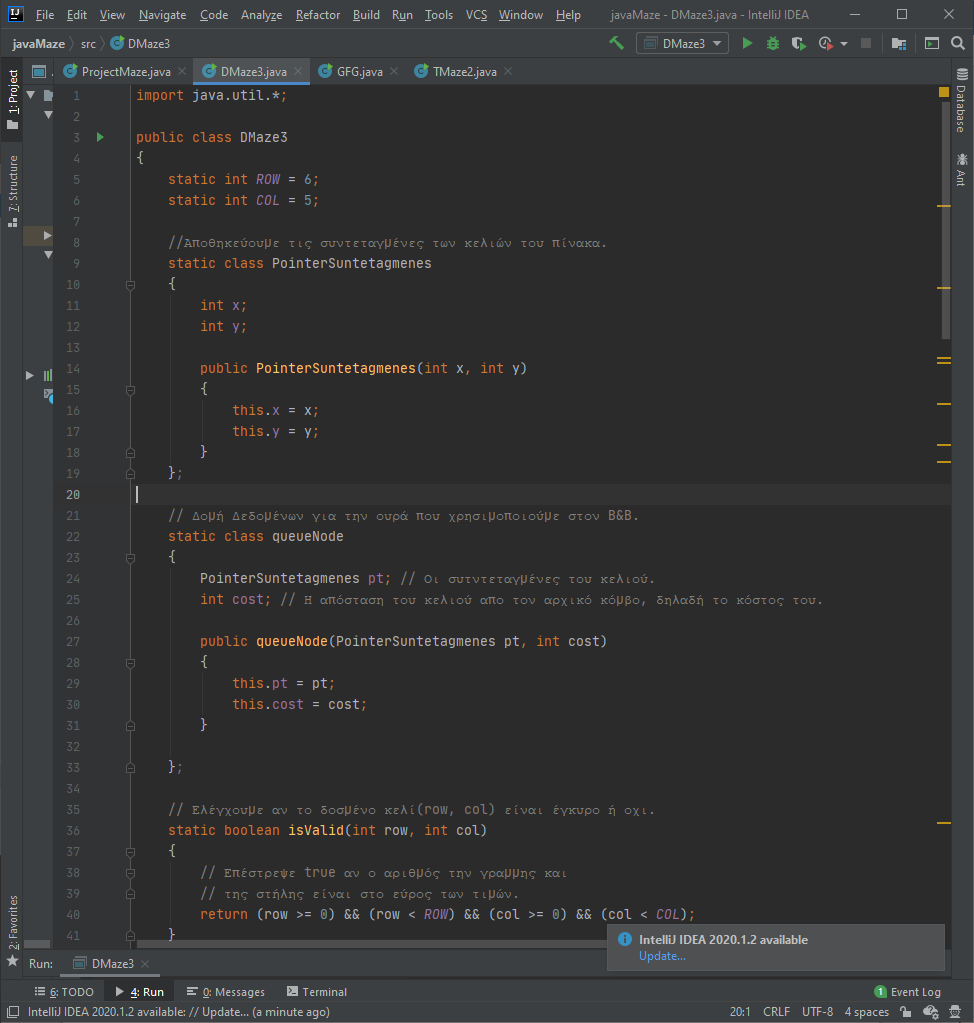
a. Δυσδιάστατος πίνακας «maze»

Ο δυσδιάστατος πίνακας «maze[6][5]» τύπου int/ακεραίου αναπαριστά τον ζητούμενο προς επίλυση λαβύρινθο με μαθηματικό τρόπο. Για τα άσπρα κελιά του λαβύρινθου ο πίνακας έχει ‘1’, αλλιώς για τα μαύρα έχει ‘0’.



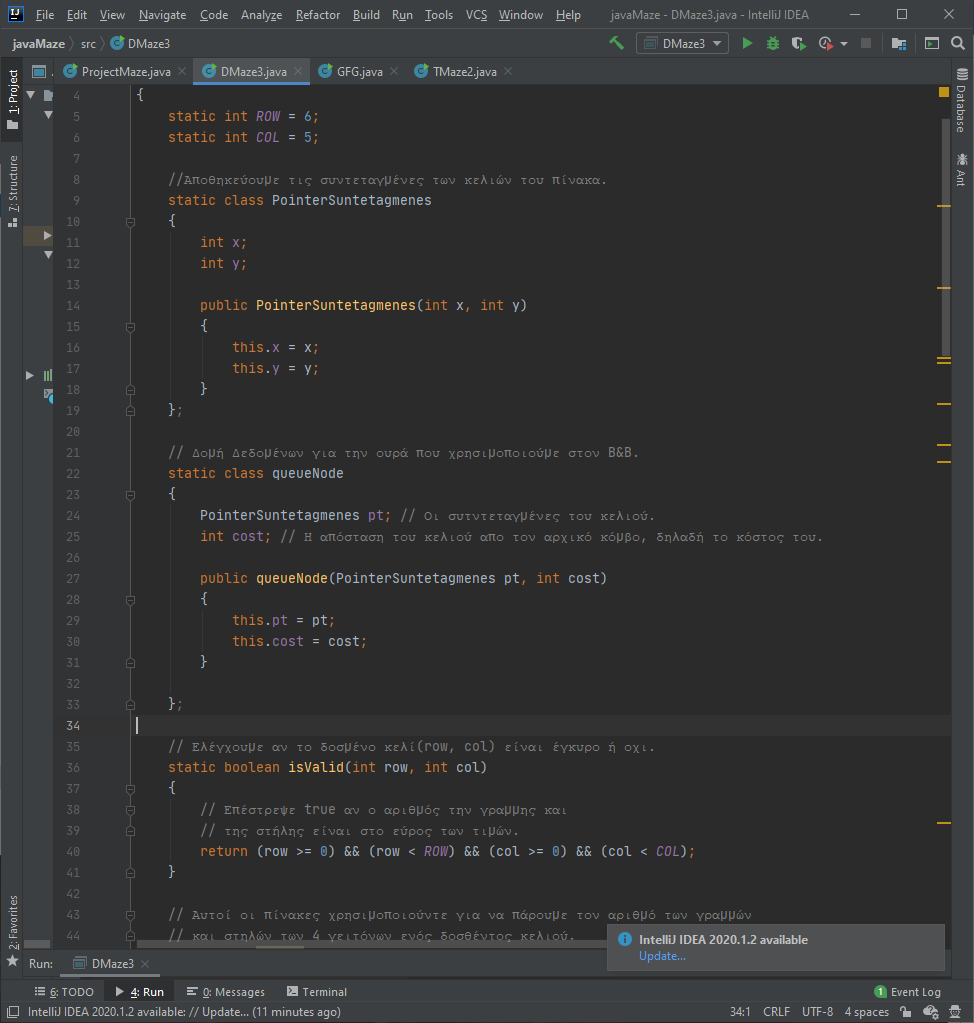
b. Κλάση «PointerSuntetagmenes»

Κατασκευάζουμε μία νέα κλάση PointerSuntetagmenes στην οποία αποθηκεύουμε τις συντεταγμένες των κελιών του πίνακα.



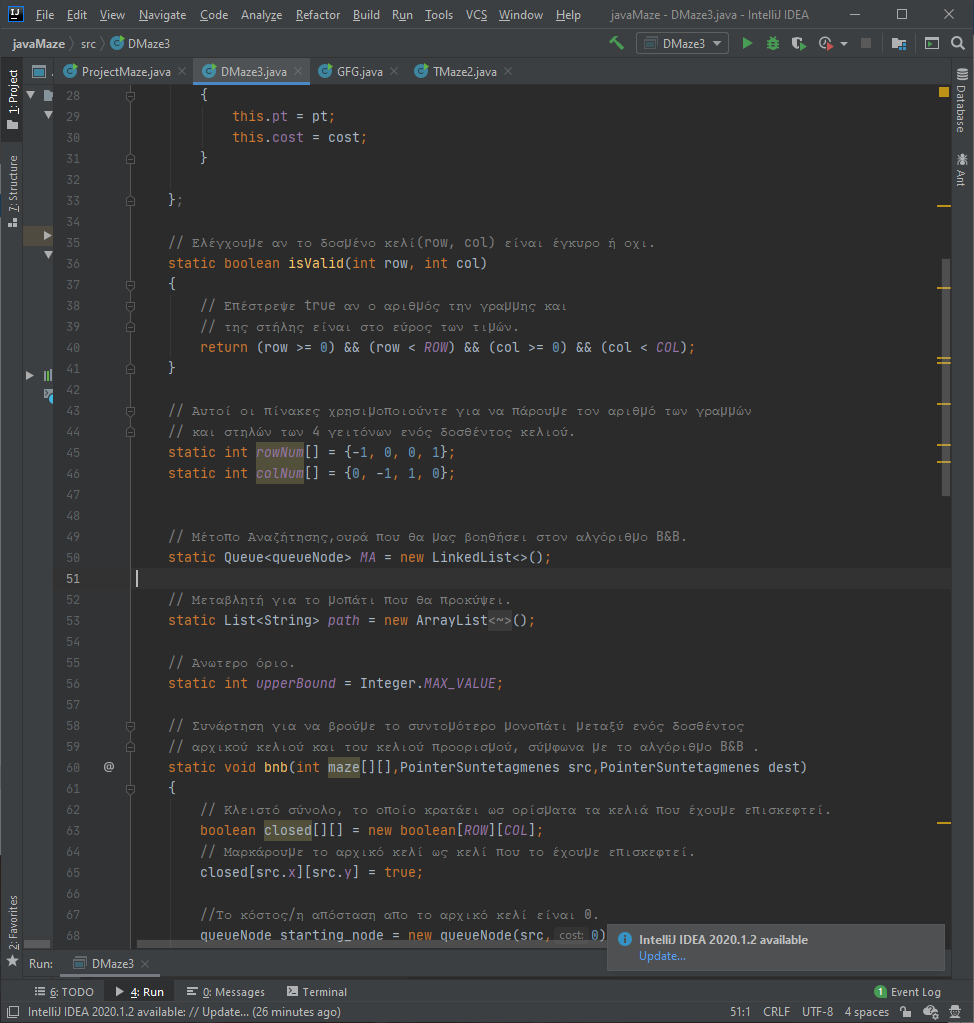
c. Κλάση «queueNode»

Κατασκευάζουμε μία νέα κλάση queueNode στην οποία ο constructor παίρνει ως ορίσματα τις συντεταγμένες του κελιού και το κόστος ή αλλιώς την απόσταση του κελιού από τον αρχικό κόμβο. Επίσης αυτή η κλάση χρησιμοποιείται ως δομή δεδομένων για την ουρά στην λειτουργία του αλγορίθμου B&B.



d.Πίνακες «rowNum[]» και «colNum[]»

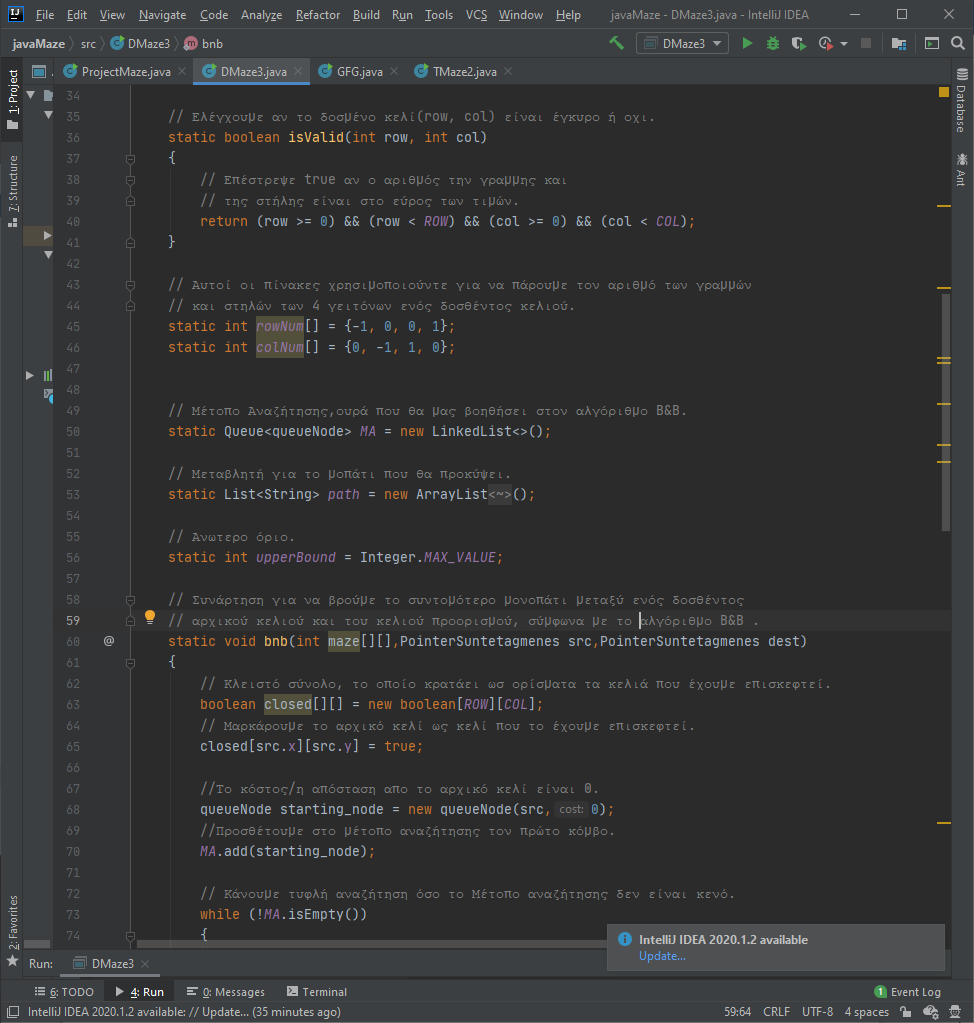
Αυτοί οι πίνακες χρησιμοποιούνται για να πάρουμε τον αριθμό των γραμμών και των στηλών των 4 γειτόνων ενός δοσθέντος κελιού.



Με τον συνδιασμό των αριθμών μέσα στους δύο αυτούς πίνακες μπορούμε να παράγουμε τις 4 κατευθύνσεις πάνω, κάτω, δεξιά, αριστερά.

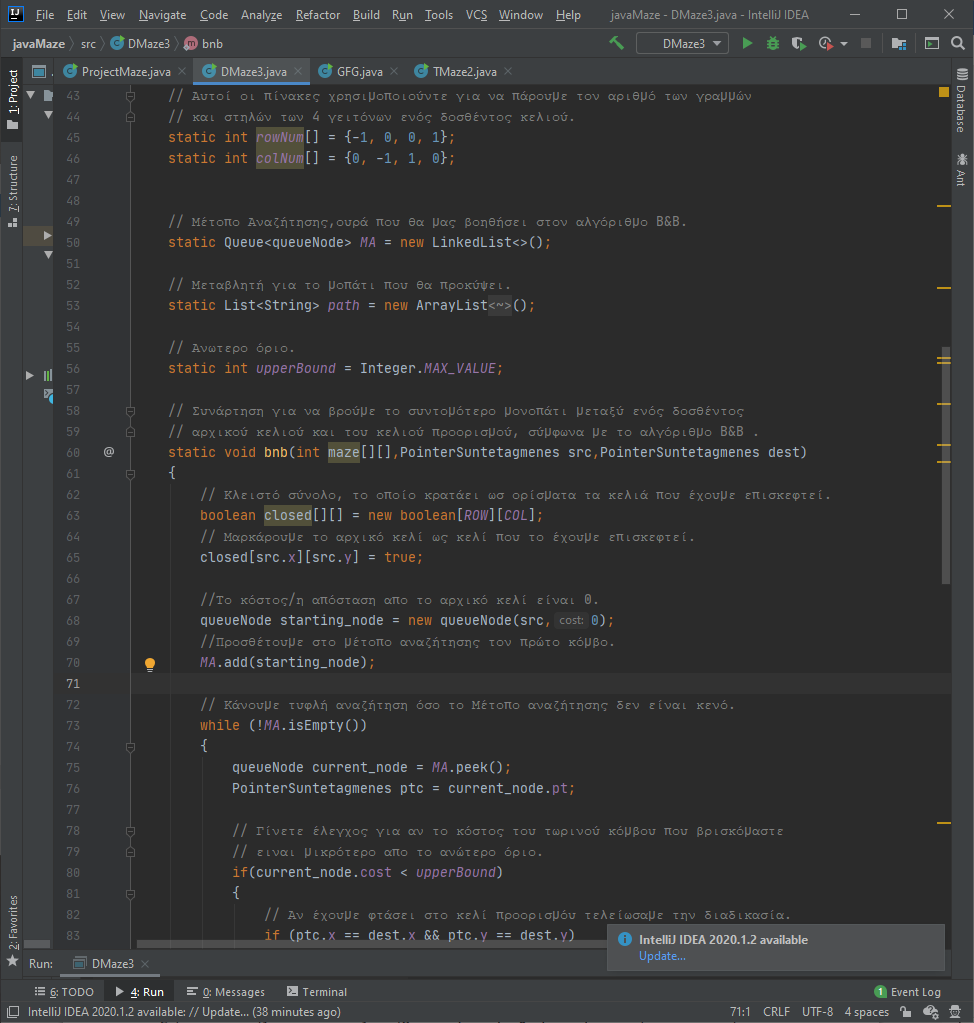
e. Μέτωπο Αναζήτησης «ΜΑ»

Η ουρά «ΜΑ», τύπου «queueNode», αποτελεί το μέτωπο αναζήτησης του αλγορίθμου «branch & bound». Το μέτωπο αναζήτησης αποτελεί την ουρά των καταστάσεων που έχουν ήδη επισκεφθεί, αλλά δεν έχουν επεκταθεί. Εξάγεται κάθε φορά το πρώτο στοιχείο της ουράς προς επέκταση, καθώς η μέθοδος που ακολουθείται από την ουρά είναι η «FIFO», δηλαδή «first-in first-out».



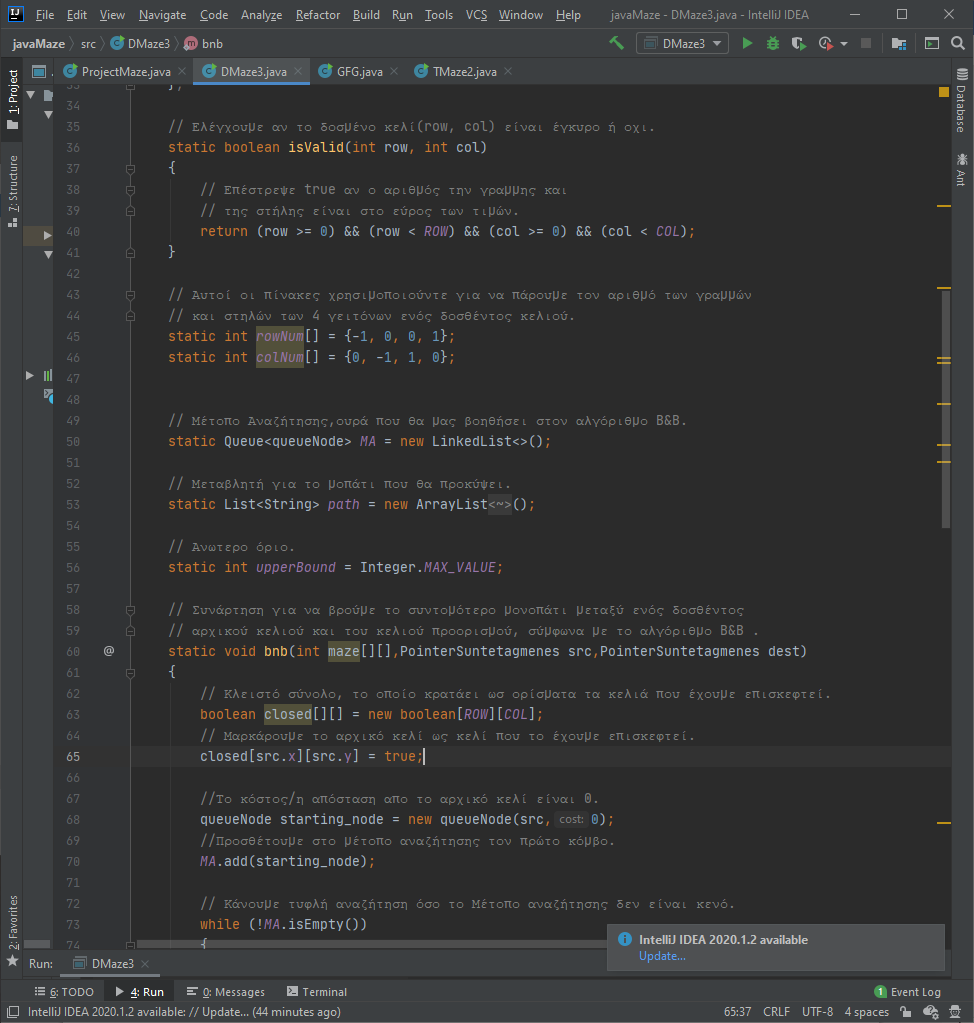
f. Κλειστό Σύνολο «closed[][]»

Ο πίνακας «closed[][]» ,τύπου Boolean, αποτελεί το κλειστό σύνολο του αλγορίθμου «branch & bound». Απαρτίζει το σύνολο όλων των καταστάσεων που έχουν ήδη επεκταθεί. Εισάγονται στη λίστα έτσι ώστε να μην ξανατοποθετηθούν στο μέτωπο αναζήτησης και ξαναεξεταστούν, καθώς η έκβαση θα είναι η ίδια.



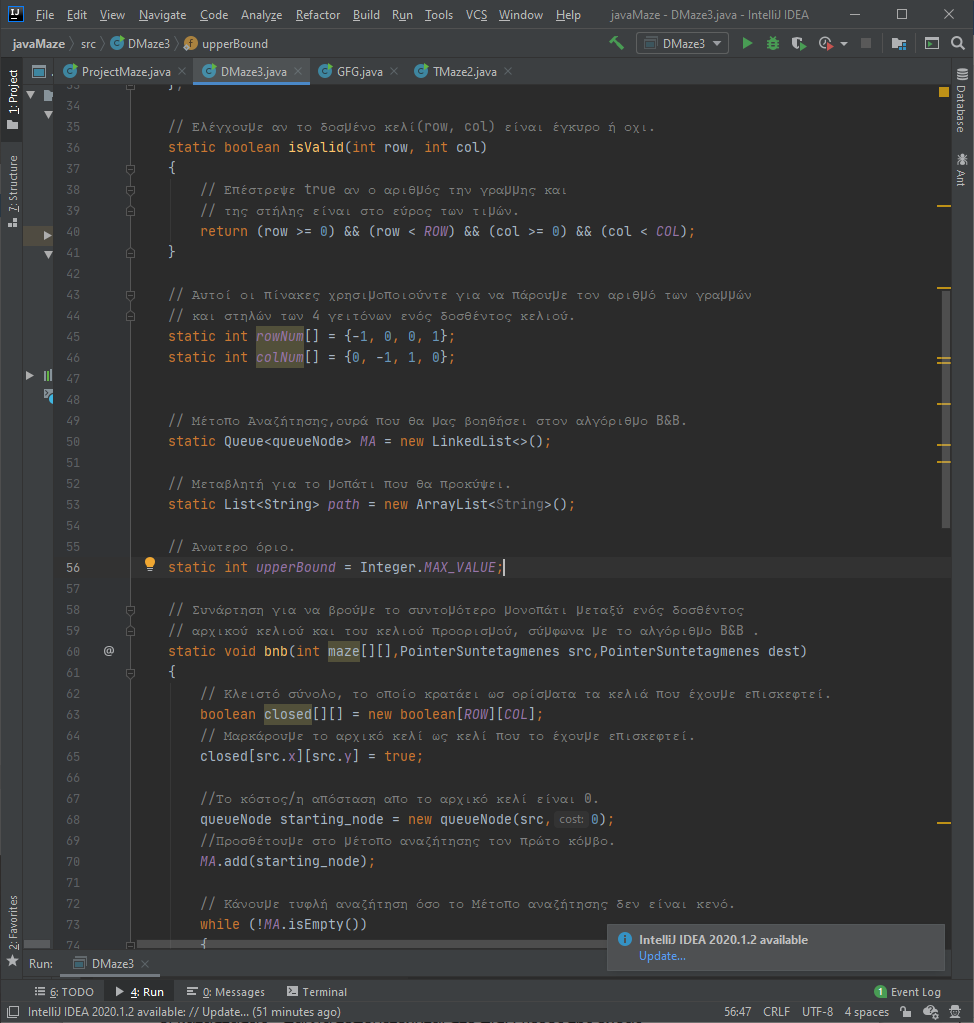
g. Μεταβλητή «upperBound»

Η μεταβλητή «upperBound» αποτελεί το ανώτερο όριο με το οποίο γίνεται η περικοπή των καταστάσεων και αντιστοιχεί στο ελάχιστο κόστος της τρέχουσας κάθε φοράς συντομότερης διαδρομής. Στην αρχή παίρνει την μεγαλύτερη εφικτή ακέραια τιμή του συστήματος και όταν μία διαδρομή φτάνει στον τελικό κόμβο «G», αναθεωρείται. Αν το κόστος της νέας διαδρομής είναι μικρότερο από την τιμή της τρέχουσας «upperBound», η «upperBound» αλλάζει και παίρνει την νέα αυτή τιμή.



h. Λίστα «path»

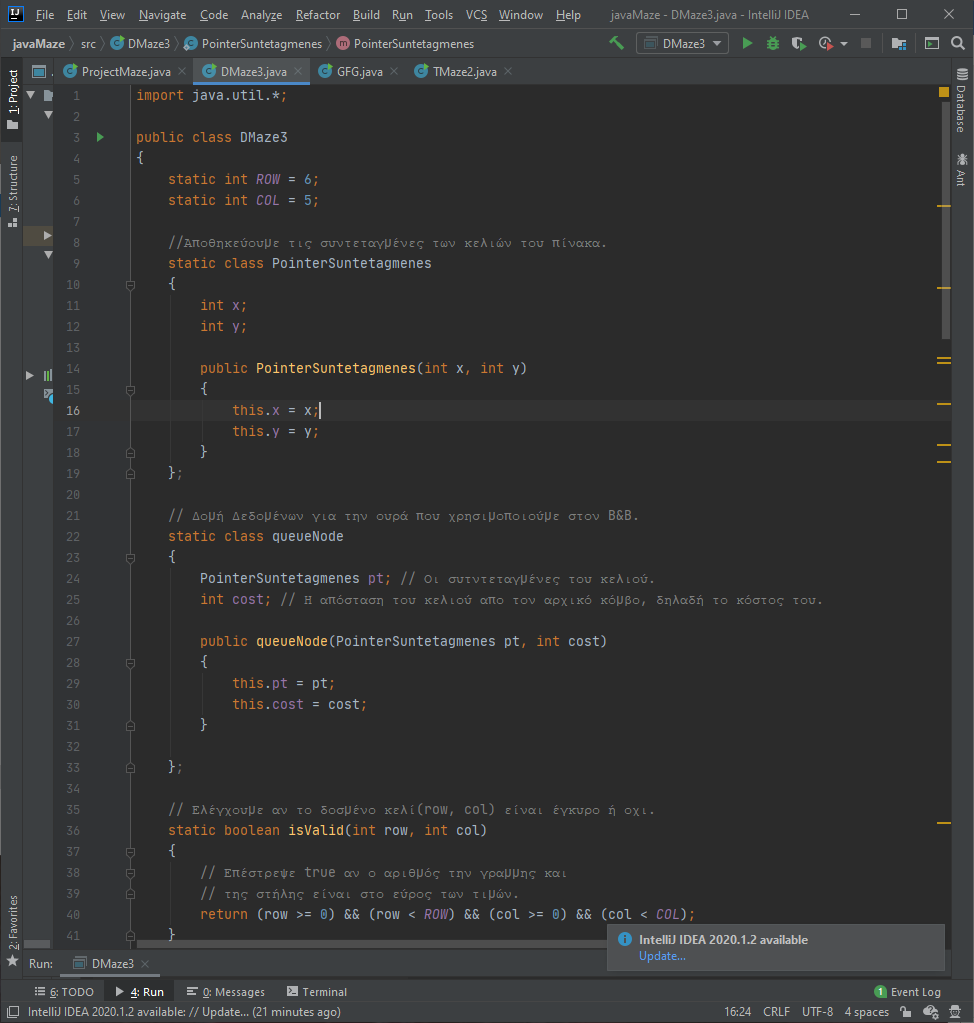
Η λίστα «path»,τύπου String, αποθηκεύει την διαδρομή που ακολούθησε ο αλγόριθμος «branch & bound», δηλαδή την διαδικασία εύρεσης μονοπατιού για να φτάσει στο εκάστοτε κελί.

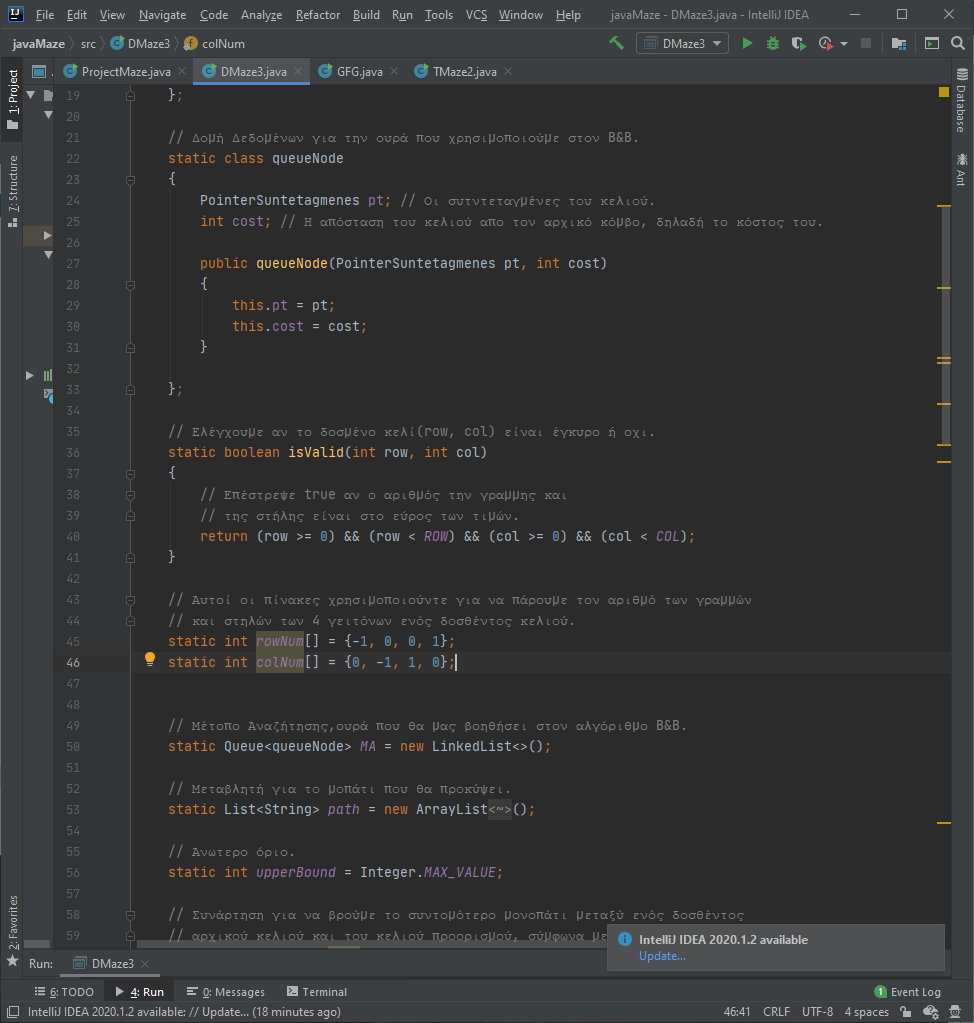


2. Μέθοδοι Προγράμματος

a. Μέθοδος «isValid»

Η συνάρτηση isValid κατασκευάστηκε για να ελέγχει αν το δοσμένο κελί (row, col) είναι έγκυρο ή όχι. Η συνάρτηση αυτή επιστρέφει Boolean δηλαδή false/true. Σε αυτήν την περίπτωση επιστρέφει true αν ο αριθμός της γραμμής και της στήλης είναι μέσα στο εύρος των τιμών ROW, COL όπου έχουν δοθεί ως 6 και 5 αντίστοιχα.





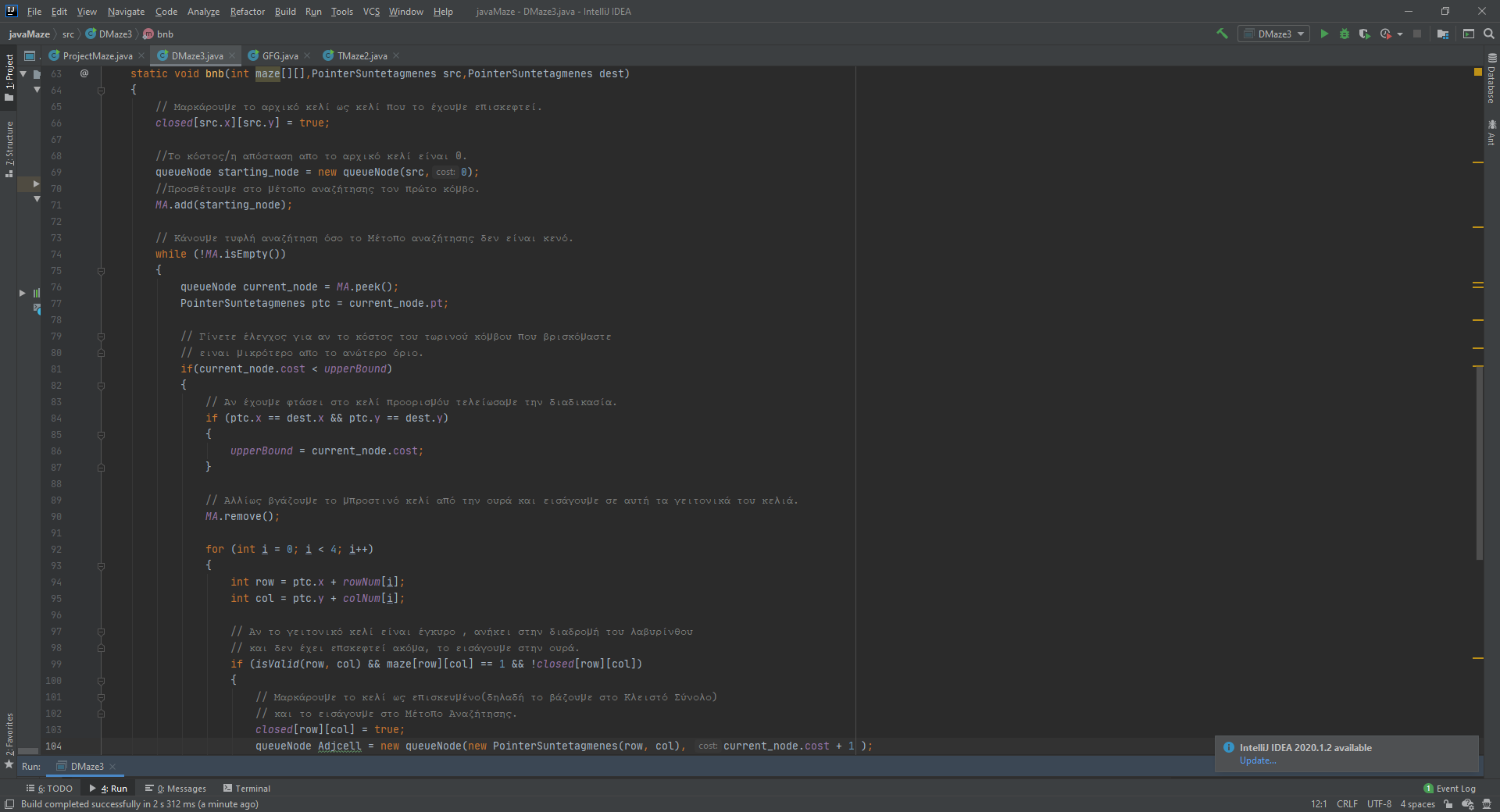
b. Μέθοδος «bnb»

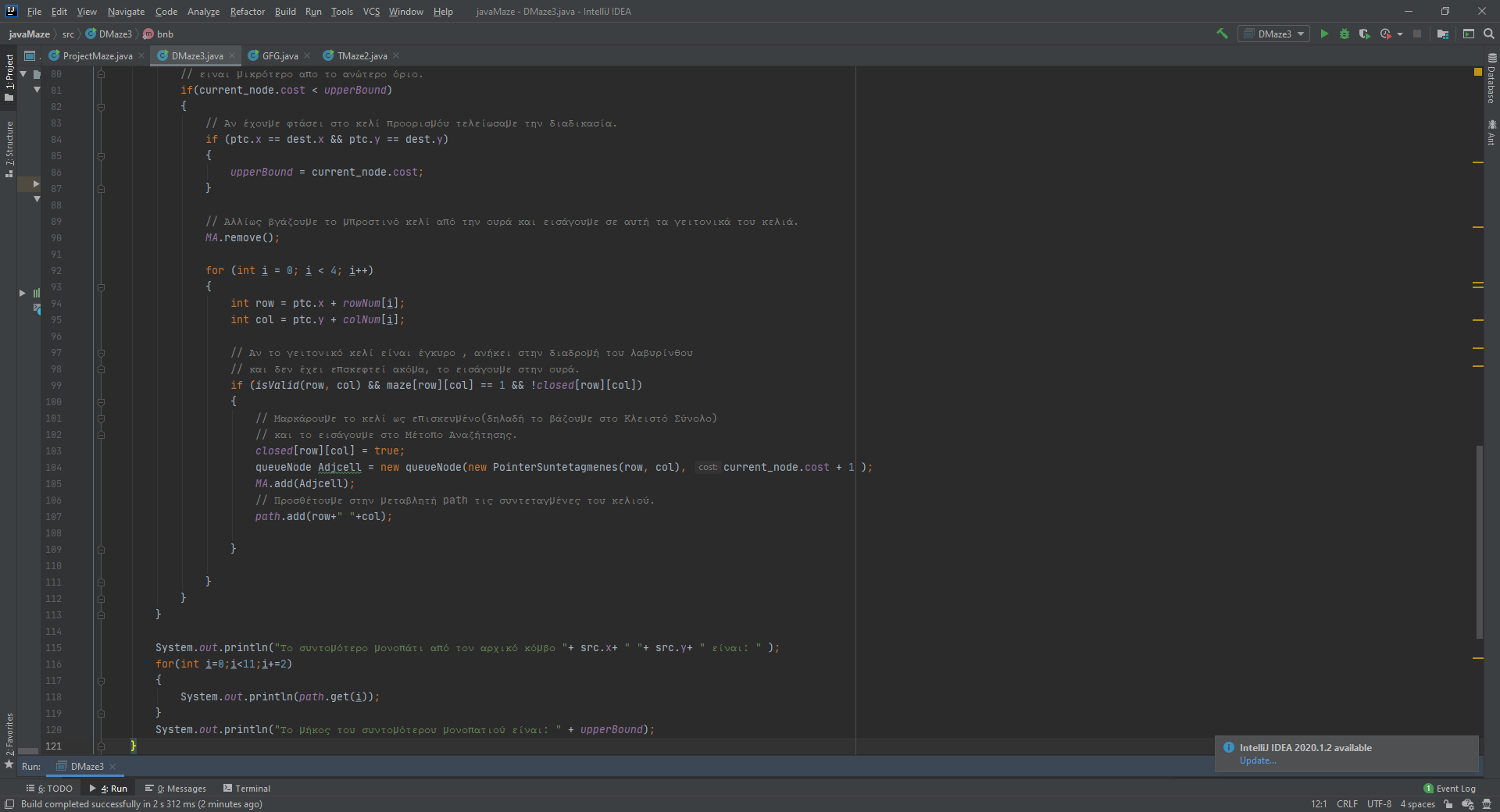
Μέθοδος που περιγράφει τον αλγόριθμο «branch and bound».Αρχικά μαρκάρουμε το κελί εκκίνησης ως κελί που έχουμε επισκεφτεί και στην συνέχεια το προσθέτουμε στο μέτωπο αναζήτησης με κόστος 0.

Ορίζεται ένας προσωρινός κόμβος «current\_node» που βοηθά στην εξέταση των κατάστασεων. Η παρακάτω διαδικασία επαναλαμβάνεται έως ότου δεν υπάρχουν άλλα στοιχεία στο μέτωπο αναζήτησης, δηλαδή στην ουρά «ΜΑ».

1. Τίθεται ο προσωρινός κόμβος ίσος με το πρώτο στοιχείο του μετώπου αναζήτησης που εξάγεται προς εξέταση
2. Ελέγχεται αν το κόστος του τωρινού κόμβου που βρισκόμαστε είναι μικρότερο από το ανώτερο όριο που έχουμε ορίσει.
3. Αν η παραπάνω πρόταση αληθεύει τότε γίνεται έλεγχος αν ο κόμβος που εξετάζουμε είναι ο τελικός και αν ναι τότε το ανώτερο όριο παίρνει την τιμή του κόστους του κόμβου που είμαστε.
4. Αλλιώς βγάζουμε το μπροστινό κελί από το μέτωπο αναζήτησης και εισάγουμε σε αυτό τα γειτονικά του κελιά.
5. Τα γειτονικά κελιά τα βρίσκουμε με την χρήση ενός for loop όπου με βάση τους πίνακες rowNum[] , colNum[] (οι οποίοι ορίζουν τις 4 διευθύνσεις όπως προαναφέραμε στο 1.d ) σε συνδιασμό με τις συντεταγμένες του κόμβου που βρισκόμαστε.
6. Αν το γειτονικό κελί είναι έγκυρο , ανήκει στην διαδρομή του λαβυρίνθου και δεν το έχουμε επσκεφτεί ακόμα, το εισάγουμε στο κλειστό σύνολο για να μην ξαναεισαχθεί σε επόμενη επέκταση άλλου κόμβου στην ουρά.
7. Δημιουργούμε έτσι από τον κόμβο τον οποίο βρισκόμασταν μια γειτονική κατάσταση και ανεβάζουμε την τιμή του κόστους κατά 1 μέχρι να φτάσουμε στον προορισμό μας.

Μόλις αδειάσει η ουρά «ΜΑ» και δεν υπάρχουν άλλες καταστάσεις προς εξέταση, ο αλγόριθμος σταματά. Βρίσκεται έτσι η καλύτερη λύση, εφόσον υπάρχει, δηλαδή η μικρότερη σε μήκος διαδρομή από το σημείο εισόδου «Ι» στο σημείο εξόδου «G».

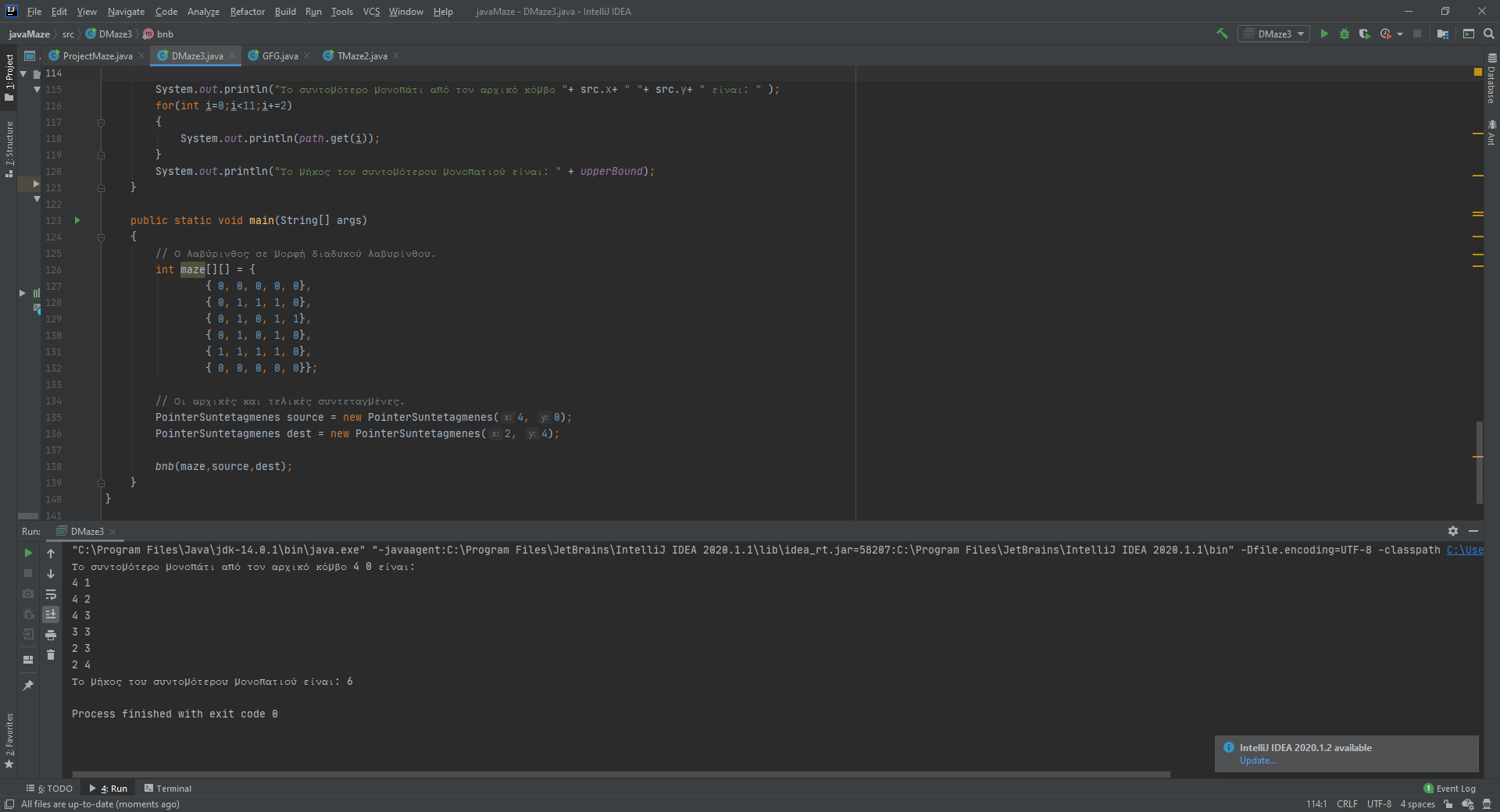




Με αυτό τον τρόπο τίθεται σε εφαρμογή η λογική πίσω από τον «branch & bound», η οποία έπεται στην επέκταση μόνο των κόμβων που έχουν προοπτική για βελτίωση της λύσης. Εχοντας ένα ανώτερο φράγμα, απορρίπτονται οι διαδρομές που οδηγούν σε μονοπάτια μεγαλύτερου μήκους (άρα και κόστους) από το ανώτερο όριο/κόστος.

3. Παράδειγμα εκτέλεσης κώδικα

Τρέχουμε το πρόγραμμα, όπου από την συνάρτηση «main» καλείται η συνάρτηση «bnb» και μας εμφανίζει το συντομότερο μονοπάτι που βρήκε ο αλγόριθμος καθώς και το μήκος του, που φυσικά είναι το μικρότερο που μπορεί να έχει μια διαδρομή από το σημείο 4,0 έως το 2,4 .



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Μέτωπο Αναζήτησης | Κλειστό Σύνολο | Κατάσταση | Όριο | Παιδιά |
| I |  | I | +∞ | A |
| A | I | A | +∞ | B, I, H |
| B, I, H | I, A | B | +∞ | C, A |
| C, A, I, H | I, A, B | C | +∞ | D, B |
| D, B, A, I, H | I, A, B, C | D | +∞ | E, C |
| E, C, B, A, I, H | I, A, B, C, D | E | +∞ | G, F, D |
| G, F, D, C, B, A, I, H | I, A, B, C, D, E | G | 6 | ΛΥΣΗ |
| F, D, C, B, A, I, H | I, A, B, C, D, E | F | 6 | X (>=6) |
| D, C, B, A, I, H | I, A, B, C, D, E, F | D | 6 | - |
| C, B, A, I, H |  | C | 6 | - |
| B, A, I, H |  | B | 6 | - |
| A, I, H |  | A | 6 | - |
| I, H |  | I | 6 | - |
| H |  | H | 6 | J, A |
| J, A | I, A, B, C, D, E, F, H | J | 6 | K, H |
| K, H, A | I, A, B, C, D, E, F, H, J | K | 6 | L, J |
| L, J, H, A | I, A, B, C, D, E, F, H, J, K | L | 6 | F, K |
| F, K, J, H, A | I, A, B, C, D, E, F, H, J, K, L | F | 6 | X (>=6) |
| K, J, H, A |  | K | 6 | - |
| J, H, A |  | J | 6 | - |
| H, A |  | H | 6 | - |
| A |  | A | 6 | - |

Ένα παράδειγμα της πιο αναλυτικής εκτέλεσης του αλγορίθμου B&B είναι το ακόλουθο :