Πανεπιστήμιο Πειραιώς
Σχολή Τεχνολογιών Πληροφορικής και Επικοινωνιών
Τμήμα Ψηφιακών Συστημάτων
Δομές Δεδομένων 2019-2020 – 1η Εργασία

Χρήστος Δουλκερίδης Ορέστης Τελέλης

1 Περιγραφή

Πρόβλημα. Ένας πωλητής διαθέτει m διαφορετικά αντικείμενα προς πώληση, που δεικτοδοτούνται με $j = 0, \ldots, m-1$. Υπάρχουν n επίδοξοι αγοραστές, που δεικτοδοτούνται με $i = 0, \ldots, n-1$. Κάθε αγοραστής i επιθυμεί συγκεκριμένο υποσύνολο αντικειμένων $S_i \subseteq \{0, 1, \ldots, m-1\}$. Αποδίδει στο υποσύνολο αντικειμένων S_i αξία v_i, την οποία προτίθεται να πληρώσει. Ζητείται να αποφασιστεί το υποσύνολο αγοραστών στους οποίους ο πωλητής θα εκχωρήσει *ακριβώς* τα υποσύνολα αντικειμένων που ζητούν, έτσι ώστε: ***(i)*** κάθε αντικείμενο $j = 0, 1, \ldots, m-1$ να δοθεί το πολύ σε έναν αγοραστή, ***(ii)*** το άθροισμα των αξιών των αγοραστών που επιλέχθηκαν να είναι μέγιστο.

Παράδειγμα. Θεωρήστε $m = 5$ αντικείμενα και $n = 5$ αγοραστές με:

$$S_0 = \{0, 1\}, v_0 = 10, \quad S_1 = \{2, 3\}, v_1 = 15, \quad S_2 = \{1, 4\}, v_2 = 22,$$
$$S_3 = \{0, 1, 2\}, v_3 = 30, \quad S_4 = \{3, 4\}, v_4 = 35$$

Μία ***εφικτή λύση*** του προβλήματος συνίσταται στην επιλογή των αγοραστών 0 και 1. Εκχωρούνται τα αντικείμενα $S_0 \cup S_1 = \{0, 1, 2, 3\}$ και δεν υπάρχει δυνατότητα επιλογής κάποιου επιπλέον αγοραστή, δίχως να εκχωρηθεί για δεύτερη φορά κάποιο από τα ήδη εκχωρημένα αντικείμενα. Η ολική αξία της λύσης είναι 25. Στη ***βέλτιστη εφικτή λύση*** επιλέγονται οι αγοραστές 3 και 4, με ολική αξία 65.

2 Ζητούμενα Εργασίας

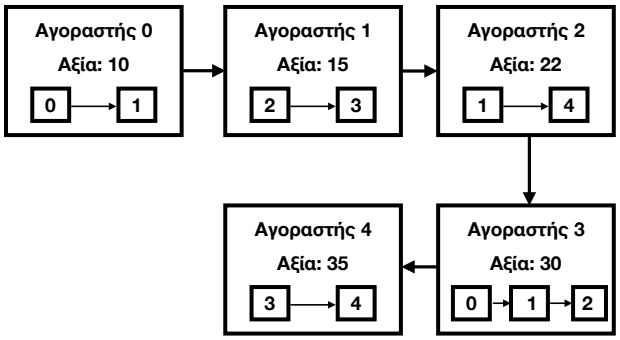
Η εργασία ζητά την υλοποίηση ενός *«άπληστου»* (greedy) αλγορίθμου και τον πειραματισμό με αυτόν, για την προσεγγιστική επίλυση του προβλήματος, όταν κάθε σύνολο S_i, $i = 0, 1, \ldots, n-1$ αναπαρίσταται με απλά συνδεδεμένη λίστα και το σύνολο όλων των αγοραστών αναπαρίσταται επίσης με απλά συνδεδεμένη λίστα. Δείτε το σχήμα 1 για σχηματική αναπαράσταση του αριθμητικού παραδείγματος. Ο άπληστος αλγορίθμος έχει ως εξής:

Είσοδος: δείκτες αντικειμένων $M = \{0, \ldots, m-1\}$, δείκτες αγοραστών $N = \{0, \ldots, n-1\}$, δεδομένα αγοραστών για $i = 0, 1, \ldots, n-1$: $S_i \subseteq M$, v_i

Έξοδος: υποσύνολο δεικτών αγοραστών $\mathcal{B} \subseteq N$

1. αρχικοποίησε $\mathcal{B} := \emptyset$
2. Επανάλαβε:
   1. εντόπισε $i \in N$ με $S_i \subseteq M$ και μέγιστο κλάσμα $v_i / |S_i|$
   2. ***αν*** $S_i \not\subseteq M$ για κάθε $i \in N$, ***επίστρεψε*** $\mathcal{B}$
   3. διάγραψε το i από το N, είσαγε το i στη λύση $\mathcal{B}$, θέσε $M := M \setminus S_i$

Στο αριθμητικό παράδειγμα, ο αλγόριθμος αυτός θα επιλέξει πρώτα τον αγοραστή 4 και στη συνέχεια τον αγοραστή 3, μεγιστοποιώντας πλήρως το άθροισμα των αξιών. Αυτό όμως δε συμβαίνει *για κάθε* είσοδο του προβλήματος. Δηλαδή, ο άπληστος αλγόριθμος δεν εγγυάται βέλτιστη λύση (μέγιστης αξίας). Για περισσότερες πληροφορίες, δείτε τις σημειώσεις στο τέλος της εκφώνησης.



Σχήμα 1: Σχηματική αναπαράσταση του αριθμητικού παραδείγματος με συνδεδεμένες λίστες.

2.1 Υλοποίηση

Θα υλοποιήσετε μεθόδους των κλάσεων `ItemsList` και `BuyersList`, που προδιαγράφονται στο *δεδομένο* αρχείο `BuyersList.java`. Στην υλοποίησή σας:

- ***Μπορείτε να προσθέσετε μόνο `private` πεδία ή μεθόδους.***
- ***Δε θα χρησιμοποιήσετε καθόλου arrays*** (με εξαίρεση στη `main`, εφόσον χρειάζονται).
- ***Δε θα χρησιμοποιήσετε `ArrayList` ή άλλες δομές βιβλιοθήκης της Java.***

2.1.1 Κλάσεις `ItemNode` και `BuyerNode`

Ορίζουν τον τύπο κόμβου των λιστών `ItemsList` και `BuyersList` αντίστοιχα. Η κλάση `ItemNode` περιέχει ένα `public` πεδίο `item` (τύπου `int`), που αποθηκεύει τον δείκτη ενός αντικειμένου στο πρόβλημα, και ένα `public` πεδίο `next` (κλάσης `ItemNode`) που αποθηκεύει αναφορά στον επόμενο κόμβο στη λίστα. Η κλάση `BuyerNode` περιέχει τα `public` πεδία `id` και `value` (τύπου `int`) που αντιστοιχούν στον δείκτη ενός αγοραστή και στην αξία που αποδίδει στο συγκεκριμένο υποσύνολο που επιθυμεί. Επίσης, περιέχει το `public` πεδίο `itemsList` (κλάσης `ItemsList`), που αποθηκεύει αναφορά στη λίστα που αναπαριστά το υποσύνολο αντικειμένων για το οποίο ενδιαφέρεται ο αγοραστής. Τέλος, περιέχει και ένα `public` πεδίο `next` (τύπου `BuyerNode`) που αποθηκεύει αναφορά στον επόμενο κόμβο στη λίστα τύπου `BuyersList`. Και οι δύο κλάσεις `ItemNode` και `BuyerNode` περιλαμβάνουν `public` κατασκευαστές (constructors).

2.1.2 Κλάση `ItemsList`

Ορίζει απλά συνδεδεμένη λίστα που αποθηκεύει ακεραίους (στο πρόβλημα, δείκτες αντικειμένων). Περιέχει `private` πεδία `first`, `last` (κλάσης `ItemNode`), που πρέπει να συντηρούνται ενδιαμέσως εισαγωγών/διαγραφών κόμβων και το πεδίο `nbNodes` (τύπου `int`) που μετρά το πλήθος κόμβων στη λίστα.

`public ItemsList()` Κατασκευάζει κενή λίστα κλάσης `ItemsList`.

`public int size()` Επιστρέφει το πλήθος κόμβων της λίστας.

`public boolean empty()` Επιστρέφει `true` αν η λίστα είναι κενή, διαφορετικά επιστρέφει `false`.

`public boolean contains(ItemsList ilist)` Επιστρέφει `true` αν η λίστα στην οποία καλείται η μέθοδος αναπαριστά υπερσύνολο του συνόλου που αναπαρίσταται από τη λίστα όρισμα `ilist`. Διαφορετικά επιστρέφει `false`.[1]

`public int append(int item)` Δημιουργεί κόμβο κλάσης `ItemNode`, στον οποίο εγγράφει τον δεδομένο ακέραιο `item`, και επισυνάπτει τον κόμβο στο τέλος της λίστας στην οποία καλείται η μέθοδος. Επιστρέφει το (ενημερωμένο) πλήθος κόμβων της λίστας.

`public void remove(ItemsList ilist)` Διαγράφει από τη λίστα στην οποία καλείται όλους τους κόμβους των οποίων τα στοιχεία/κλειδιά περιέχονται σε κόμβους της λίστας `ilist`.

[1] Υπόδειξη Java: Οι μέθοδοι μίας κλάσης έχουν πρόσβαση στα `private` πεδία όλων των αντικειμένων της κλάσης.

2.1.3 Κλάση `BuyersList`

Ορίζει απλά συνδεδεμένη λίστα, με έναν κόμβο για κάθε αγοραστή. Περιέχει `private` πεδία `first` και `last` (κλάσης `BuyerNode`) – αναφορές στον πρώτο και στον τελευταίο κόμβο της λίστας αντίστοιχα. Περιέχει ένα `private` πεδίο `nbNodes` (τύπου `int`), που μετρά το τρέχον πλήθος κόμβων της λίστας. Διαθέτει ένα `public` πεδίο `opt` τύπου `int`, που εξυπηρετεί εγγραφή/ανάγνωση αξίας της βέλτιστης λύσης του προβλήματος, που αναπαριστά η λίστα (εγγράφεται από τη μέθοδο `readFile` – δείτε παρακάτω).

`public BuyersList()` Κατασκευάζει κενή λίστα κλάσης `BuyersList`.

`public int readFile(String filename)` Διαβάζει αρχείο δεδομένων για το πρόβλημα και γράφει κόμβους στη λίστα κλάσης `BuyersList` στην οποία καλείται. Επίσης, γράφει στο `public` πεδίο `opt` της λίστας την αξία της βέλτιστης λύσης, για την είσοδο του προβλήματος που διαβάστηκε. Επιστρέφει το πλήθος m των διαφορετικών αντικειμένων του προβλήματος (με δείκτες $0, 1, \ldots, m-1$). Η μέθοδος είναι υλοποιημένη. Καλεί τις `append` των `ItemsList`, `BuyersList` που χρειάζεται να υλοποιήσετε.

`public int size()` Επιστρέφει το τρέχον πλήθος των κόμβων της λίστας.

`public boolean empty()` Επιστρέφει `true` αν η λίστα είναι κενή, διαφορετικά επιστρέφει `false`.

`public int totalValue()` Υπολογίζει και επιστρέφει την ολική (αθροιστική) αξία όλων των αγοραστών που αποθηκεύονται στη λίστα κλάσης `BuyersList`, στην οποία καλείται η μέθοδος.

`public int append(int id, int value, ItemsList ilist)` Δημιουργεί κόμβο `BuyerNode` που αποθηκεύει τον δείκτη (`id`), την αξία (`value`) και τη λίστα με αντικείμενα ενδιαφέροντος ενός αγοραστή (`ilist`), και τον επισυνάπτει στο τέλος της λίστας στην οποία καλείται η μέθοδος. Επιστρέφει το τρέχον μήκος της λίστας κατόπιν της επισύναψης.

`public BuyersList greedy(int m)` Υλοποιεί τον άπληστο αλγόριθμο προς επιλογή εφικτού υποσυνόλου αγοραστών με μέγιστη αθροιστική αξία. Δέχεται σαν όρισμα το πλήθος των αντικειμένων προς πώληση. Επιστρέφει λίστα κλάσης `BuyersList`, με το υποσύνολο των επιλεγμένων αγοραστών.

`public static void main(String[] args)` Υλοποιεί τον πειραματισμό που περιγράφεται παρακάτω.

2.2 Πειραματισμός και Τεχνική Αναφορά

Δίνονται 10 αρχεία δεδομένων, που περιγράφουν εισόδους διαφορετικών διαστάσεων του προβλήματος. Συγκεκριμένα: 5 αρχεία με προβλήματα διαστάσεων $m = 500$ και $n = 1000, 3000, 5000, 7000, 9000$ και 5 αρχεία με προβλήματα διαστάσεων $n = 2000$ και $m = 200, 400, 600, 800, 1000$. Για κάθε αρχείο, θα εκτελέσετε τον άπληστο αλγόριθμο τουλάχιστον 10 φορές και θα μετρήσετε τον μέσο χρόνο εκτέλεσης της μεθόδου `greedy`. Θα συγκρίνετε την αξία της λύσης που επιστρέφει, με την αξία της βέλτιστης λύσης που υπάρχει στο αρχείο και εγγράφεται στο πεδίο `opt` της λίστας `BuyersList`, από τη μέθοδο `readFile`. Με την εκτέλεση του προγράμματος, η `main` θα εκτυπώνει τα αποτελέσματα στην οθόνη, ως εξής:

```
* m = 500:
– n = 1000 avgTime = ..... greedy value = .... opt value = ....
.............................................................
* n = 2000:
– m = 200 avgTime = ..... greedy value = .... opt value = ....
.............................................................
```

Θα συντάξετε τεχνική αναφορά, όπου θα καταγράψετε και θα σχολιάσετε τα αποτελέσματα του πειραματισμού σας. Η αναφορά θα περιλαμβάνει πίνακες, έναν για κάθε ομάδα αρχείων δεδομένων ($m = 500$ και $n = 2000$), όπου θα καταγράφονται: ο μέσος χρόνος εκτέλεσης της μεθόδου `greedy` για κάθε αρχείο δεδομένων, η αξία της βέλτιστης λύσης και η αξία της λύσης που βρίσκει η μέθοδος `greedy`. Θα απεικονίσετε τους (μέσους) χρόνους εκτέλεσης της `greedy` σε ***δύο γραφικές παραστάσεις***, χρησιμοποιώντας στον οριζόντιο άξονα **(1)** τις διαφορετικές τιμές του n, για τα αρχεία με $m = 500$, και **(2)** τις διαφορετικές τιμές του m, για τα αρχεία με $n = 2000$. Θα περιγράψετε συνοπτικά την υλοποίηση της μεθόδου `greedy` και μεθόδων που αυτή χρησιμοποιεί, με στόχο τη θεωρητική δικαιολόγηση της πολυπλοκότητας της `greedy` ως συνάρτησης των m και n. Να σχολιάσετε αν και πώς επαληθεύεται η θεωρητική έκφραση της πολυπλοκότητας από τις γραφικές σας απεικονίσεις των πειραματικών εκτελέσεων της `greedy`.

3 Διαδικαστικά Θέματα

ΠΑΡΑΔΟΤΕΑ Θα πρέπει να παραδώσετε ένα αρχείο AM_Επώνυμο_Όνομα.zip (όπου AM ο αριθμός μητρώου) που θα περιλαμβάνει **δύο (2) αρχεία μόνο:**

1. Το αρχείο πηγαίου κώδικα **BuyersList.java**, επαρκώς σχολιασμένο.
   Προσοχή: σχόλια στα ελληνικά, μόνο σε κωδικοποίηση UTF-8. Άλλες κωδικοποιήσεις (ISO-8859-7, Windows-1253) *απορρίπτονται από τον Java Compiler*. Eclipse/Netbeans: δεξί click στο Project, επιλογή Properties και στην ιδιότητα «Encoding» επιλέγετε UTF-8.
2. Την τεχνική αναφορά σας **σε μορφή pdf**. **ΘΑ ΑΓΝΟΗΘΕΙ** *κάθε άλλη μορφή εγγράφου*.

ΠΑΡΑΔΟΣΗ αποκλειστικά μέσω της πλατφόρμας «ΕΥΔΟΞΟΣ» του τμήματος, έως και την **15/12/2019, 23:59**. Ανεβάστε το AM_Επώνυμο_Όνομα.zip στην περιοχή «Εργασίες».

ΕΡΩΤΗΣΕΙΣ/ΑΠΟΡΙΕΣ/ΔΙΕΥΚΡΙΝΙΣΕΙΣ αποκλειστικά μέσα από την Περιοχή Συζήτησης **«1η Εργασία (2019-20)»** της πλατφόρμας «ΕΥΔΟΞΟΣ». Δε θα απαντηθούν Email με απορίες.

ΔΕ ΘΑ ΑΞΙΟΛΟΓΗΘΟΥΝ:

- Εργασίες που θα παραδοθούν εκπρόθεσμα ή με άλλο τρόπο (email, CD κ.λ.π.)
- Εργασίες που παραδίδονται σε μορφή συμπίεσης διαφορετική από .zip.
- Εργασίες που περιλαμβάνουν μέσα άσχετα αρχεία.

ΒΑΘΜΟΛΟΓΗΣΗ Η εργασία μετρά 20% στον τελικό βαθμό (εφόσον το γραπτό σας βαθμολογηθεί με τουλάχιστον 4).

- Φτωχή / Δυσανάγνωστη τεκμηρίωση ή έλλειψη αυτής στην τεχνική αναφορά για τα ζητούμενα της εργασίας θα επιφέρει ανάλογη βαθμολογική επιβάρυνση.
- Η τεχνική αναφορά ή τμήματα αυτής δε θα ληφθούν υπόψην στη βαθμολόγηση αν δε συνοδεύονται από κώδικα που μεταγλωττίζεται και εκτελείται ορθά.
- **Η εργασία είναι αυστηρά ατομική.** Αντιγραφή στον κώδικα και/ή στην τεχνική αναφορά επιφέρει άμεσο μηδενισμό της εργασίας. Οι εργασίες θα ελεγχθούν (όλες ανά ζεύγη).

Σημειώσεις

Το πρόβλημα που περιγράφεται στην εργασία είναι γνωστό σαν *«Συσκευασία Συνόλου»* (*Set Packing*). Δεν υπάρχει γνωστός αλγόριθμος πολυωνυμικού χρόνου που υπολογίζει τη βέλτιστη λύση για κάθε είσοδο του προβλήματος. Όλοι οι γνωστοί αλγόριθμοι εύρεσης της λύσης μέγιστης ολικής αξίας έχουν εκθετική πολυπλοκότητα. Η θεωρία πολυπλοκότητας κατατάσσει το πρόβλημα αυτό στα *«υπολογιστικά δύσκολα»* προβλήματα.

Θεωρήστε την εξής είσοδο, με $n = 2$ αγοραστές και m αντικείμενα, που αναγκάζει τον άπληστο αλγόριθμο να επιλέξει εφικτή λύση με ολική αξία σχεδόν m φορές μικρότερη από την αξία της βέλτιστης λύσης. Έστω $S_0 = \{0\}$, $v_0 = 1 + \epsilon$, για κάποιο πολύ-πολύ μικρό $\epsilon > 0$. Για τον αγοραστή 1, έστω $S_1 = \{0, 1, \ldots, m-1\}, v_1 = m$. Το βέλτιστο υποσύνολο αγοραστών είναι το $\{1\}$ με ολική αξία m. Όμως ο άπληστος αλγόριθμος θα επιλέξει τον αγοραστή 0, με μέγιστο κλάσμα $v_0/|S_0| = (1+\epsilon)/1 = 1+\epsilon$. Η αξία $1+\epsilon$ που επιτυγχάνει ο αλγόριθμος είναι σχεδόν m φορές μικρότερη από τη βέλτιστη δυνατή αξία, για αρκετά μικρό ϵ. Είναι επίσης σχετικά εύκολο να αποδειχτεί, ότι η λύση που υπολογίζει ο άπληστος αλγόριθμος έχει πάντα αξία *τουλάχιστον ίση με κλάσμα* $1/m$ επί της αξίας της βέλτιστης λύσης.