ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

5ο εξάμηνο

2ηεργασία – Ικανοποιησιμότητα προτάσεων στην προτασιακή λογική

Ικανοποιησιμότητα προτάσεων (Boolean satisfiability)

Έστω ένα σύνολο N προτασιακών συμβόλων (*propositional symbols*) P_1, P_2, …, P_N, κάθε μία από τις οποίες μπορεί να είναι είτε αληθής (True), είτε ψευδής (False). Μας δίνεται ένα σύνολο διαζεύξεων (*clauses*) C_1, C_2, …, C_M, κάθε μία από τις οποίες είναι της μορφής:

$$C_j = L_{j_1} \vee L_{j_2} \vee \ldots \vee L_{j_K}$$

όπου *K* το πλήθος των όρων κάθε διάζευξης (έστω το ίδιο για όλες τις διαζεύξεις). Κάθε όρος L_{j_k}της διάζευξης C_j ισούται είτε με κάποιο προτασιακό σύμβολο P_i, είτε με την άρνησή του, δηλαδή:

$$L_{j_k}=P_i \text{ ή } L_{j_k}=\neg P_i, \text{ για κάποιο } i.$$

Για παράδειγμα, έστω τα προτασιακά σύμβολα P_1, P_2, P_3, P_4 και P_5 και οι M=7 διαζεύξεις (για *K*=3):

$$C_1 = P_1 \vee P_3 \vee \neg P_4$$

$$C_2 = P_2 \vee P_4 \vee P_5$$

$$C_3 = \neg P_1 \vee P_2 \vee \neg P_5$$

$$C_4 = P_1 \vee \neg P_2 \vee P_4$$

$$C_5 = \neg P_3 \vee \neg P_4 \vee P_5$$

$$C_6 = \neg P_2 \vee \neg P_3 \vee \neg P_4$$

$$C_7 = P_1 \vee P_2 \vee \neg P_3$$

Το πρόβλημα της ικανοποιησιμότητας συνίσταται στην εύρεση μιας ανάθεσης τιμών στα προτασιακά σύμβολα P_1 έως P_N, έτσι ώστε να αληθεύουν όλες οι διαζεύξεις C_j. Στο παραπάνω παράδειγμα, μια τέτοια ανάθεση τιμών είναι η εξής:

$$P_1=\text{True}, P_2=\text{True}, P_3=\text{False}, P_4=\text{False}, P_5=\text{True}$$

Στο συγκεκριμένο πρόβλημα υπάρχουν άλλες 7 λύσεις. Γενικότερα πάντως, προβλήματα σαν και αυτό μπορεί να έχουν καμία, μία ή περισσότερες λύσεις.

Υπάρχουν πολλοί τρόποι να λυθούν τέτοια προβλήματα. Παρακάτω σας περιγράφονται και σας δίνονται υλοποιημένοι δύο τέτοιοι τρόποι. Σας ζητείται να τους συγκρίνετε και να καταγράψετε τα αποτελέσατα σας.

1η προσέγγιση: Επίλυση με αναρρίχηση λόφων

Έστω ο εξής άπληστος αλγόριθμος για την επίλυση του προβλήματος της ικανοποιησιμότητας τύπων Bool (Boolean Constraint Satisfaction Problem):

Αλγόριθμος 1: Αναρρίχηση λόφων με επανεκινήσεις (hill climbing)

1. Ανέθεσε τυχαία σε κάθε προτασιακό σύμβολο P_i μια τιμή μεταξύ των True και False. 2. Έστω T_0 το πλήθος των διαζεύξεων C_j που δεν ικανοποιούνται από την τρέχουσα ανάθεση. a. Εάν T_0=0, τότε επέστρεψε ως λύση την τρέχουσα ανάθεση τιμών. Τερμάτισε. 3. T=T_0, x=0.
4. Για i=1 έως N
    a. Έστω R το πλήθος των διαζεύξεων C_j που δεν θα ικανοποιούνταν εάν άλλαζε η τιμή του προτασιακού συμβόλου P_i (από True σε False ή αντίστροφα).
    b. Εάν R<T τότε
        i. T=R, x=i.
5. Εάν T<T_0 τότε
    a. άλλαξε την τιμή του προτασιακού συμβόλου P_x.
    b. Εάν T=0 τότε
        i. Επέστρεψε ως λύση την τρέχουσα ανάθεση τιμών. Τερμάτισε.
    c. αλλιώς
        i. πήγαινε στο βήμα 2.
6. αλλιώς
    a. Πήγαινε στο βήμα 1

Η λογική του παραπάνω αλγορίθμου είναι η εξής: Αρχικά αναθέτουμε στα προτασιακά σύμβολα P_i τυχαίες τιμές (True ή False). Μετράμε πόσες διαζεύξεις C_j δεν ικανοποιούνται. Στη συνέχεια επιλέγουμε εκείνη την αλλαγή τιμής κάποιου προτασιακού συμβόλου P_i (από True σε False ή από False σε True, αναλόγως ποια είναι η τρέχουσα τιμή του), η οποία οδηγεί στη μεγαλύτερη μείωση του πλήθους των διαζεύξεων C_j που δεν ικανοποιούνται. Η διαδικασία αυτή συνεχίζεται μέχρι να ικανοποιηθούν όλες οι διαζεύξεις C_j, οπότε επιστρέφουμε την τρέχουσα ανάθεση τιμής στα P_i ως λύση. Εάν δεν καταστεί δυνατή η ικανοποίηση όλων των διαζεύξεων, δηλαδή αν φθάσουμε σε μια ανάθεση τιμών στα P_i όπου δεν έχουν ικανοποιηθεί όλες οι C_j, ενώ παράλληλα δεν υπάρχει τρόπος να ικανοποιηθούν περισσότερες εξ αυτών με αλλαγή της τιμής κάποιου εκ των P_i, τότε ξεκινάμε από την αρχή, επιλέγοντας μια άλλη τυχαία αρχική ανάθεση τιμών σε όλα τα P_i.

Σε περίπτωση μη ύπαρξης λύσης, δεν υπάρχει αυστηρό κριτήριο τερματισμού του αλγορίθμου 1. Δυνητικά ο αλγόριθμος θα μπορούσε να εκτελείται για πάντα, χωρίς να βρίσκει λύση (ακόμη και αν υπάρχει!). Ωστόσο, επειδή ενδιαφερόμαστε να πάρουμε αποτελέσματα σε πεπερασμένο χρονικό διάστημα, μπορούμε να θέσουμε ένα χρονικό όριο, π.χ. 1 min, για την επίλυση κάθε προβλήματος.

Ο παραπάνω αλγόριθμος έχει υλοποιηθεί στο πρόγραμμα `bcsp.c` (σας δίνεται) στη γλώσσα προγραμματισμού C. Για να λύσετε ένα πρόβλημα, αφού μεταγλωττίσετε τον πηγαίο κώδικα (έστω `bcsp.exe` το παραγόμενο εκτελέσιμο), το καλείται ως εξής:

```
bcsp.exe hill input-file output-file
```

όπου `hill` είναι το όνομα του αλγορίθμου που θα χρησιμοποιηθεί για να λυθεί το πρόβλημα, `input-file` είναι ένα αρχείο με την περιγραφή του προβλήματος και `output-file` είναι ένα αρχείο στο οποίο θα εγγράφεται η λύση του προβλήματος. Για το παράδειγμα της εκφώνησης, το αρχείο `input-file` θα έπρεπε να έχει τα εξής περιεχόμενα:

```
5 7 3
1 3 -4
2 4 5
-1 2 -5
1 -2 4
-3 -4 5
-2 -3 -4
1 2 -3
```

όπου στην πρώτη σειρά υπάρχουν οι τιμές των παραμέτρων N, M και K (οι τιμές αυτών των παραμέτρων ορίζονται δυναμικά κατά την εκτέλεση του προγράμματός σας), ενώ κάθε μία από τις επόμενες M σειρές περιλαμβάνει K αριθμούς, κάθε ένας από τους οποίους είναι ένας ακέραιος στα διαστήματα [-N,1] ή [1,N]. Η απόλυτη τιμή αυτών των αριθμών δηλώνει ποιο είναι το αντίστοιχο προτασιακό σύμβολο κάθε διάζευξης ενώ η ύπαρξη αρνητικού προσήμου δηλώνει ότι το αντίστοιχο προτασιακό σύμβολο έχει άρνηση. Έτσι για παράδειγμα η τριάδα `1 3 -4` αντιστοιχεί στη διάζευξη $P_1 \vee P_3 \vee \neg P_4$.

Το αρχείο `output-file` περιλαμβάνει μια μόνο σειρά με μια τιμή -1 ή +1 για κάθε προτασιακό σύμβολο, με

την τιμή -1 να αντιστοιχεί στην τιμή False και την τιμή +1 στην τιμή True. Έτσι, εφόσον βρεθεί η λύση:

P_1=True, P_2=True, P_3=False, P_4=False, P_5=True

το αρχείο `output-file` περιλαμβάνει τις τιμές:

```
1 1 -1 -1 1
```

Σημειώνεται ότι ο αλγόριθμος που περιγράφηκε είναι στοχαστικός. Ως στοχαστικοί χαρακτηρίζονται οι αλγόριθμοι οι οποίοι εμπεριέχουν στοιχείο τύχης, οπότε το αποτέλεσμα της εκτέλεσής τους στο ίδιο πρόβλημα δεν είναι πάντα το ίδιο (για αυτό και ζητείται να εκτελεστεί ο αλγόριθμος περισσότερες από μια φορές σε κάθε πρόβλημα). Στην περίπτωσή μας η στοχαστικότητα προκύπτει από το γεγονός ότι αρχικά ανατίθενται τυχαίες τιμές στις προτάσεις P_i (βήμα 1), με αποτέλεσμα κάθε επανεκκίνηση να οδηγεί σε διαφορετικό αποτέλεσμα (διαφορετική λύση ή αδυναμία επίλυσης). Οι στοχαστικοί αλγόριθμοι έχουν την ιδιότητα ότι με περισσότερες δοκιμές μεγαλώνει και η πιθανότητα εύρεσης λύσης, ωστόσο ποτέ δεν είναι απολύτως βέβαιο ότι θα βρεθεί λύση σε πεπερασμένο χρονικό διάστημα, ακόμη και αν υπάρχει. Τέλος δεν μπορούν ποτέ να αποδείξουν ότι δεν υπάρχει λύση ακόμη και αν δεν υπάρχει!

2η προσέγγιση: Επίλυση με αναζήτηση πρώτα σε βάθος

Το πρόβλημα της ικανοποιησιμότητας τύπων Bool μπορεί να λυθεί με χρήση του αλγορίθμου αναζήτησης πρώτα σε βάθος (depth-first search), αναθέτοντας διαδοχικά τιμές στα προτασιακά σύμβολα και προσέχοντας ώστε κάθε διάζευξη για την οποία έχουν ανατεθεί τιμές σε όλα τα προτασιακά της σύμβολα να καθίσταται αληθής. Ο ψευδοκώδικας του αλγορίθμου μπορεί να γραφεί ως εξής:

Αλγόριθμος 2: Αναζήτηση κατά βάθος (depth-first search)

1. Έστω μια κενή στοίβα. Τοποθέτησε τον «κενό» κόμβο στη στοίβα.
2. Όσο η στοίβα δεν είναι άδεια.
    a. Βγάλε το πρώτο στοιχείο της στοίβας, έστω V.
    b. Για κάθε έγκυρο παιδί[1] V_i του V:
        i. Εάν το V_i αποτελεί λύση[2]
            1. εμφάνισέ την και τερμάτισε
        ii. αλλιώς
            1. πρόσθεσε το V_i στην κορυφή της στοίβας.
3. Ανέφερε ότι δεν υπάρχει λύση.

Ο παραπάνω ψευδοκώδικας τυπικά δεν κατασκευάζει το δένδρο αναζήτησης, μιας και η μοναδική δομή δεδομένων που χρησιμοποιεί είναι η στοίβα. Ωστόσο, επειδή κάθε κόμβος που δημιουργείται περιλαμβάνει όλη την πληροφορία όσον αφορά τις αναθέσεις τιμών αληθείας στα προτασιακά σύμβολα του προβλήματος, όταν θα συναντήσουμε μια λύση δεν θα χρειαστεί να ακολουθήσουμε προς τα πίσω τη διαδρομή προς τη ρίζα.

[1] Τα παιδιά του V προκύπτουν από τις δύο δυνατές αναθέσεις τιμών στο **επόμενο** προτασιακό σύμβολο. Ένα παιδί είναι έγκυρο αν με την τρέχουσα μερική ανάθεση τιμών στα προτασιακά σύμβολα δεν προκύπτει διάζευξη που είναι σίγουρα ψευδής.

[2] Για να αποτελεί λύση ένας κόμβος του δένδρου αναζήτησης, θα πρέπει να έχουν αποδοθεί τιμές (αληθές ή ψευδές) σε όλα τα προτασιακά σύμβολα και όλες οι διαζεύξεις να ικανοποιούνται.

Το πρόγραμμα bscp.c που σας δόθηκε υλοποιεί και τον τον παραπάνω αλγόριθμο. Καλέστε το πρόγραμμα με:

```
bcsp.exe depth input-file output-file
```

όπου ισχύουν οι ίδιες προδιαγραφές για τα `input-file` και `output-file`.

ΖΗΤΟΥΜΕΝΑ

Α) (1 μονάδα)

Επιλέξτε μια τιμή για το K (μεγαλύτερη του 3) και δοκιμάστε να λύσετε προβλήματα διαφόρων μεγεθών, για διάφορες τιμές των N και M (θα μπορούσατε να κρατήσετε σταθερό το N, π.χ., N=10, και να δοκιμάσετε να λύσετε προβλήματα για διάφορες τιμές του M).

Στόχος είναι να βρείτε ποια είναι η κρίσιμη τιμή του λόγου M/N, δηλαδή η τιμή εκείνη για την οποία η πλειοψηφία των προβλημάτων από ικανοποιήσιμα γίνεται μη-ικανοποιήσιμα. Για την ίδια τιμή του λόγου M/N, ο χρόνος επίλυσης είναι ο μεγαλύτερος δυνατός.

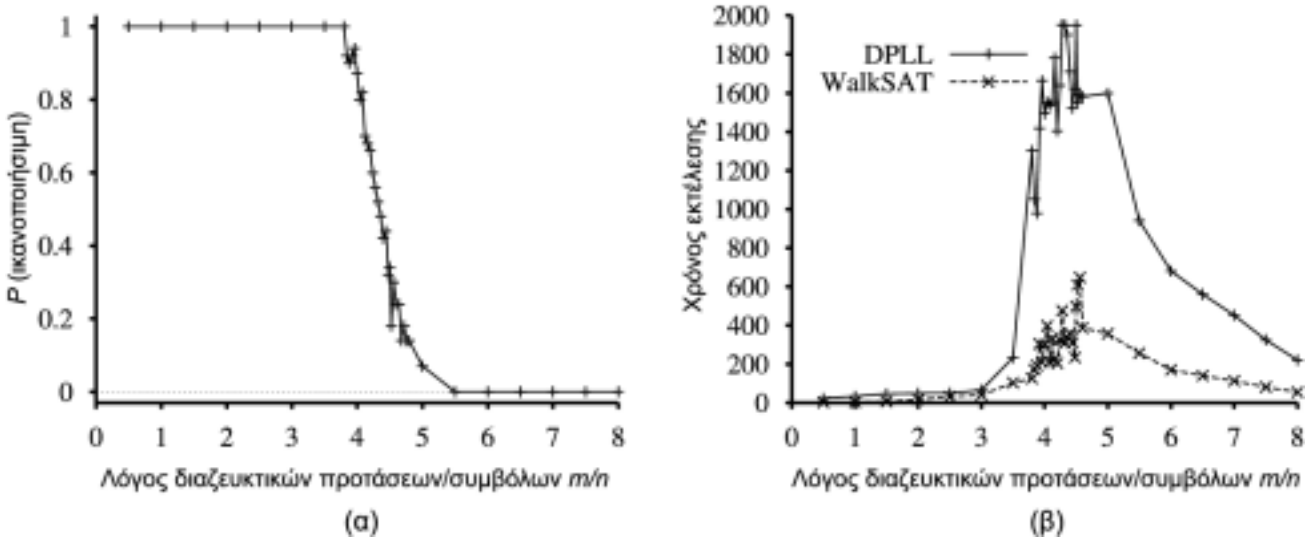
Για να βρείτε την τιμή του λόγου αυτού, κατασκευάστε δύο διαγράμματα:

- Το διάγραμμα του ποσοστού των επιλύσιμων προβλημάτων, ως συνάρτηση της τιμής του λόγου M/N.
- Το διάγραμμα του μέσου χρόνου που απαιτείται για την επίλυση των (επιλύσιμων) προβλημάτων, για κάθε έναν από τους δύο αλγορίθμους, ως συνάρτηση της τιμής του λόγου M/N .

Υποδείξεις: Για κάθε τιμή του λόγου M/N δοκιμάστε να λύσετε 10 προβλήματα τουλάχιστον. Από το πόσα προβλήματα λύθηκαν (έστω και από έναν αλγόριθμο – με την αναρρίχηση λόφων μπορείτε να δοκιμάστε να λύσετε πολλές φορές το ίδιο πρόβλημα μέχρις ότου ενδεχομένως λυθεί) μπορείτε να εκτιμήσετε το ποσοστό των επιλύσιμων προβλημάτων (πρώτο ζητούμενο διάγραμμα). Όσον αφορά το χρόνο επίλυσης (δεύτερο διάγραμμα), κάθε ένα από τα επιλύσιμα προβλήματα που εντοπίσατε στο προηγούμενο ερώτημα για δεδομένη τιμή του λόγου M/N θα πρέπει να επιχειρήσετε να το λύσετε:

- μια φορά με τον αλγόριθμο πρώτα σε βάθος
- πέντε φορές με τον αλγόριθμο αναρρίχησης λόφων και να υπολογίσετε τον μέσο όρο χρόνου

Στις παρακάτω δύο εικόνες[3] φαίνονται τα ζητούμενα διαγράμματα για K=3.



[3] Από το βιβλίο «Τεχνητή Νοημοσύνη, μια σύγχρονη προσέγγιση», 2η έκδοση (ελληνική μετάφραση), εικόνα 7.18.

Β) (επιπλέον 1 μονάδα, 0.5 ανά ερώτημα)

α) Βελτιώστε τον αλγόριθμο της αναζήτησης πρώτα σε βάθος (ή γράψτε νέο από την αρχή) με τις τρεις βελτιώσεις που υποστηρίζει αλγόριθμος DPLL (https://en.wikipedia.org/wiki/DPLL_algorithm) και μετρήστε την απόδοση του στο ίδιο σετ προβλημάτων με αυτό του ζητούμενου Α.

β) Υλοποιήστε τον αλγόριθμο WalkSAT (https://en.wikipedia.org/wiki/WalkSAT) ή τον αλγόριθμο της προσομοιωμένης ανόπτησης (επιλέξτε όποιον από τους δύο θέλετε) και μετρήστε την απόδοση του στο ίδιο σετ προβλημάτων με αυτό του ζητούμενου Α.

Σας δίνονται τα εξής:

- Το πρόγραμμα `bcsp_generate.c` το οποίο μπορείτε να χρησιμοποιήσετε για να δημιουργήσετε τα προβλήματα στα οποία θα δοκιμάσετε το πρόγραμμά σας.

• Το πρόγραμμα `bcsp_validate.c`, το οποίο μπορείτε να χρησιμοποιήσετε για να ελέγξετε τις λύσεις που βγάζει το πρόγραμμά σας.

Οδηγίες υποβολής: Η εργασία θα πρέπει να υποβληθεί μέσω του Classroom μέχρι την προβλεπόμενη ημερομηνία. Για το ζητούμενο Α θα υποβάλλετε ένα έγγραφο κειμένου με περιγραφή των πειραμάτων σας και των αποτελεσμάτων σας, καθώς και τα αρχεία προβλημάτων που χρησιμοποιήσατε (όχι όμως και τα αρχεία των λύσεων). Για το ζητούμενο Β θα υποβάλλετε επιπρόσθετα τον κώδικα που γράψατε, τεκμηρίωση για αυτόν (τι και πώς το υλοποιήσατε), και τα αποτελέσματα που πήρατε.

Καλή Επιτυχία!!!