Τασος Τσούκας ΑΜ: 3140213

Βασίλης Σταυριανουδάκης ΑΜ: 3140193

Έκτορας Τυπάλδος ΑΜ: 3140216

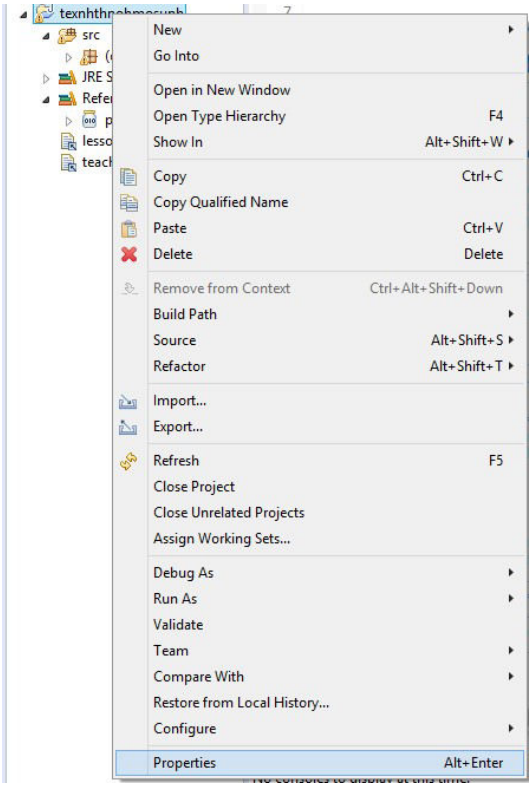
Η πρώτη εργασία για το μάθημα της Τεχνητής Νοημοσύνης αναφέρεται σε ένα πρόγραμμα το οποίο κατασκευάζει ρεαλιστικά ένα ωρολόγιο πρόγραμμα μαθημάτων ενός γυμνασίου χρησιμοποιώντας μεθόδους αναζήτησης σε χώρους καταστάσεων. Η εργασία υλοποιήθηκε σε **Java**.

Η παρούσα εργασία δοκιμάστηκε σε ρεαλιστικά δεδομένα γυμνασίου καθώς βρήκαμε τον μέγιστο αριθμό ορών διδασκαλίας απο το υπουργείο (32 ώρες ανα τμήμα) και τον αριθμό και τα μαθήματα που ειναι υποχρεωτικό να διδαχθούν .Στην συνέχεια λάβαμε υπόψην ενα κανονικό πρόγραμμα διδασκαλίας ενός γυμνασίου της Αθήνας και προσπαθήσαμε όσο το δυνατών να προσεγγίσουμε τον αριθμό των καθηγητών που υπήρχαν εκεί συνδέοντας παράλληλα και τους περιορισμούς που υπάρχουν στην εργασία.

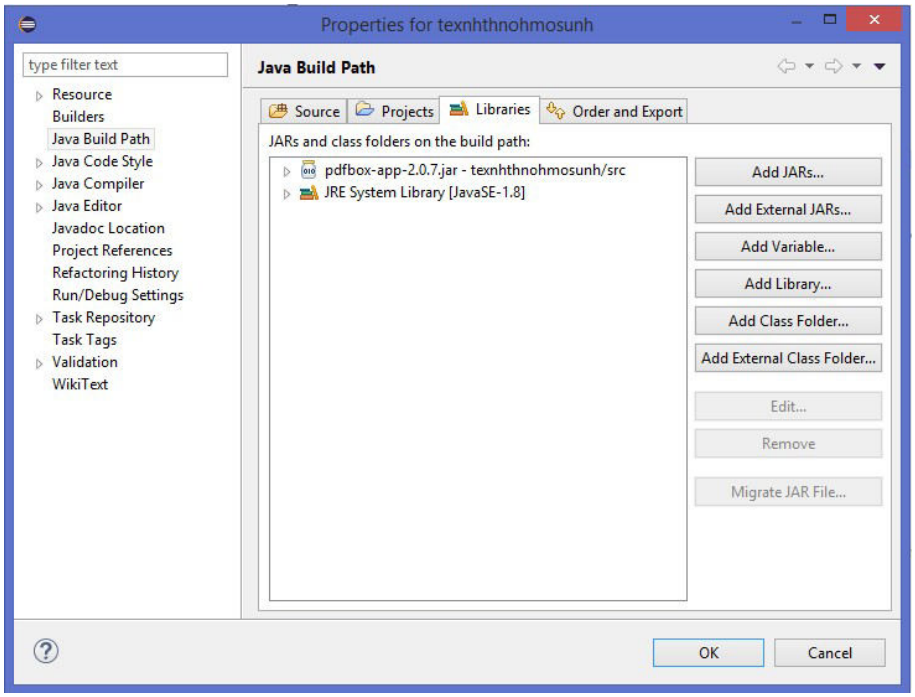
Το πρόγραμμα αρχικά δέχεται ως είσοδο δύο αρχεία απλού κειμένου txt **lessons** και **teachers**

- Το **lessons** περιέχει τους κωδικούς των μαθημάτων οι οποίοι αναπαράγονται ως διψήφιοι αριθμοί που ξεκινάνε απο 00 και τελειώνουν σε 54.Στην συνέχεια περιέχει τα ονόματα όλων των μαθημάτων που διδάσκονται στο γυμνάσιο καθώς και τις τάξεις στις οποίες διδάσκονται.Τέλος αναγράφεται το πλήθως των ορών που διδάσκεται το κάθε μάθημα στην κάθε τάξη.Ολα τα παραπάνω χαρακτηριστικά χωρίζονται με κόμμα (,).
- Το **teachers** περιέχει τους κωδικούς των καθηγητών οι οποίοι αναπαράγονται ως τριψήφιοι αριθμοί που ξεκινάνε απο το 000 και τελειώνουν στο 019.Στην συνέχεια περιέχει τα ονόματα όλων των καθηγητών που διδάσκουν στο γυμνάσιο.Επίσης αναγράφονται οι κωδικοί (απο το txt lessons) των μαθημάτων που διδάσκει ο κάθε καθηγητής.Τέλος περιέχεται ο μέγιστον αριθμός ωρών ανα μέρα και ανα εβδομάδα που επιτρέπεται να διδάσκει ενας καθηγητής.Ολα τα παραπάνω χαρακτηριστικά χωρίζονται με κόμμα (,).

Για να τρέξει η εργασία ορθά και να εμφανίσει το πρόγραμμα σε μορφή pdf θα πρέπει να γίνει import το αρχείο jar **pdfbox-app-2.0.7.**Για να γίνει αυτο στο eclipse θα πρέπει να πάτε στο όνομα του java project να κάνετε δεξί κλικ και να πατήσετε στην επιλογή κάτω κάτω properties.



Στην συνέχεια επιλέγετε στα αριστερά την επιλογή Java Build Path και πάτε στο παράθυρο libraries.



Τέλος πατάτε το Add External JARs στα δεξιά και επιλέγετε το αρχείο **pdfbox-app-2.0.7** και μετά OK.

Στην εργασία χρησιμοποιείτε ο αλγόριθμος Τεχνητής Νοημοσύνης **Beam Search** δίνοντάς του εναν αριθμό καταστάσεων **numbOfBeams**. Επίσης δίνεται η δυνατότητα για numbOfBeams=1 να τρέξει ο αλγόριθμος **Hill Climbing**.

Οι κλάσεις που χρησιμοποιήσαμε είναι οι εξής:

-Η κλάση Lesson αντιπροσωπεύει το κάθε μάθημα, συγκεκριμένα περιεχέι τις παρακάτω μεταβλητές:

id = αποθηκεύει τον αριθμό του κάθε μαθήματος.

lessonName = το όνομα του κάθε μαθήματος.

ClassRoom = την τάξη στην οποία διδάσκετε το μάθημα.

Hourly = πόσες ώρες την βδομάδα διδάσκετε το μάθημα.

-Η κλάση Teacher αντιπροσωπεύει τον κάθε καθηγητή, συγκεκριμένα περιεχέι τις παρακάτω μεταβλητές:

id = Αποθηκεύει τον κωδικό του κάθε καθηγητή όπως τον πήρε απ το txt αρχείο.

InsideId = Αποθηκεύει ένα μοναδικό int ID για κάθε καθηγητή ώστε να μας διευκολύνει να χειριστούμε αργότερα τα δεδομένα.

Name = το ονοματεπώνυμο του καθηγητή.

ArrayList<Integer> lessons = αποθηκεύει τα Id Των μαθημάτων που διδάσκει ο καθηγητής αυτός.

MaxDay = Τις μέγιστες ώρες που διδάσκει ο καθηγητής την ήμερα.

MaxWeek = Τις μέγιστες ώρες που διδάσκει ο καθηγητής την βδομάδα.

-Η κλάση LessonWithTeacher αντιπροσωπεύει τον συνδυασμό κάθε μαθήματος με κάθε καθηγητή που το διδάσκει, συγκεκριμένα περιεχέι τις παρακάτω μεταβλητές:

lessonId = ο κωδικός του μαθήματος.

TeacherId = ο κωδικός του καθηγητη.

-Η κλάση DataForSchedule είναι η κλάση στην οποία διαβάζουμε τις πληροφορίες από το txt, δημιουργούμε τα παραπάνω αντικείμενα και κρατάμε τα δεδομένα για την μελλοντική χρήση τους στο πρόγραμμα:

συγκεκριμένα έχει τα παρακάτω ArrayLists που κρατάνε πληροφορίες:

```
 private ArrayList<LessonWithTeacher> lessonsA :  LessonWithTeachers που διδάσκονται στην A τάξη.

 private ArrayList<LessonWithTeacher> lessonsB :  LessonWithTeachers που διδάσκονται στην B τάξη.
```

`private ArrayList<LessonWithTeacher> lessonsC` : **LessonWithTeachers** που διδάσκονται στην Γ τάξη.

`private ArrayList<Lesson> les` : Ολα τα μαθηματα.

`private ArrayList<Teacher> teac` : Όλοι οι καθηγητές.

Η **DataForSchedule** περιεχει τις παρακατω βασικες μεθοδους:

`public void readLessonsFromText(String textName)`: διαβάζει απ το txt τα μαθήματα και γεμίζει το les.

`public void readTeachersFromText(String textName)`: διαβάζει απ το txt τους καθηγητές και γεμίζει το teac.

`public void createDataBase()`: με βαση τον les και τον teac γεμιζει τα ArrayLists lessonsA, lessonsB, lessonsC.

-Η κλάση Report είναι η κλάση με την οποία αξιολογούμε πόσο κάλο είναι ένα πρόγραμμα διδασκαλίας που ήδη έχουμε δημιουργήσει, ενώ επιπλέον επιστρέφει αναλυτική αναφορά της κάθε αστοχίας. Συγκεκριμένα κρατάει ένα totalScore το οποίο διαμορφώνεται αθροιστικά από τις αστοχίες που εντοπίζουμε στο πρόγραμμα αυτό και ένα arraylist με τον κωδικό και τις συντεταγμένες της αστοχίας μέσα στο πρόγραμμα.

Τα κριτήρια που αξιολογεί η Report είναι 6 ,τα βάλαμε με βάση την εκφώνηση και την κρίση μας, και είναι τα εξής (με σειρά προτεραιότητας και με ανάλογη βαρύτητα στο άθροισμα του totalScore.):

1)Να μην υπάρχουν κενά.

2)Να μην διδάσκει ο καθηγητής ταυτόχρονα σε δυο η περισσότερες τάξεις (η τμήματα).

3)Να μη διδάσκει κανένας καθηγητής περισσότερες από δύο συνεχόμενες ώρες.

4)Να μην υπερβαίνει ένας καθηγητής τον μέγιστο αριθμό ωρών ανά βδομάδα.

5)Να μην υπερβαίνει ένας καθηγητής τον μέγιστο αριθμό ωρών ανά μέρα.

6)Οι ώρες διδασκαλίας κάθε μαθήματος σε ένα τμήμα να είναι κατά το δυνατόν ομοιόμορφα κατανεμημένες σε όλες τις ήμερες της εβδομάδας.

(Σημειώση: το τελευταίο κριτήριο που ζητείται στην εκφώνηση, δηλαδή Ο αριθμός ωρών διδασκαλίας ανά εβδομάδα να είναι κατά το δυνατόν ομοιόμορφος για όλους τους καθηγητές (π.χ. να μη διδάσκει ένας 25 ώρες την εβδομάδα και άλλος μόνο 5), θεωρήσαμε ότι δεν είναι ρεαλιστικό για τα πραγματικά δεδομένα ενός σχολείου, για παράδειγμα ένας γυμναστής δεν είναι δυνατόν να διδάσκει περίπου ίδιες ώρες με μια φιλόλογο η οποία έχει πολύ περισσότερα μαθήματα να διδάξει, έτσι αποφασίσαμε να μην το υλοποιήσουμε.)

Οι Βασικές μέθοδοι στην κλάση Report είναι οι εξής:

`public long score(LessonWithTeacher[][] sched, Schedule holeObj)`: η οποία επιστρέφει το score κάποιου προγράμματος διδασκαλίας τύπου Schedule holeObj.

public ArrayList<Integer> getTheProblem(): η οποία επιστρέφει το κριτήριο που πρέπει να ικανοποιηθεί με την μεγαλύτερη προτεραιότητα μέσα σε ένα ArrayList ώστε να υπάρχουν και οι συντεταγμένες του στοιχείου που δημιουργεί το πρόβλημα.

-Η κλάση Schedule είναι η κλάση η οποία αναπαριστά το πρόγραμμα διδασκαλίας. Στην ουσία ο κορμός της είναι ένας πινάκας ο LessonWithTeacher[][] sched ο οποίος είναι [63][5] διαστάσεων (επειδή οι γραμμές είναι 9 τμήματα X 7 ώρες = 63 και στήλες είναι 5 ήμερες) και στην ουσία το κάθε στοιχείο του, είναι μια ώρα διδασκαλίας με κάποιον καθηγητή για το συγκεκριμένο μάθημα, δηλαδή ένα LessonWithTeacher. Σκεφτήκαμε ότι οι πρώτες 7 γραμμές είναι για το Α1, οι επόμενες 7 για το Α2, κλπ, μέχρι τις τελευταίες 7 γραμμές που αντιπροσωπεύουν το ωρολόγιο πρόγραμμα για το Γ3.

Οι σημαντικότερες μέθοδοι της κλάσης αυτής είναι οι εξής:

public void createRandomSchedule(): δημιουργεί ένα τυχαίο πρόγραμμα διδασκαλίας 32 ωρών για το κάθε τμήμα. (32 ώρες είδαμε ότι προβλέπετε από το υπουργείο άρα θα υπάρχουν 3 κενά σε κάθε τμήμα σε τυχαία σημεία αρχικά.)

```
public long makeReport(){
            Report rep = new Report();
            rep.setTeachers(data.getTeachers());
            return rep.score(sched,this);
```

} : η οποία ουσιαστικά γυρνάει το score για το συγκεκριμένο πρόβλημα.

public ArrayList<Schedule> createChildren(int beams):

η συγκεκριμένη κλάση χρησιμοποιώντας την μέθοδο getTheProblem() της κλάσης Report προσπαθεί να αντιμετωπίσει το κριτήριο που της επιστρέφει δημιουργώντας παιδιά κάθε φορά ανάλογα με το κριτήριο που χρήζει αντιμετώπισης. Για παράδειγμα, αν έχουμε ένα κενό αντιμεταθέτουμε αυτήν την ώρα διδασκαλίας με όλες τις τελευταίες ώρες της βδομάδας ελπίζοντας να φτιάξουμε ένα καλύτερο πρόγραμμα διδασκαλίας, για κάποιο άλλο κριτήριο αντιστοιχία περνούμε την προβληματική ώρα και την κάνουμε αντιμεταθέσεις με άλλες πιθανές ώρες, στην περίπτωση όμως που ένας καθηγητής ξεπερνάει τον μέγιστο αριθμό ωρών διδασκαλίας της εβδομάδας του εκτός από τις αντιμεταθέσεις ψάχνουμε και για άλλους καθηγητές που ίσως να κάνουν το ίδιο μάθημα. Τέλος, από όλα τα παιδιά που θα έχουμε δημιουργήσει θα επιστρέψουμε (στην main) τα N παιδιά εκείνα με την χαμηλότερη βαθμολογία που όμως δεν θα έχουνε την ίδια βαθμολογία μεταξύ τους ώστε να υπάρχει ποικιλία στα παιδιά. Αν υπάρχουν πολλά παιδιά με τις ίδιες βαθμολογίες υπάρχει περίπτωση να επιστρέφει και λιγότερα από N παιδιά. (οπού N τα beams του BeamSearch αλγόριθμου).

-Σε οτι αφορα την Main:

Στο τέλος οταν το πρόγραμμα βρεί παιδί με score 0 γυρνάει σε μορφή pdf το ωρολόγιο πρόγραμμα όλου του γυμνασίου το οποίο ικανοποιεί όλους τους περιορισμούς.Σε περίπτωση που περάσει ο χρόνος αναμονής που έχουμε υποδείξει και δεν έχει βρεί "τέλειο" παιδί(score 0) επιστρέφει το παιδί εκείνο το οποίο είχε το καλύτερο score μέχρι τότε και το εμφανίζει σε μορφή pdf.

-Η κλάση ScheduleToPdf είναι η κλάση η οποία, χρησιμοποιώντας το .jar αρχείο της βιβλιοθήκης "pdfBox", εξάγει το πρόγραμμα διδασκαλίας σε μορφή PDF.

-Τρόπος βαθμολογίας ενός προγράμματος διδασκαλίας:

Το καλύτερο score για ένα πρόγραμμα διδασκαλίας είναι το 0, άρα ένα πρόγραμμα παίρνει μεγαλύτερη βαθμολογία ανάλογα με τα κριτήρια που δεν έχει ικανοποιήσει.

Αποφασίσαμε να βάλουμε διαφορετικά βάρη βαθμολογίας στο κάθε κριτήριο που δεν ικανοποιείτε ώστε ο αλγόριθμος σταδιακά να επιλεγεί το πρόγραμμα διδασκαλίας με την μικρότερη δυνατή βαθμολογία έχοντας πρώτα επιλύσει τα προβλήματα με την μεγαλύτερη προτεραιότητα.

Για παράδειγμα τα παρακάτω είναι τα score που ζημιώνετε ένα πρόβλημα για κάθε ανικανοποίητο κριτήριο:

1)Να μην υπάρχουν κενά.

Ποινή : + 1000000000/κενό σε λάθος σημείο.

2)Να μην διδάσκει ο καθηγητής ταυτόχρονα σε δυο η περισσότερες τάξεις (η τμήματα).

Ποινή : + 10000000/κάθε καθηγητή που διδάσκει ταυτόχρονα σε δυο τμήματα.

3)Να μη διδάσκει κανένας καθηγητής περισσότερες από δύο συνεχόμενες ώρες.

Ποινή : + 100000/για κάθε 3 συνεχόμενες ώρες που διδάσκει ο ίδιος καθηγητής.

4)Να μην υπερβαίνει ένας καθηγητής τον μέγιστο αριθμό ωρών ανά βδομάδα.

Ποινή : + 1000/για κάθε καθηγητή που υπερβαίνει το MaxHoursPerWeek του.

5)Να μην υπερβαίνει ένας καθηγητής τον μέγιστο αριθμό ωρών ανά μέρα.

Ποινή : + 100/για κάθε καθηγητή που υπερβαίνει το MaxHoursPerDay του.

6)Οι ώρες διδασκαλίας κάθε μαθήματος σε ένα τμήμα να είναι κατά το δυνατόν ομοιόμορφα κατανεμημένες σε όλες τις ήμερες της εβδομάδας.

Ποινή : + 1/μάθημα που διδάσκονται όλες οι ώρες του, ή οι περισσότερες, σε μια μέρα.(εξαιρούνται όσα είναι μια ώρα την βδομάδα.)

-Παράδειγμα Score:

Ένα πρόγραμμα διδασκαλίας που έχει score = 252301814, θα έχει :

-2 αστοχίες στο κριτήριο 1.

-5 αστοχίες στο κριτήριο 2.

-23 αστοχίες στο κριτήριο 3.

-1 αστοχία στο κριτήριο 4.

-8 αστοχίες στο κριτήριο 5.

-1 αστοχίες στο κριτήριο 6.

(φυσικά το συγκεκριμένο είναι ένα πολύ κακό score)

ΠΑΡΑΔΕΙΓΜΑΤΑ ΧΡΗΣΗΣ ΚΑΙ ΠΕΙΡΑΜΑΤΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

Το πρόγραμμα μας διαμορφώνει ένα τελικό score μετά το πέρας του χρόνου αναμονής ή επιστρέφει score=0

- Για αριθμό καταστάσεων **numbOfBeams** = 1 και χρόνο αναμονής **seconds** = 90

  1) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 9,7sec
  2) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 8,6 sec
  3) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 3sec
  4) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 11 sec
  5) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 3,1 sec
  6) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 10,4 sec
  7) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 3,2 sec
  8) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 11,6 sec
  9) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 1,8 sec
  10) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 3 sec

- Για αριθμό καταστάσεων **numbOfBeams** = 3 και χρόνο αναμονής **seconds** = 90.

  1) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 5,7sec
  2) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 10,6 sec
  3) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 10,6sec
  4) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 8,8 sec
  5) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 24,1 sec
  6) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 11,3 sec
  7) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 17,5 sec
  8) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 46,7 sec
  9) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 27,3 sec
  10) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 5 sec

- Για αριθμό καταστάσεων **numbOfBeams** = 4 και χρόνο αναμονής **seconds** = 90.

    1) **score =** 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 75 sec

    2) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 16 sec

    3) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 28,5sec

    4) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 22,5 sec

    5) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 8,166 sec

    6) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 7.837 sec

    7) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 35.854 sec

    8) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 19.4 sec

    9) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 9.5 sec

    10) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 15.1 sec

- Για αριθμό καταστάσεων **numbOfBeams** = 6 και χρόνο αναμονής **seconds** = 90.

    1) **score =** 1

    2) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 74,1 sec

    3) **score** = 2

    4) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 13 sec

    5) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 21,7 sec

    6) **score** =1

    7) **score** = 1

    8) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 38,4 sec

    9) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 39,4 sec

    10) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 50,2 sec

- Για αριθμό καταστάσεων **numbOfBeams** = 10 και χρόνο αναμονής **seconds** = 90.

    1) **score =** 2

    2) **score** =1

    3) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 28,4sec

    4) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 32,2 sec

    5) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 53,4 sec

    6) **score** = 1

    7) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 32,2 sec

    8) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 60,2 sec

    9) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 62,1 sec

    10) **score** = 0 και χρόνος εύρεσης της τέλεια λύσης ειναι 26,6 sec

Έχουμε συμπεριλάβει ένα ενδεικτικό αρχείο pdf με τέλειο πρόγραμμα διδασκαλίας για όλες τις τάξεις του γυμνασίου.

Το όποιο είναι,για numbOfBeams = 3 το "Example1.pdf".

Σημείωση: Παρατηρήσαμε ότι όσο μικρότερο είναι το numbOfBeams, τόσο μικρότερος είναι και ο χρόνος εύρεσης της τελείας λύσης (score 0). Αντίθετα όσο μεγαλύτερο είναι το numbOfBeams,ο χρόνος εύρεσης της τελείας λύσης αυξάνεται και όσο μεγαλώνει το numbOfBeams πολλές φόρες χρειάζεται και περισσότερο χρόνο από τον μέγιστο που του δίνεται (default time = 90sec.) για να βρει τελεία λύση. Ενώ λοιπόν παρατήσουμε ότι με numbOfBeams = 1 (δηλαδή ως Hill Climping) έχουμε τους μικρότερους χρόνους ως προς την εύρεση της τελείας λύσης, αποφασίσαμε το πρόγραμμα να τρέχει by default με numbOfBeams = 3, ώστε να υλοποιείται η εργασία με τον αλγόριθμο beam search, καθώς δίνεται η δυνατότητα να τρέξει και ως Hill Climping search αν αλλάξουμε το numbOfBeams σε 1. Επίσης, με τον αλγόριθμο beam search, δίνουμε την δυνατότητα επέκτασης του προγράμματος, αν για παράδειγμα προστεθούν και άλλα επιπλέον κριτήρια.