Άπληστοι Αλγόριθμοι
(CLR, κεφάλαιο 17)

Στην ενότητα αυτή θα μελετηθούν τα εξής θέματα:

Σχεδιασμός αλγορίθμων με Άπληστους Αλγόριθμους

Στοιχεία άπληστων αλγορίθμων

Το πρόβλημα επιλογής εργασιών

Άπληστοι Αλγόριθμοι

- Προβλήματα στα οποία ζητείται μια βέλτιστη λύση (optimization problems) συχνά λύνονται από αλγόριθμους οι οποίοι ακολουθούν μια σειρά βημάτων με ένα σύνολο επιλογών σε κάθε βήμα.
- **Κύρια ιδέα Άπληστων Αλγορίθμων**: σε κάθε βήμα επέλεξε την επιλογή που φαίνεται καλύτερη για τη δεδομένη στιγμή (the locally optimal choice).
- Ποιο είναι το κατάλληλο κριτήριο για τη λήψη μιας τοπικά βέλτιστης επιλογής;
- Ερώτημα: αυτή η ακολουθία τοπικά βέλτιστων επιλογών δίνει τη βέλτιστη λύση του προβλήματος;
  - εξαρτάται από το πρόβλημα
  - για ορισμένα προβλήματα εγγυείται μόνο ΄κοντινή΄ (approximate) λύση

Άπληστοι Αλγόριθμοι

- Παραδείγματα άπληστων αλγόριθμων:
  1. οι αλγόριθμοι εύρεσης ελάχιστου δέντρου σκελετού του Kruskal και του Prim
  2. ο αλγόριθμος υπολογισμού των κωδικών Huffman
  3. ο αλγόριθμος επεξεργασίας εργασιών (job scheduling)

- Παραδείγματα προβλημάτων στα οποία δεν προτρέπεται η χρήση άπληστου αλγόριθμου
  1. προβλήματα στα οποία τα υποπροβλήματα δεν είναι ανεξάρτητα μεταξύ τους.
  2.

Επιλογή Εργασιών σύμφωνα με προθεσμίες

Το πρόβλημα

- Έχουμε ένα σύνολο εργασιών S={1,2,…n} κάθε μία από τις οποίες χρειάζεται μία μονάδα χρόνου για να ολοκληρωθεί. Κάθε εργασία έχει κάποια προθεσμία d_i και εκτέλεσή της πριν από την πάροδο της προθεσμίας της μας επιφέρει κέρδος g_i.
- Για τη συμπλήρωση των εργασιών έχουμε στη διάθεση μας ένα επεξεργαστή (ανά πάσα στιγμή ο επεξεργαστής είναι χρησιμοποιήσιμος από το πολύ μια εργασία).
- Ένα σύνολο εργασιών ονομάζεται *συμβατό* αν υπάρχει σειρά εκτέλεσης των εργασιών του η οποία ικανοποιεί τις προθεσμίες της κάθε εργασίας.
- **Στόχος:** η εύρεση του συνόλου συμβατών εργασιών που επιφέρει το μέγιστο δυνατό κέρδος.

Παράδειγμα

- Εργασίες:

i	1	2	3	4
g_i	50	10	15	30
d_i	2	1	2	1

- Σύνολα Συμβατών Εργασιών

Σύνολο Εργασιών	**1**	**2**	**3**	**4**	**1,3**	**2,1**	**2,3**	**4,1**	**4,3**
Κέρδος	50	10	15	30	65	60	25	80	45

- Πως μπορούμε να διαλέξουμε την πρώτη εργασία για δημιουργία του συμβατού συνόλου με το μέγιστο κόστος;
  1. αυτή με την πιο κοντινή προθεσμία;
  2. αυτή με το μεγαλύτερο κέρδος;

Ο Αλγόριθμος

- Υποθέτουμε ότι οι εργασίες είναι ταξινομημένες σε φθίνουσα σειρά ως προς το κέρδος που επιφέρουν. Αν όχι τις ταξινομούμε σε χρόνο O(n log n).
- Αρχικά επιλέγουμε την εργασία με το πιο μεγάλο κέρδος.
- Στη συνέχεια επιλέγουμε την επόμενη πιο κερδοφόρα εργασία η οποία είναι συμβατή με το σύνολο που έχουμε επιλέξει μέχρι στιγμής.

```
greedy {
 A={1};
 for (i=2; i<=n; i++){
        if legal(A∪{i})
        A=A∪{i};
 }
 return A;
}
```

- Η διαδικασία legal(B) αποφασίζει αν το σύνολο B είναι συμβατό.

Η διαδικασία legal

- Σε κάθε επανάληψη του βρόχου επιλέγεται η εργασία με το πιο μεγάλο κέρδος η οποία είναι συμβατή με την υπάρχουσα επιλογή.
- Πως μπορούμε να αποφασίσουμε κατά πόσο ένα σύνολο εργασιών είναι συμβατό;

Λήμμα 1: *Έστω σύνολο εργασιών με k εργασίες {1,2, ...,k} αριθμημένες έτσι ώστε οι προθεσμίες τους να βρίσκονται σε αύξουσα σειρά, δηλ.* $d_1 \leq d_2 \leq \ldots \leq d_n$.

Αν οι k εργασίες είναι συμβατές τότε κατά την εκτέλεσή τους σε σειρά ⟨1,2,...,k⟩ (δηλαδή διαλέγοντας κάθε φορά εκείνη με την πιο κοντινή προθεσμία) καμιά εργασία δεν χάνει την προθεσμία της.

Απόδειξη με αντίφαση:

- Έστω ότι η το σύνολο των εργασιών είναι συμβατό αλλά κατά την εκτέλεσή τους σε σειρά ⟨1,2,...,k⟩ κάποια εργασία χάνει την προθεσμία της.
- Έστω ότι η εργασία r είναι η πρώτη τέτοια εργασία. Τότε $d_r \leq r-1$.

Η διαδικασία legal

- Αφού οι εργασίες βρίσκονται σε αύξουσα σειρά προθεσμίας τότε οι εργασίες 1,...,r-1 έχουν επίσης προθεσμίες ≤r-1.
- Επομένως υπάρχουν τουλάχιστον r εργασίες με προθεσμίες ≤ r-1.
- Όπως και αν εκτελέσουμε αυτές τις r εργασίες κάποια από αυτές θα χάσει την προθεσμία της!
- Επομένως το σύνολο των εργασιών δεν είναι συμβατό. Αντίφαση!

- Για να ελέγχουμε τη συμβατότητα συνόλων εργασιών πρέπει
  1. να κρατούμε την ανά πάσα στιγμή επιλογή μας ταξινομημένη σε αύξουσα σειρά ως προς τον χρόνο προθεσμίας των εργασιών και
  2. κάθε φορά που θεωρούμε μια καινούρια εργασία να ελέγχουμε αν μπορεί να εκτελεστεί η εργασία χωρίς να αναγκάσει κάποια από τις υπάρχουσες να χάσει την προθεσμία της.

Ο Αλγόριθμος

```
greedy2(int d[n]){

int j[n];                  {ο πίνακας όπου φυλάγουμε την
                            επιλογή εργασιών}
int k;                     {μετρητής των επιλεγμένων εργασιών}

j[0] = 0;
j[1] = 1;                  {διάλεξε την πρώτη εργασία}
k = 1;

for (i=2; i<=n; i++){
   r = k;

   while (d_j[r] > max(d_i,r))   r--;
   if (d_i > r)
      for (m=k; m≥r+1; m--)
           j[m+1] = j[m];
      j[r+1] = i;
      k=k+1;
return j;
}
```

{βρες το σημείο όπου μπορεί να εισαχθεί η εργασία i}

αν η i μπορεί να εισαχθεί στη θέση r+1}

{μετακίνησε τις εργασίες από τη θέση r+1 και μετά μια θέση προς τα δεξιά και εισήγαγε την i}

Ο Αλγόριθμος

- Χρόνος Εκτέλεσης:

  Στη χείριστη περίπτωση κατά την i-οστή εκτέλεση του βρόχου χρειάζονται Θ(i) βήματα στο εσωτερικό while-loop και Θ(i) βήματα στο εσωτερικό for-loop). Επομένως ο χρόνος εκτέλεσης χειρίστης περίπτωσης είναι $\Theta(n^2)$.

- Παράδειγμα εκτέλεσης:

i	d_i	g_i	
1	4	70	⟨1⟩
2	2	60	⟨2, 1⟩
3	4	50	⟨2, 1, 3⟩
4	3	40	⟨2, 4, 1, 3⟩
5	1	30	⟨2, 4, 1, 3⟩
6	4	20	⟨2, 4, 1, 3⟩
7	6	10	⟨2, 4, 1, 3, 7⟩

Απόδειξη της ορθότητας του αλγόριθμου

Θεώρημα 2: Έστω Α η ακολουθία εργασιών που παράγεται ως δεδομένο εξόδου από τον αλγόριθμο. Τότε,

1. Η Α περιέχει συμβατές εργασίες, και
2. Η Α έχει το μέγιστο δυνατό κέρδος συνόλου που ικανοποιεί τις προδιαγραφές του προβλήματος.

Το πρώτο μέρος μπορεί να αποδειχθεί εύκολα με τη μέθοδο της επαγωγής.

Απόδειξη του 2:

Θα δείξουμε ότι υπάρχει βέλτιστη λύση στο πρόβλημα που περιέχει την εργασία 1, δηλαδή την εργασία με το μεγαλύτερο κέρδος.

Έστω σύνολο $B \subseteq S$ μια βέλτιστη λύση στο πρόβλημα.

Αν $1 \in B$ τότε η λύση περιέχει την άπληστη επιλογή.

Διαφορετικά, αφού το σύνολο εργασιών Β είναι συμβατό, σύμφωνα με το Λήμμα 1, οι εργασίες της Β μπορούν να εκτελεστούν σε αύξουσα σειρά προθεσμίας, χωρίς καμιά να χάσει την προθεσμία της.

Απόδειξη της ορθότητας του αλγόριθμου

Έστω η σειρά αυτή είναι η $\langle k \rangle B'$, δηλαδή η εργασία k είναι η εργασία στο B με την κοντινότερη προθεσμία.

Θεωρείστε την ακολουθία $A = \langle 1 \rangle B'$. Προφανώς η ακολουθία αυτή αντιστοιχεί σε ένα συμβατό σύνολο εργασιών (καμιά από τις εργασίες στο B΄ δεν πρόκειται να χάσει την προθεσμία της).

Επιπλέον, αφού

$g_k \leq g_1$,

τότε

το A είναι επίσης βέλτιστη λύση του προβλήματος.

Επομένως υπάρχει βέλτιστη λύση η οποία περιέχει την άπληστη επιλογή.

Μπορούμε να χρησιμοποιήσουμε τα ίδια επιχειρήματα και για την εργασία με το δεύτερο μεγαλύτερο κέρδος και ούτω καθεξής. (μαθηματική επαγωγή.)

Στοιχεία άπληστων αλγορίθμων

- Ένας άπληστος αλγόριθμος προσπαθεί να βρει τη βέλτιστη λύση σε ένα πρόβλημα εφαρμόζοντας μια ακολουθία επιλογών. Κάθε μια από τις επιλογές επιλέγει την καλύτερη λύση στη δεδομένη στιγμή σύμφωνα με κάποιο κριτήριο.
- Αυτή η μέθοδος δεν βρίσκει πάντα τη βέλτιστη λύση.
- Πως μπορούμε να αποφασίσουμε κατά πόσο ένα πρόβλημα μπορεί να λυθεί από ένα άπληστο αλγόριθμο;
- Γενικά δεν υπάρχει χαρακτηρισμός `άπληστων΄ προβλημάτων.
- Εντούτοις, οι δύο πιο κάτω ιδιότητες προβλημάτων συχνά εισηγούνται την καταλληλότητα της χρήσης άπληστου αλγόριθμου.
  1. Ιδιότητα της άπληστης επιλογής.
  2. Ιδιότητα της βέλτιστης υποδομής.

Στοιχεία άπληστων αλγορίθμων

Η Ιδιότητα της άπληστης επιλογής

- Μια ολικά βέλτιστη λύση μπορεί να βρεθεί με τη λήψη μιας τοπικά βέλτιστης απόφασης.
- Η λήψη μιας απόφασης δεν εξαρτάται από μελλοντικές αποφάσεις.
- Απόδειξη ικανοποίησης της ιδιότητας από κάποιο πρόβλημα χρειάζεται και επιχειρηματολογία της μορφής:

  αν Α είναι μια βέλτιστη λύση του προβλήματος, τότε μπορούμε να δημιουργήσουμε παραλλαγή της Α, Α΄, όπου η πρώτη απόφαση είναι άπληστη και η Α΄ εξακολουθεί να είναι βέλτιστη. Με τη μέθοδο της επαγωγής μπορούμε να δείξουμε ότι άπληστη επιλογή μπορεί να εφαρμοστεί σε κάθε βήμα του αλγόριθμου.

Η ιδιότητα της βέλτιστης υποδομής

- Μια βέλτιστη λύση του προβλήματος περιέχει βέλτιστες λύσεις υποπροβλημάτων.
- (Συγκρίνετε με τη μέθοδο του δυναμικού προγραμματισμού.)

Άπληστος αλγόριθμος ή δυναμικός προγραμματισμός;

Δίνοντας ρέστα (1)

- Το Αυστριακό νομισματικό σύστημα περιέχει νομίσματα των 1, 5, 10, 25 και 100 σελινιών.
- Να σχεδιασθεί άπληστος αλγόριθμος ο οποίος, θεωρώντας την ύπαρξη απεριόριστης ποσότητας όλων των νομισμάτων, για οποιοδήποτε ακέραιο X επιστρέφει τον ελάχιστο αριθμό νομισμάτων με συνολική αξία X.
- Για παράδειγμα, για X=46, ο αλγόριθμός σας θα πρέπει να επιστρέψει την τιμή 4 αφού η ελάχιστη συλλογή νομισμάτων αξίας 46 είναι η ⟨25, 10, 10,1⟩.

Δίνοντας ρέστα (2)

- Να εισηγηθείτε αλγόριθμο ο οποίος να λύνει το πρόβλημα δεδομένου ότι τα νομίσματα αξίας 5 σελινιών έχουν εξαντληθεί.

Δίνοντας Ρέστα

- Και τα δύο προβλήματα ικανοποιούν την ιδιότητα βέλτιστης υποδομής:
  1. Για το πρώτο πρόβλημα, αν για κάποια βέλτιστη επιλογή κερμάτων Ε αξίας Χ αφαιρέσουμε κάποιο κέρμα a αξίας x, $x \in \{1,5,10,25,100\}$, η επιλογή E-a είναι η βέλτιστη επιλογή κερμάτων αξίας X-x.
  2. Στο δεύτερο πρόβλημα, αν για κάποια βέλτιστη επιλογή κερμάτων Ε αξίας Χ αφαιρέσουμε κάποιο κέρμα a αξίας x, $x \in \{1,10,25,100\}$, η επιλογή E-a είναι η βέλτιστη επιλογή κερμάτων αξίας X-x.
- Μπορούν τα προβλήματα να λυθούν με άπληστο αλγόριθμό;
- Κριτήριο επιλογής:

  διάλεξε το πιο μεγάλο νόμισμα που χωρεί στην αξία που θέλεις να επιστρέψεις.

Δίνοντας Ρέστα

Δίνοντας ρέστα (1)

```
change (int x)
     A = 0;
     while(x>0){
         if x>=100 A=A + x div 100; x=x mod 100;
         if x>=25  A=A + x div 25;  x=x mod 25;
         if x>=10  A=A + x div 10;  x=x mod 10;
         if x>=5   A=A + x div 5;   x=x mod 5;
         if x>=1   A=A + x div 1;   x=x mod 1;
     return A;
```

Δίνοντας ρέστα (2)

```
change (int x)
     A = ∅;
     while(x>0)
         if x>=100 A=A + x div 100; x=x mod 100;
         if x>=25  A=A + x div 25;  x=x mod 25;
         if x>=10  A=A + x div 10;  x=x mod 10;
         if x>=1   A=A + x div 1;   x=x mod 1;
     return A;
```

Δίνοντας Ρέστα

- Επιτυγχάνουν το στόχο τους οι δύο αλγόριθμοι; δηλ. ελαχιστοποιούν τον αριθμό νομισμάτων της επιλογής που επιστρέφεται;
- Για το πρώτο πρόβλημα ναι, μπορούμε να το αποδείξουμε με παρόμοιο τρόπο όπως και το πρόβλημα επιλογής εργασιών.
- Αντιπαράδειγμα για το δεύτερο αλγόριθμο:

  Έστω ότι η αξία που θέλουμε να επιστρέψουμε είναι 30: Με τον άπληστο αλγόριθμο επιλέγονται νομίσματα

  $\langle 25, 1, 1, 1, 1, 1 \rangle$

  ενώ βέλτιστη λύση είναι η επιλογή:

  $\langle 10, 10, 10 \rangle$

- Γιατί δουλεύει ο αλγόριθμος για το πρώτο και όχι για το δεύτερο πρόβλημα;
- Πως μπορεί να λυθεί το δεύτερο πρόβλημα;