Αλγόριθμοι και Πολυπλοκότητα

Ν. Μ. Μισυρλής

Τμήμα Πληροφορικής και Τηλεπικοινωνιών,

Πανεπιστήμιο Αθηνών

Άπληστοι Αλγόριθμοι

Εισαγωγή

- Ένας άπληστος αλγόριθμος (greedy algorithm) κάνει σε κάθε βήμα την επιλογή που φαίνεται βέλτιστη.
- Δεν είναι απαραίτητο ότι θα οδηγήσει στην βέλτιστη λύση του προβλήματος.
- Η ''άπληστη'' επιλογή είναι δυνατόν να οδηγήσει σε μία κακή λύση.
- Υπάρχουν άπληστοι αλγοριθμοι που οδηγούν στην βέλτιστη λύση του προβλήματος (Prim, Kruskal και Dijkstra).
- Θα μελετήσουμε την γενική μορφή ενός άπληστου αλγορίθμου και θα δούμε την εφαρμογή του σε προβλήματα που είτε εφαρμόζεται αποδοτικά (βρίσκεται η βέλτιστη λύση) είτε όχι.

Άπληστοι Αλγόριθμοι

Γενική μορφή ενός Άπληστου Αλγόριθμου

- Μία λύση στο πρόβλημα:ένα υποσύνολο $F \subseteq E$, $C(F) = true$.
- F είναι μία εφικτή λύση (feasible solution).
- Ο στόχος θα είναι να βρεθεί εκείνο το F που είναι βέλτιστο μεταξύ των υποσυνόλων του E, δηλαδή το

$$\sum_{e \in F} v(e)$$

να είναι μέγιστο ή ελάχιστο.

Άπληστοι Αλγόριθμοι

Γενική μορφή ενός Άπληστου Αλγόριθμου

GREEDY (E : σύνολο στοιχείων)

1. $F = \emptyset$
2. **while** $E \neq \emptyset$ **do**
3. Άπληστη Επιλογή $x \in E$
4. $E = E - \{x\}$
5. **if** $F \cup \{x\}$ feasible **then**
6. $F = F \cup \{x\}$
7. **end if**
8. **end while**
9. return F

Άπληστοι Αλγόριθμοι

Ο αλγόριθμος του Kruskal

- Εύρεση ενός δέντρου επικάλυψης ελαχίστου κόστους σε μη κατευθυνόμενο γράφο με βάρη $G = (V, E)$.
- Σε αυτό το πρόβλημα η άπληστη επιλογή οδηγεί στην βέλτιστη λύση.

Άπληστοι Αλγόριθμοι

Ο αλγόριθμος του Kruskal

KRUSKAL $(G = (V, E, W))$

1. $T = \emptyset$
2. **while** $(|T| < n - 1)$ **and** $(E \neq \emptyset)$ **do**
3. $\quad e =$smallest edge in E
4. $\quad E = E - \{e\}$
5. $\quad$ **if** $T \cup \{e\}$ has no cycle **then**
6. $\quad\quad T \leftarrow T \cup \{e\}$
7. $\quad$ **end if**
8. **end while**
9. **if** $(|T| < n - 1)$ **then**
10. $\quad$ write "graph disconnected"

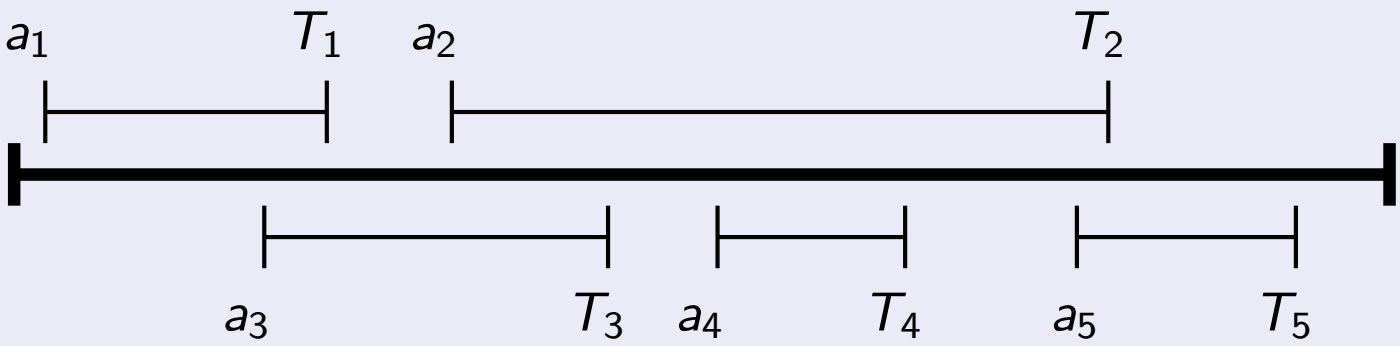
Άπληστοι Αλγόριθμοι

Ανάθεση ενός πόρου

- Δίδεται ένας πόρος (για παράδειγμα ένα αυτοκίνητο προς ενοικίαση ή ένα υπολογιστικό σύστημα προς ανάθεση)
- Ένα σύνολο αιτήσεων $P = \{P_1, \ldots, P_n\}$ χρησιμοποιήσης του πόρου.
- Κάθε αίτηση P_i, αναφέρεται σε ένα χρονικό διάστημα (a_i, T_i) για τη χρησιμοποίηση του πόρου.
- Ο στόχος μας είναι να μεγιστοποιήσουμε το πλήθος των πελατών που θα χρησιμοποιήσουν τον πόρο.

Άπληστοι Αλγόριθμοι

Ένα παράδειγμα του προβλήματος ανάθεσης πόρου (σχ.1)



Άπληστοι Αλγόριθμοι

Ανάθεση ενός πόρου

- Μία βέλτιστη λύση είναι $\{P_3, P_4, P_5\}$.
- Η λύση $\{P_1, P_2\}$ δεν είναι βέλτιστη.

Πρόβλημα: Ανάθεση Πόρου

Βέλτιστη Υποδομή-Θεώρημα

Αν η

$$F_{1,p} =< x_1, x_2, \ldots, x_k, \ldots, x_p >$$

είναι βέλτιστη τότε

$$F_{1,p} = F_{1,k-1} \bigcup \{x_k\} \bigcup F_{k+1,p}$$

και οι $F_{1,k-1}$, $F_{k+1,p}$ είναι βέλτιστες.

Πρόβλημα: Ανάθεση Πόρου

Απόδειξη

Αποκοπή και επικόλληση.
Αν υπήρχε μια λύση $F'_{1,k-1}$, η οποία να περιλαμβάνει περισσότερες αιτήσεις - δραστηριότητες απ' ότι η $F_{1,k-1}$, θα αποκόπταμε την $F_{1,k-1}$ από την $F_{1,p}$ και θα επικολλούσαμε την $F'_{1,k-1}$. Σε αυτή την περίπτωση η $F_{1,p}$ θα είχε περισσότερες αιτήσεις πράγμα που δηλώνει ότι η $F_{1,p}$ δεν είναι βέλτιστη. Αυτό όμως αντιφάσκει με την υπόθεση μας ότι η $F_{1,p}$ είναι βέλτιστη. Το ίδιο σκεπτικό ισχύει και για την $F_{k+1,p}$.

Άπληστοι Αλγόριθμοι

Άπληστος Αλγόριθμος Ανάθεσης Πόρου

ΑΝΑΘΕΣΗ-ΠΟΡΟΥ (P : σύνολο αιτήσεων)

1. ΤΑΞΙΝΟΜΗΣΗ(P)
2. $F = \emptyset$
3. **for** $i = 1$ **to** n
4. Επιλογή P_i
5. **if** $F \cup \{P_i\}$ συμβατή **then**
6. $F = F \cup \{P_i\}$
7. **end if**
8. **end while**
9. return F

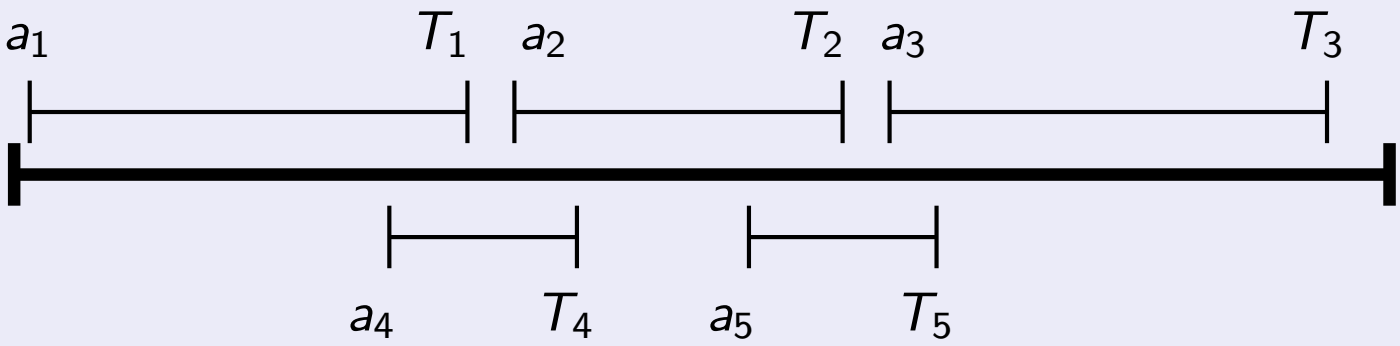
Άπληστοι Αλγόριθμοι

Ανάθεση ενός πόρου

- Η εφικτότητα εδώ σημαίνει ότι όταν εισάγουμε το αίτημα P_i δεν θα πρέπει να επικαλύπτεται με οποιοδήποτε από τα $P_j,\ 1 \leq j < i$.
- Θα πρέπει να ευρεθεί η ταξινόμηση που θα γίνει στα διαστήματα.
- Αύξουσα ταξινόμηση μεγέθους του διαστήματος.
- Στο παράδειγμα του σχήματος η προσέγγιση αυτή θα μας επέστρεφε μία βέλτιστη λύση.
- Δεν δουλεύει γενικά.
- Η βέλτιστη λύση είναι το σύνολο $\{P_1, P_2, P_3\}$, η λύση που επιστρέφει η άπληστη μέθοδος είναι η $\{P_4, P_5\}$.

Άπληστοι Αλγόριθμοι

Ένα παράδειγμα του προβλήματος ανάθεσης πόρου που η επιλογή του μικρότερου δεν επιστρέφει την βέλτιστη λύση (σχ.2).



Άπληστοι Αλγόριθμοι

Ανάθεση ενός πόρου

- Ας θεωρήσουμε τώρα την αύξουσα ταξινόμηση τέλους διαστήματος.
- $[P_1, P_3, P_4, P_2, P_5]$, παράδειγμα του σχήματος 1
- $[P_1, P_4, P_2, P_5, P_3]$, παράδειγμα του σχήματος 2.
- Η εφαρμογή του άπληστου αλγορίθμου δίνει τις αντίστοιχες βέλτιστες λύσεις.

Ορθότητα: Ανάθεση Πόρου

Θεώρημα

Έστω

$$F =< x_1, x_2, \ldots, x_j, \ldots, x_p >$$

η λύση που επιστρέφει ο άπληστος αλγόριθμος. Τότε αυτή η λύση είναι η βέλτιστη.

Ορθότητα: Ανάθεση Πόρου

Απόδειξη ...

Με χρήση της επαγωγής στο j.
Για $j = 1$

$$F =< x_1 >$$

και αυτή είναι η βέλτιστη λύση διότι ο χρόνος περάτωσης της x_1 είναι f_1 και

$$f_1 = \min_{1 \leq i \leq p} \{f_i\}.$$

Συνεπώς η x_1 αφήνει το μέγιστο ελεύθερο χώρο πράγμα που δηλώνει ότι είναι η βέλτιστη επιλογή.

Ορθότητα: Ανάθεση Πόρου

... Απόδειξη

Επαγωγικό βήμα
Υποθέτουμε ότι η

$$F_i =< x_1, x_2, \ldots, x_i >$$

είναι βέλτιστη λύση για όλα τα $i < j$ και θα δείξουμε ότι προσθέτοντας την x_j τότε και η

$$F_j =< x_1, x_2, \ldots, x_i, x_j >$$

είναι βέλτιστη. Πράγματι, αν προσθέτοντας την x_j στην F_i η F_j δεν είναι βέλτιστη, τότε θα είναι κάποια άλλη, έστω $x_j^{'}$, η οποία θα καθιστά την F_j βέλτιστη. Αλλά σε αυτή την περίπτωση

$$f_j < f_j^{'}$$

που δηλώνει ότι η x_j αφήνει λιγότερο ελέυθερο χώρο από την $x_j^{'}$ άρα η F_j δεν είναι βέλτιστη και καταλήξαμε σε αντίφαση.

Άπληστοι Αλγόριθμοι

Ορθότητα: Ανάθεση Πόρου

- Απόδειξη (2ος τρόπος)
  Έστω

$$F = \{x_1, x_2, \ldots, x_k, \ldots, x_p\}$$

$$Opt = \{y_1, y_2, \ldots, y_k, \ldots, y_q\}, q \geq p.$$

- Αν

$$x_1 = y_1, x_2 = y_2, \ldots, x_{k-1} = y_{k-1}, x_k \neq y_k,$$

  τότε

$$Opt' = \{x_1, x_2, \ldots, x_{k-1}, x_k, y_{k+1}, \ldots, y_q\}$$

- Το ίδιο επιχείρημα μπορεί να επαναληφθεί και για τα υπόλοιπα στοιχεία $y_{k+1}, \ldots, y_q$ που σημαίνει ότι $p = q$.
- Η λύση που επιστρέφει ο άπληστος αλγόριθμος είναι βέλτιστη.

Ένα πρόβλημα Χρονοπρογραμματισμού

Το πρόβλημα

- Δίνεται ένας πόρος (π.χ. processor).
- Να εκτελεστούν πολλές εργασίες (π.χ. processes).

Ερώτηση

Ποιά διάταξη θα χρησιμοποιήσουμε για τις εργασίες (jobs);

Υπόθεση

Κάθε εργασία έχει:

- Ένα βάρος w_j (προτεραιότητα).
- Ένα χρόνο εκτέλεσης l_j.

Χρόνος ολοκλήρωσης

Ορισμός

Ο χρόνος ολοκλήρωσης c_j της εργασίας j είναι το άθροισμα των χρόνων εκτέλεσης των εργασιών μέχρι και της j συμπεριλαμβανομένης.

Παράδειγμα

Δίνονται τρεις εργασίες με χρόνους εκτέλεσης $l_1 = 1, l_2 = 2, l_3 = 3$.
Ποιός είναι ο χρόνος ολοκλήρωσης κάθε εργασίας ;

α) 1,2,3

β) 3,5,6

γ) 1,3,6

δ) 1,4,6

Χρόνος ολοκλήρωσης

Επιθυμητό:

Ελάχιστος χρόνος ολοκλήρωσης

Ερώτηση

Ποιός είναι ο χρονοπρογραμματισμός των εργασιών έτσι ώστε ο χρόνος ολοκλήρωσής τους να είναι ελάχιστος;

Απάντηση

Εξαρτάται από την αντικειμενική συνάρτηση.

Αντικειμενική Συνάρτηση

Σκοπός

Ελαχιστοποίηση του σταθμισμένου αθροίσματος του χρόνου ολοκλήρωσης:

$$min \sum_{j=1}^{n} w_j c_j$$

Παράδειγμα

Αν $w_1 = 3$, $w_2 = 2$, $w_3 = 1$, και $l_1 = 1$, $l_2 = 2$, $l_3 = 3$ και η διάταξη των εργασιών είναι η $< 1, 2, 3 >$ τότε $c_1 = 1, c_2 = 3, c_3 = 6$ και

$$\sum_{j=1}^{3} w_j c_j = 3 \cdot 1 + 2 \cdot 3 + 1 \cdot 6 = 15$$

Αυτή είναι η ελάχιστη τιμή του αθροίσματος από όλες τις 3! διατάξεις των εργασιών.

Αντικειμενική Συνάρτηση

Ερώτηση

Δεδομένων των $w_i, c_i, i = 1, 2, ..., n$ ποιά είναι η διάταξη των εργασιών η οποία ελαχιστοποιεί το σταθμισμένο άθροισμά τους;

Σκέψεις για Αλγόριθμο

Σκοπός:Ανάπτυξη ενός άπληστου αλγορίθμου.

Αν υπάρχει ένας τέτοιος αλγόριθμος τότε πώς πρέπει να σκεφτούμε για να τον κατασκευάσουμε;

Σκέψεις για Αλγόριθμο

Ειδικές περιπτώσεις

- Αν οι εργασιές έχουν τον ίδιο χρόνο εκτέλεσης τότε ποιός είναι ο χρονοπρογραμματισμός τους;
- Ξεκινώντας από τις μεγαλύτερου ή μικρότερου βάρους εργασίες;
- Αν έχουν το ίδιο βάρος τότε ποιός είναι ο χρονοπρογραμματισμός τους;
- Ξεκινώντας από τις συντομότερες ή αργότερες εργασίες;

α) μεγαλύτερο\συντομότερες
β) μικρότερο\συντομότερες
γ) μεγαλύτερο\αργότερες
δ) μικρότερο\αργότερες

Αν οι εργασίες έχουν τον ίδιο χρόνο εκτέλεσης τότε η ακολουθία των χρόνων ολοκλήρωσης $c_j, j = 1, 2, ..., n$ θα είναι η

$$1, 2, ..., n$$

και θα είναι πάντα η ίδια. Συνεπώς η ποσότητα

$$\sum_{j=1}^{n} w_j c_j \tag{1}$$

ελαχιστοποιείται αν τα w_j επιλέγουν από το μεγαλύτερο προς το μικρότερο.

Αν τα βάρη $w_j, j = 1, 2, ..., n$ είναι όλα ίσα, τότε η ελαχιστοποίηση της (1) επιτυγχάνεται αν τα $c_j, j = 1, 2, .., n$ επιλεγούν από το μικρότερο προς το μεγαλύτερο. Όσο καθυστερεί μια εργασία τόσο χειρότερο για τις υπόλοιπες.

Παράδειγμα

Δυο εργασίες $w_1 = 1, l_1 = 1, w_2 = 1, l_2 = 2$. Για τον χρονοπρογραμματισμό $< 1, 2 >$ έχουμε

$$c_1 = l_1 = 1, c_2 = l_1 + l_2 = 3$$

άρα

$$1.c_1 + 1.c_2 = 1 + 3 = 4$$

ενώ για τον χρονοπρογραμματισμό $< 2, 1 >$ έχουμε

$$c_2 = l_2 = 2, c_1 = l_2 + l_1 = 3$$

άρα

$$1.c_2 + 1.c_1 = 2 + 3 = 5$$

Ερώτηση

Αν $w_i > w_j$ και $l_i > l_j$;

Ιδέα

Ανάθεση βαθμών στις εργασίες οι οποίες είναι:

- αύξουσες ως προς το βάρος
- φθίνουσες ως προς το χρόνο εκτέλεσης.

Ποιά συνάρτηση θα μπορούσαμε να σκεφτούμε έτσι ώστε να αντιπροσωπεύει το βαθμό ;

Έχουμε πολλές επιλογές:

1. Ταξινόμηση των εργασιών σε φθίνουσα τιμή της $w_j - l_j$.
2. Ταξινόμηση των εργασιών σε φθίνουσα σειρά ως προς το λόγο w_j/l_j.

Μια επιλογή είναι σωστή. Θα απορρίψουμε τη μια με βάση τα παραδείγματα.

Παραδείγματα

1. $l_1 = 5, w_1 = 3$
2. $l_2 = 2, w_2 = 1$

Ερώτηση

Ποίος είναι ο χρόνος ολοκλήρωσης για τις επιλογές 1, 2;

α) 22 και 23

β) 23 και 22

γ) 17 και 17

δ) 17 και 11

Παράδειγμα

1. $l_1 = 5, w_1 = 3$
2. $l_2 = 2, w_2 = 1$

Απάντηση

1. $w_1 - l_1 = 3 - 5 = -2,\ w_2 - l_2 = 1 - 2 = -1$ άρα: $< 2, 1 >$,

$$c_2 w_2 + c_1 w_1 = 2 \cdot 1 + 7 \cdot 3 = 23.$$

2. $w_1/l_1 = 3/5 = 0.6,\ w_2/l_2 = 1/2 = 0.5$ άρα: $< 1, 2 >$,

$$c_1 w_1 + c_2 w_2 = 5 \cdot 3 + 7 \cdot 1 = 22.$$

Συμπεράσματα

- Η επιλογή 1 δεν είναι πάντα σωστή.
- Η επιλογή 2 είναι πάντα σωστή (απόδειξη στη συνέχεια).

Παρατηρήσεις

- Η απόδειξη της ορθότητας ενός άπληστου αλγορίθμου παρουσιάζει δυσκολίες.
- Αντίθετα η πολυπλοκότητα του είναι εύκολο να αναλυθεί.
- Η πολυπλοκότητα του άπληστου αλγορίθμου είναι

$$O(nlogn)$$

Απόδειξη Ορθότητας

Θεώρημα

Η διάταξη των εργασιών σε φθίνουσα σειρά ως προς το λόγο w_j/l_j είναι η βέλτιστη.

Απόδειξη

Θα χρησιμοποιήσουμε τη μέθοδο της Ανταλλαγής Επιχειρημάτων.
Υπόθεση : Κάθε εργασία έχει διαφορετικούς λόγους.

Απόδειξη Ορθότητας

Απόδειξη

- Έστω n εργασίες. Θα εφαρμόσουμε την εις άτοπο απαγωγή.
- Θεωρούμε σ ο άπληστος χρονοπρογραμματισμός και σ^* ο βέλτιστος χρονοπρογραμματισμός (καλύτερος από τον σ).
- Θα αποδείξουμε ότι υπάρχει χρονοπρογραμματισμός καλύτερος από τον σ^* επομένως αυτός δεν είναι ο βέλτιστος.
- Υποθέτουμε ότι όλοι οι λόγοι είναι διαφορετικοί και ότι ισχύει:

$$\frac{w_1}{l_1} > \frac{w_2}{l_2} > ... > \frac{w_n}{l_n}$$

.

- Τότε, ο άπληστος χρονοπρογραμματισμός θα είναι ο

$$< 1, 2, 3, ..., n >$$

.

Απόδειξη Ορθότητας

- Αν ο βέλτιστος χρονοπρογραμματισμός $\sigma^* \neq \sigma$, τότε υπάρχουν συνεχόμενες εργασίες i, j με $i > j$ στη σ^* δρομολόγηση, ώστε η i να εκτελείται νωρίτερα.
- Αυτό ισχύει διότι η μόνη περίπτωση που οι δείκτες αυξάνονται είναι αυτή του χρονοπρογραμματισμού

$$< 1, 2, 3, ..., n >,$$

οποιοσδήποτε άλλος χρονοπρογραμματισμός θα έχει δυο συνεχόμενες εργασίες i, j όπου η μια εκτελείται μετά την άλλη και $i > j$.

Τεχνική ανταλλαγής επιχειρημάτων

- Έστω ότι ανταλλάσεται η διάταξη των i, j στη σ^* (αφήνοντας τις υπόλοιπες εργασίες ως έχουν).

Ανάλυση κόστους

Ερώτηση

Ποιά είναι η επίδραση αυτής της ανταλλαγής στο χρόνο ολοκλήρωσης;

1. μιας εργασίας k διαφορετικής από τις i, j
2. της εργασίας i
3. της εργασίας j

Απάντηση

α) όχι αρκετή πληροφόρηση/αυξάνεται/μειώνεται

β) όχι αρκετή πληροφόρηση/μειώνεται/αυξάνεται

γ) δεν επηρρεάζεται/αυξάνεται/μειώνεται

δ) δεν επηρρεάζεται/μειώνεται/αυξάνεται

Απάντηση

Η σωστή απάντηση είναι η γ).
Οι εργασίες πριν και μετά τις i και j δεν επηρρεάζονται αφού ο χρόνος ολοκλήρωσής τους δε μεταβάλλεται.

Συμπέρασμα

1. Το κόστος της ανταλλαγής είναι

$$Cost = w_i l_j$$

   διότι το c_i αυξάνεται κατα l_j.

2. Το όφελος της ανταλλαγής είναι

$$Benefit = w_j l_i$$

   διότι το c_j μειώνεται κατά l_i.

Συμπέρασμα

- Αλλά

$$Cost < Benefit$$

Πράγματι,

$$w_i l_j < w_j l_i$$

ή

$$\frac{w_i}{l_i} < \frac{w_j}{l_j}$$

το οποίο ισχύει λόγω της αρχικής μας υπόθεσης αφού $i > j$.

- Συνεπώς η ανταλλαγή βελτιώνει τον χρονοπρογραμματισμό σ^* πράγμα που έρχεται σε αντίφαση με την υπόθεσή μας ότι ο σ^* είναι βέλτιστος.

Γενίκευση του Αλγορίθμου

Γενίκευση του Αλγορίθμου (ίσοι λόγοι)

Ο άπληστος αλγόριθμος (διάταξη των εργασιών κατα μη αύξουσα σειρά των λόγων w_j/l_j) είναι βέλτιστος.

Απόδειξη

Θεωρούμε σ^* οποιοδήποτε άλλο χρονοπρογραμματισμό. Θα δείξουμε ότι ο σ είναι τουλάχιστον τόσο καλός όσο και ο σ^*. Άρα ο άπληστος είναι βέλτιστος αφού είναι καλύτερος από οποιοδήποτε άλλον άρα και από τον βέλτιστο.

Απόδειξη

Υπόθεση

Ο άπληστος σ είναι ο

$$< 1, 2, 3, ..., n >$$

με

$$\frac{w_1}{l_1} \geq \frac{w_2}{l_2} \geq ... \geq \frac{w_n}{l_n}$$

θεωρούμε έναν αυθαίρετο χρονοπρογραμματισμό σ^* και θα δείξουμε ότι δεν είναι καλύτερος από τον σ .

- Αν $\sigma^* = \sigma$ τελειώσαμε.
- Διαφορετικά, υπάρχουν διαδοχικές εργασίες i, j στον σ^* με $i > j$.
- Ανταλλάσοντας τις i, j στη σ^* έχει καθαρό όφελος $w_j l_i - w_i l_j \geq 0$ ή

$$Cost \leq Benefit.$$

Απόδειξη

- Αν $i = j$ το κόστος είναι ίσο με το όφελος.
- Ανταλλάσοντας τις i και j (γειτονική αντιστροφή) ο σ^* δε γίνεται χειρότερος, γίνεται μόνο καλύτερος και μειώνεται το πλήθος των αντίστροφων ζευγαριών.
- Μετά από το πολύ $\binom{n}{2}$ ανταλλαγές ο σ^* τροποποιείται στον σ.
- ο σ τουλάχιστον τόσο καλός όσο ο σ^*
- Άρα ο άπληστος χρονοπρογραμματισμός είναι βέλτιστος.

Άπληστοι Αλγόριθμοι

Αποθήκευση αρχείων σε δίσκους

- Δίδονται n αρχεία με χωρητικότητες $l_1, l_2, \ldots, l_n$ και δύο δίσκοι ίσης χωρητικότητας L.
- Θέλουμε να αποθηκεύσουμε όσο το δυνατό περισσότερα αρχεία στους δίσκους, θεωρώντας ότι κάθε αρχείο πρέπει να αποθηκευτεί σε έναν μόνο δίσκο και

$$\sum_{i=1}^{n} l_i \leq 2L$$

.

Άπληστοι Αλγόριθμοι

Άπληστος αλγόριθμος αποθήκευσης αρχείων σε δύο ίσης χωρητικότητας δίσκους

ΑΠΟΘΗΚΕΥΣΗ-ΑΡΧΕΙΩΝ (I : σύνολο αρχείων, L : χωρητικότητα δίσκων)
/* αύξουσα σειρά μεγεθών */

1. ΑΥΞΟΥΣΑ-ΤΑΞΙΝΟΜΗΣΗ(I)
2. $i = 1$
3. **for** $j = 1$ **to** 2 /* δίσκος j */
4. $sum = 0$
5. **while** $sum + l_i \leq L$
6. καταχώρηση $i \rightarrow j$
7. $sum = sum + l_i$
8. $i = i + 1$
9. **end while**
10. **end for**

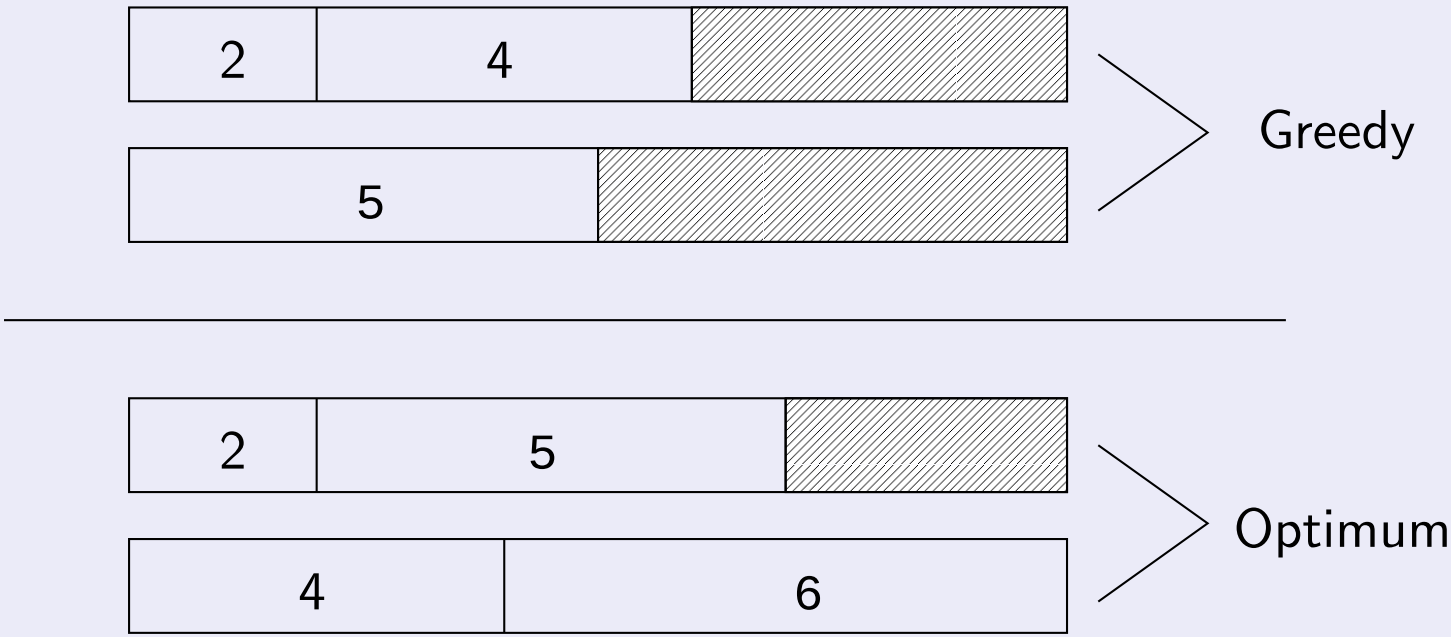
Άπληστοι Αλγόριθμοι

Αποθήκευση αρχείων σε δίσκους

- Η μέθοδος αυτή δεν δίνει τη βέλτιστη λύση.
- Οι δίσκοι έχουν χωρητικότητα 10 και υπάρχουν 4 αρχεία με αντίστοιχα μεγέθη $2, 4, 5, 6$.
- Το πρόβλημα είναι NP-δύσκολο.

Άπληστοι Αλγόριθμοι

Παράδειγμα όπου ο άπληστος αλγόριθμος δεν επιστρέφει τη βέλτιστη λύση. (Πάνω είναι η λύση του άπληστου αλγορίθμου με 3 αρχεία και κάτω η βέλτιστη λύση με 4 αρχεία.)



Άπληστοι Αλγόριθμοι

Το Διακριτό Πρόβλημα Σακιδίου (Discrete Knapsack) .

- Ένας αλπινιστής προετοιμάζει το σακίδιό του για την ανάβαση στο βουνό.
- Έχει ένα σύνολο από n αντικείμενα $X = \{x_1, x_2, \ldots, x_n\}$ με βάρη $a_1, a_2, \ldots, a_n$.
- Κάθε αντικείμενο δίνει ένα κέρδος στον αλπινιστή c_i.

Άπληστοι Αλγόριθμοι

Το Διακριτό Πρόβλημα Σακιδίου (Discrete Knapsack) .

- Αν ο αλπινιστής έχει επιλέξει το αντικείμενο i, ο στόχος είναι η μεγιστοποίηση του αθροίσματος:

$$\max \sum_{i=1}^{n} c_i z_i$$

με τον περιορισμό (b=βάρος σακιδίου):

$$\sum_{i=1}^{n} a_i z_i \leq b$$

- Για την επίλυση του προβλήματος με άπληστο αλγόριθμο, η ταξινόμηση των αντικειμένων σύμφωνα με τον όρο c_i/a_i.
- Η προσέγγιση αυτή δεν δουλεύει πάντα.

Άπληστοι Αλγόριθμοι

Άπληστος αλγόριθμος για το διακριτό πρόβλημα σακιδίου

DISCRETE-KNAPSACK (X : σύνολο αντικειμένων, a, c : αντίστοιχα βάρη και αξίες, b : βάρος σακιδίου)

1. ΦΘΙΝΟΥΣΑ-ΤΑΞΙΝΟΜΗΣΗ(c_i/a_i)
2. $S = \emptyset$
3. **for** $i = 1$ **to** n
4. **if** $b \geq a_i$
5. $S = S \cup \{x_i\}$
6. $b = b - a_i$
7. **end if**
8. **end for**

Το Διακριτό Πρόβλημα Σακιδίου (Discrete Knapsack) .

- Δίδονται 3 αντικείμενα $\{x_1, x_2, x_3\}$ με βάρη $a = (k+1, k, k)$ και αξίες $c = (k+2, k, k)$.
- Το βάρος του σακιδίου είναι $2k$.
- Η ταξινόμηση των αντικειμένων σε φθίνουσα σειρά του λόγου c_i/a_i είναι x_1, x_2, x_3 αφού
$$\frac{k+2}{k+1} > \frac{k}{k} = \frac{k}{k}$$
- Ο αλγόριθμος θα επιλέξει το αντικείμενο x_1 με κέρδος $k+2$ αντί να επιλέξει τα αντικείμενα x_2, x_3 με κέρδος $2k$.
- Το πρόβλημα του σακιδίου είναι και αυτό $NP-$δύσκολο.

Άπληστοι Αλγόριθμοι

Το Συνεχές Πρόβλημα Σακιδίου (Continuous Knapsack) .

- Επιτρέπεται στον αλπινιστή να πάρει μέρη των αντικειμένων.
- Οι μεταβλητές απόφασης z_i του ακέραιου προβλήματος παίρνουν συνεχείς τιμές στο $[0, 1]$.
- Εδώ η φθίνουσα ταξινόμηση με βάση τον λόγο c_i/a_i δουλεύει αποδοτικά και βρίσκει την βέλτιστη λύση.

Άπληστοι Αλγόριθμοι

Άπληστος αλγόριθμος για το συνεχές πρόβλημα σακιδίου

CONTINUOUS-KNAPSACK (X : σύνολο αντικειμένων, a, c : αντίστοιχα βάρη και αξίες, b : βάρος σακιδίου)

1. ΦΘΙΝΟΥΣΑ-ΤΑΞΙΝΟΜΗΣΗ(c_i/a_i)
2. **for** $i = 1$ **to** n Θέσε $z_i = 0$
3. $i = 1$
4. **while** $b \geq a_i$ **do**
5. $z_i = 1$
6. $b = b - a_i$
7. $i = i + 1$
8. **end while**
9. $z_i = b * (c_i/a_i)$

Huffman

Στοιχεία της Άπληστης Στρατηγικής

- Ένας άπληστος αλγόριθμος προσδιορίζει μια βέλτιστη λύση σε κάποιο πρόβλημα μέσω κάποιων διαδοχικών επιλογών.
- Σε κάθε «σημείο απόφασης» του αλγορίθμου, η επιλογή που προτείνεται είναι εκείνη που φαίνεται καλύτερη τη δεδομένη στιγμή.
- Η ευρύτερη αυτή στρατηγική δεν αποδίδει πάντοτε μια βέλτιστη λύση, αλλά σε ορισμένες περιπτώσεις λειτουργεί με επιτυχία.

Huffman

Σχεδιασμός Άπληστων Αλγορίθμων

Γενικότερα, για να σχεδιάσουμε άπληστους αλγορίθμους ακολουθούμε την παρακάτω ακολουθία βημάτων:

- Καταστρώνουμε το πρόβλημα βελτιστοποίησης με τέτοιο τρόπο ώστε όταν υιοθετούμε μια επιλογή να απομένει προς επίλυση μόνο ένα υποπρόβλημα.
- Αποδεικνύουμε ότι υπάρχει πάντοτε μια βέλτιστη λύση του αρχικού προβλήματος η οποία χρησιμοποιεί την άπληστη επιλογή, και επομένως η επιλογή αυτή μπορεί να εξασφαλίσει πάντοτε μια βέλτιστη λύση.
- Δείχνουμε ότι, υιοθετώντας την άπληστη επιλογή, αυτό που απομένει είναι ένα υποπρόβλημα που έχει την ιδιότητα ότι ο συνδυασμός μιας βέλτιστης λύσης του με την άπληστη επιλογή μας δίνει μια βέλτιστη λύση του αρχικού προβλήματος.

Huffman

Σχεδιασμός Άπληστων Αλγορίθμων (συν.)

Γενικότερα, για να σχεδιάσουμε άπληστους αλγορίθμους ακολουθούμε την παρακάτω ακολουθία βημάτων:

- Καταστρώνουμε το πρόβλημα βελτιστοποίησης με τέτοιο τρόπο ώστε όταν υιοθετούμε μια επιλογή να απομένει προς επίλυση μόνο ένα υποπρόβλημα.
- Αποδεικνύουμε ότι υπάρχει πάντοτε μια βέλτιστη λύση του αρχικού προβλήματος η οποία χρησιμοποιεί την άπληστη επιλογή, και επομένως η επιλογή αυτή μπορεί να εξασφαλίσει πάντοτε μια βέλτιστη λύση.
- Δείχνουμε ότι, υιοθετώντας την άπληστη επιλογή, αυτό που απομένει είναι ένα υποπρόβλημα που έχει την ιδιότητα ότι ο συνδυασμός μιας βέλτιστης λύσης του με την άπληστη επιλογή μας δίνει μια βέλτιστη λύση του αρχικού προβλήματος.

Σχεδιασμός Άπληστων Αλγορίθμων (συν.)

- Πώς μπορεί κανείς να αποφανθεί κατά πόσο ένα συγκεκριμένο πρόβλημα βελτιστοποίησης επιλύεται μέσω μιας άπληστης προσέγγισης;
- Αν και εν γένει δεν υπάρχει απάντηση στο ερώτημα αυτό, δυο βασικά χαρακτηριστικά προβλημάτων που επιδέχονται τέτοια αντιμετώπιση είναι
  - η ιδιότητα της άπληστης επιλογής και
  - η βέλτιστη υποδομή.
- Αν μπορούμε να αποδείξουμε ότι κάποιο δεδομένο πρόβλημα έχει αυτές τις δύο ιδιότητες, τότε έχουμε κάνει ένα αποφασιστικό βήμα προς την ανάπτυξη ενός άπληστου αλγορίθμου για το πρόβλημα αυτό.

Huffman

Ιδιότητα της Άπληστης Επιλογής

- Το πρώτο βασικό συστατικό είναι η ***ιδιότητα της άπληστης επιλογής:*** μια τοπικά βέλτιστη (άπληστη) επιλογή είναι δυνατόν να οδηγήσει σε μια καθολικά βέλτιστη λύση.
- Σε κάθε περίπτωση που έχουμε να επιλέξουμε μεταξύ εναλλακτικών δυνατοτήτων, κάνουμε την επιλογή που φαίνεται καλύτερη τη δεδομένη στιγμή για το τρέχον πρόβλημα, αγνοώντας τα αποτελέσματα των διαφόρων υποπροβλημάτων.
- Στον δυναμικό προγραμματσμό, κάνουμε μια επιλογή σε κάθε βήμα, που συνήθως εξαρτάται από τις λύσεις υποπροβλημάτων.
- Σε προβλήματα δυναμικού προγραμματισμού εργαζόμαστε κατά κανόνα αναβιβαστικά, οδεύοντας από μικρότερα προς μεγαλύτερα υποπροβλήματα.

Huffman

Ιδιότητα της Άπληστης Επιλογής (συν.)

- Στην άπληστη προσέγγιση, υιοθετούμε την επιλογή που φαίνεται καλύτερη τη δεδομένη στιγμή και εν συνεχεία επιλύουμε το υποπρόβλημα που προκύπτει με βάση αυτήν την επιλογή. Επομένως, αντίθετα προς το δυναμικό προγραμματισμό, όπου τα προβλήματα επιλύονται αναβιβαστικά, στην άπληστη προσέγγιση η πορεία της λύσης είναι συνήθως καταβιβαστική: το κάθε στιγμιότυπο προβλήματος ανάγεται σε ένα μικρότερο πρόβλημα, με διαδοχικές άπληστες επιλογές.
- Θα πρέπει να αποδείξουμε ότι μια άπληστη επιλογή σε κάθε βήμα δίνει μια καθολικά βέλτιστη λύση.
- Για να αποδείξουμε την ιδιότητα αυτή εξετάζουμε μια καθολικά βέλτιστη λύση σε κάποιο υποπρόβλημα. Εν συνεχεία, αποδεικνύουμε ότι η λύση μπορεί να τροποποιηθεί ώστε να περιλαμβάνει την άπληστη επιλογή, οπότε προκύπτει ένα παρόμοιο αλλά μικρότερο υποπρόβλημα.

Huffman

Βέλτιστη Υποδομή

- Η ιδιότητα της ***βέλτιστης υποδομής*** ενός προβλήματος αποτελεί ένα βασικό κριτήριο της εφαρμοσιμότητας τόσο του δυναμικού προγραμματισμού όσο και της άπληστης προσέγγισης.
- Ένα πρόβλημα διαθέτει βέλτιστη υποδομή αν μια βέλτιστη λύση του εμπεριέχει βέλτιστες λύσεις υποπροβλημάτων.
- Υιοθετώντας την άπληστη επιλογή στο αρχικό πρόβλημα καταλήγουμε σε ένα υποπρόβλημα. Πρέπει να δείξουμε ότι ο συνδυασμός μιας βέλτιστης λύσης του υποπροβλήματος με την άπληστη επιλογή που έχουμε ήδη κάνει δίνει μια βέλτιστη λύση του αρχικού προβλήματος.
- Ο μηχανισμός αυτός βασίζεται σιωπηρά στην εφαρμογή της επαγωγικής μεθόδου στα υποπροβλήματα μέσω της οποίας αποδεικνύουμε ότι η υιοθέτηση της άπληστης επιλογής σε κάθε βήμα δίνει μια βέλτιστη λύση.

Huffman

Άπληστη Στρατηγική έναντι Δυναμικού Προγραμματισμού

- Δεδομένου ότι η ιδιότητα της βέλτιστης υποδομής αποτελεί τη βάση τόσο της άπληστης όσο και της δυναμικής στρατηγικής, είναι πιθανό να παρασυρθεί κανείς να καταστρώσει μια δυναμική λύση σε περιπτώσεις όπου θα αρκούσε μια άπληστη, ή να θεωρήσει εσφαλμένα ότι μπορεί να εφαρμοστεί μια άπληστη λύση σε περιπτώσεις όπου απαιτείται δυναμική λύση.
- Για να αποσαφηνίσουμε τις λεπτές διαφορές μεταξύ των δύο τεχνικών, θα μελετήσουμε δύο παραλλαγές ενός κλασικού προβλήματος βελτιστοποίησης.

Huffman

Άπληστη Στρατηγική έναντι Δυναμικού Προγραμματισμού

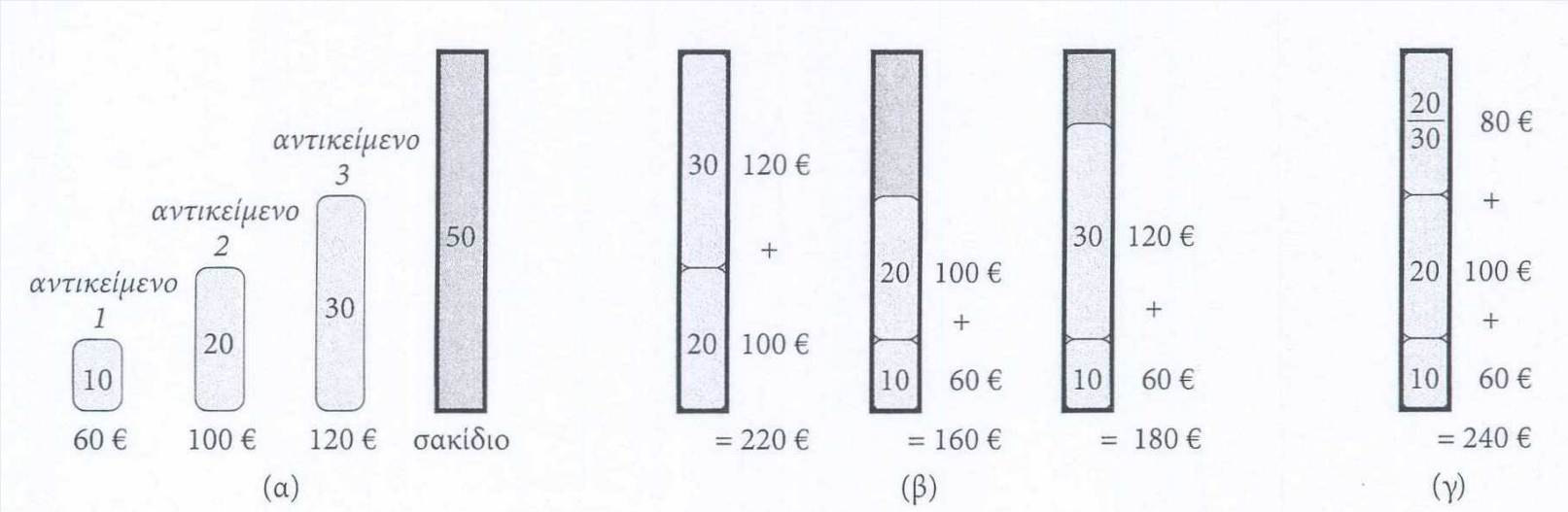
- Το ***ακέραιο πρόβλημα του σακιδίου*** μπορεί να διατυπωθεί ως εξής.
  - Ένας διαρρήκτης παραβιάζει ένα κατάστημα και βρίσκει n αντικείμενα· το $i-$οστό αντικείμενο έχει αξία v_i ευρώ και βάρος w_i κιλά, όπου τα v_i και w_i είναι ακέραιοι αριθμοί. Ο διαρρήκτης θέλει να αποκομίσει όσο το δυνατόν πιο πολύτιμη λεία, αλλά μπορεί να μεταφέρει στο σακίδιό του φορτίο βάρους το πολύ ίσο με W κιλά, όπου W ακέραιος αριθμός. Ποιά αντικείμενα θα πρέπει να πάρει;
- Στο ***κλασικό πρόβλημα του σακιδίου***, έχουμε το ίδιο σκηνικό μόνο που ο διαρρήκτης δεν είναι υποχρεωμένος να επιλέξει «διαζευτικά» για το κάθε αντικείμενο αλλά μπορεί να πάρει και κλάσματα αντικειμένων.
- Μπορούμε να θεωρήσουμε ότι στο ακέραιο πρόβλημα του σακιδίου το κάθε αντικείμενο είναι κάτι σαν χρυσό κόσμημα, ενώ στο κλασικό πρόβλημα το κάθε αντικείμενο είναι κάτι σαν σακουλάκι με χρυσόσκονη.
- Και οι δυο εκδοχές του προβλήματος του σακιδίου διαθέτουν την ιδιότητα της βέλτιστης υποδομής.

Huffman

Άπληστη Στρατηγική έναντι Δυναμικού Προγραμματισμού (συν.)

- Αν και τα δύο προβλήματα είναι παρόμοια, το κλασματικό πρόβλημα μπορεί να επιλυθεί με την άπληστη στρατηγική, ενώ το ακέραιο όχι.
- Για να επιλύσουμε το κλασματικό πρόβλημα, αρχικά υπολογίζουμε την αξία ανά μονάδα βάρους (δηλ. ανά «κιλό») v_i/w_i για κάθε αντικείμενο. Ακολουθώντας την άπληστη στρατηγική, ο διαρρήκτης αρχικά παίρνει τη μέγιστη δυνατή ποσότητα από το αντικείμενο με τη μέγιστη αξία ανά μονάδα βάρους.
- Με την ταξινόμηση των αντικειμένων κατά φθίνουσα σειρά ως προς την αξία ανά μονάδα βάρους, ο άπληστος αλγόριθμος εκτελείται σε χρόνο $O(n \log n)$
- Το ότι αυτή η άπληστη στρατηγική δεν δίνει σωστά αποτελέσματα για το ακέραιο πρόβλημα του σακιδίου μπορεί να αποδειχθεί μέσω του στιγμιοτύπου του προβλήματος που απεικονίζεται γραφικά στο παρακάτω σχήμα.

Άπληστη Στρατηγική έναντι Δυναμικού Προγραμματισμού (συν.)



- Η άπληστη στρατηγική δεν δίνει σωστά αποτελέσματα για το ακέραιο πρόβλημα του σακιδίου.
- Οποιαδήποτε λύση που περιλαμβάνει το αντικείμενο 1 είναι χειρότερη της βέλτιστης, παρ΄ όλο που το αντικείμενο αυτό έχει τη μέγιστη αξία ανά μονάδα βάρους.
- Για το κλασματικό πρόβλημα του σακιδίου, η επιλογή των αντικειμένων κατά φθίνουσα σειρά αξίας ανά μονάδα βάρους δίνει μια βέλτιστη λύση.

Huffman

Άπληστη Στρατηγική έναντι Δυναμικού Προγραμματισμού (συν.)

- Επομένως, στο ακέραιο πρόβλημα, για να αποφασίσουμε αν θα συμπεριλάβουμε το κάθε αντικείμενο στο σακίδιο, θα πρέπει να συγκρίνουμε τη λύση για το υποπρόβλημα στο οποίο το αντικείμενο αυτό συμπεριλαμβάνεται με τη λύση για το υποπρόβλημα στο οποίο το αντικείμενο αποκλείεται.
- Όταν το πρόβλημα διατυπώνεται με αυτόν το τρόπο ανακύπτουν πολλά επικαλυπτόμενα υποπροβλήματα-χαρακτηριστικό γνώρισμα των προβλημάτων δυναμικού προγραμματισμού.
- Πράγματι, το ακέραιο πρόβλημα του σακιδίου μπορεί να λυθεί μέσω δυναμικού προγραμματισμού όπως θα δούμε στη συνέχεια.

Huffman

Κώδικες Huffman

- Οι κώδικες Huffman αποτελούν μια ευρέως διαδεδομένη και πολύ αποτελεσματική τεχνική συμπίεσης δεδομένων.
- Στην προκειμένη περίπτωση, θα θεωρήσουμε ότι τα δεδομένα είναι μια ακολουθία αλφαρηθμητικών χαρακτήρων.
- Ο άπληστος αλγόριθμος του Huffman προσδιορίζει έναν βέλτιστο τρόπο αναπαράστασης του κάθε χαρακτήρα υπό τη μορφή δυαδικής συμβολοσειράς, με βάση τις συχνότητες εμφάνισης των χαρακτήρων.

	α	β	γ	δ	ε	ζ
Συχνότητα (σε χιλιάδες)	45	13	12	16	9	5
Κωδικός σταθερού μήκους	000	001	010	011	100	101
Κωδικός μεταβλητού μήκους	0	101	100	111	1101	1100

Huffman

Απροθηματικοί Κώδικες

- Στην ανάλυση που ακολουθεί, θα περιοριστούμε σε κώδικες στους οποίους κανένας κωδικός δεν αποτελεί ταυτόχρονα πρόθημα κάποιου άλλου κωδικού.
- Αυτού του τύπου οι κώδικες ονομάζονται ***απροθηματικοί***.
- Η κωδικοποίηση κάποιου αρχείου με βάση έναν δυαδικό αλφαρηθμητικό κώδικα είναι απλούστατη· απλώς συναρμόζουμε τους κωδικούς που αναπαριστούν τον κάθε χαρακτήρα του αρχείου.

Huffman

Απροθηματικοί Κώδικες

- Τα πλεονεκτήματα των απροθηματικών κωδικών είναι η εύκολη αποκωδικοποίησή τους.
- Δεδομένου ότι κανένας κωδικός δεν συμπίπτει με το πρόθημα κάποιου άλλου, ο εναρκτήριος κωδικός σε ένα κωδικοποιημένο αρχείο είναι μονοσήμαντα ορισμένος.
- Επομένως, μπορούμε απλώς να προσδιορίσουμε τον εναρκτήριο κωδικό, να τον αποκωδικοποιήσουμε, και να επαναλάβουμε τη διαδικασίατης αποκωδικοποίησης για το εναπομένον τμήμα του κωδικοποιημένου αρχείου.
- Στον κώδικα του πίνακα, η συμβολοσειρά 001011101 αναλύεται μονοσήματα στην αλληλουχία $0 \cdot 0 \cdot 101 \cdot 1101$, η αποκωδικοποίηση της οποίας δίνει αabe.

Huffman

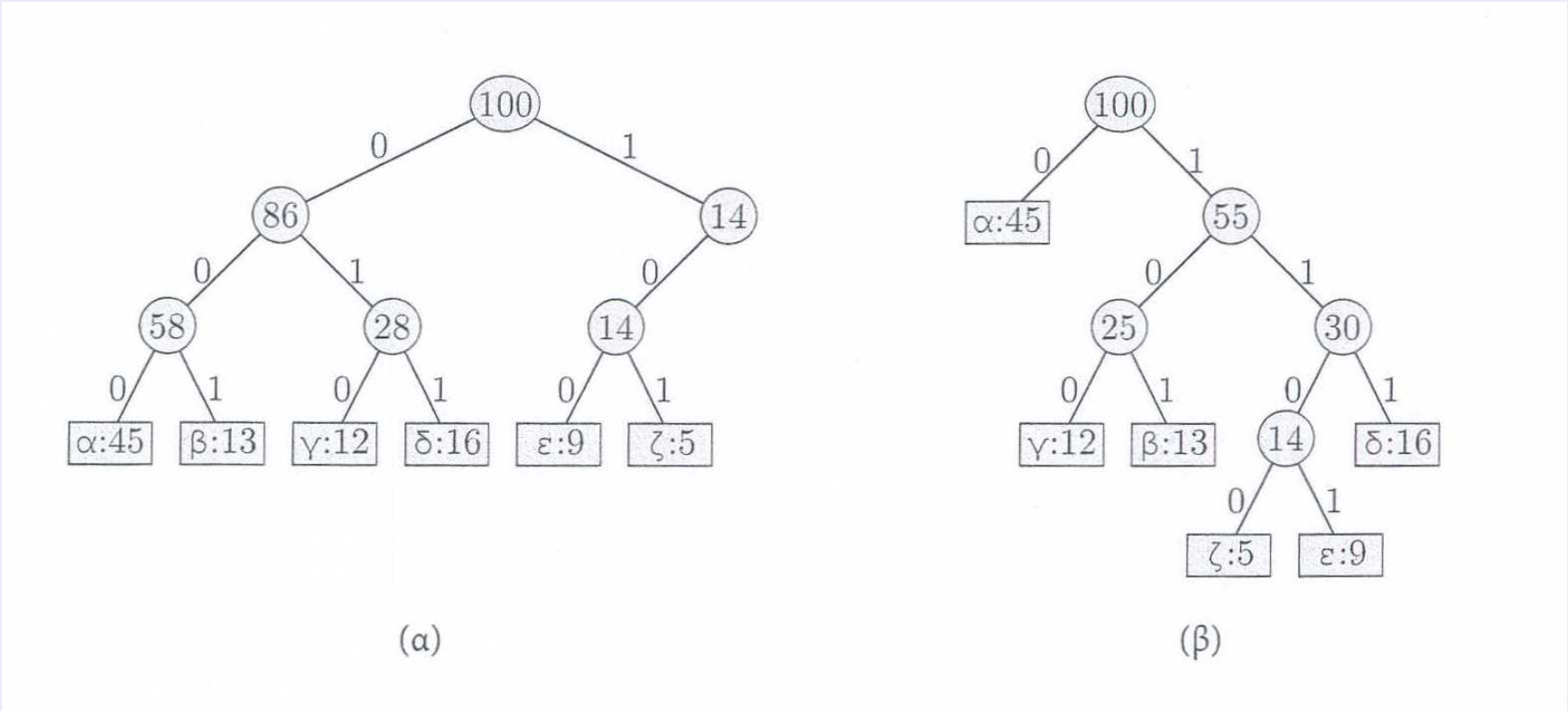
Απροθηματικοί Κώδικες

- Για την διαδικασία της αποκωδικοποίησης μας χρειάζεται μια εύχρηστη αναπαράσταση του απροθηματικού κώδικα η οποία να μας δίνει τη δυνατότητα να προσδιορίζουμε εύκολα τον εναρκτήριο κωδικό.
- Μια τέτοια αναπαράσταση επιτυγχάνεται μέσω ενός δυαδικού δέντρου του οποίου οι καταληκτικοί κόμβοι αντιστοιχούν στους δεδομένους χαρακτήρες.
- Σε ένα τέτοιο δέντρο, ο δυαδικός κωδικός ενός χαρακτήρα συνίσταται στη διαδρομή από τη ρίζα του δέντρου μέχρι τον συγκεκριμένο χαρακτήρα, όπου το 0 σημαίνει «μετάβαση στον αριστερό θυγατρικό κόμβο» και το 1 «μετάβαση στον δεξιό θυγατρικό κόμβο»

Huffman

Απροθηματικοί Κώδικες (συν.)

- Στο παρακάτω σχήμα φαίνονται τα δέντρα που αναπαριστούν τους δύο ενδεικτικούς κώδικες που έχουμε ήδη αναφέρει.



Huffman

Απροθηματικοί Κώδικες (συν.)

- Ένας βέλτιστος κώδικας αναπαρίσταται πάντοτε από ένα πλήρες δυαδικό δέντρο, στο οποίο κάθε κόμβος (εξαιρουμένων, βέβαια, των «καταληκτικών») έχει δύο θυγατρικούς.
- Ο κώδικας σταθερού μήκους του παραδείγματός μας δεν είναι βέλτιστος, αφού το δέντρο του, το οποίο απεικονιζόταν στο παραπάνω σχήμα, δεν είναι ένα πλήρες δυαδικό δένδρο.
- Δεδομένου ότι μπορούμε πλέον να περιορίσουμε την ανάλυσή μας σε πλήρη δυαδικά δένδρα, μπορούμε να πούμε ότι αν οι χαρακτήρες του κειμένου μας προέρχονται από κάποιο αλφάβητο C και οι συχνότητες τους είναι όλες θετικές, τότε το δένδρο που αναπαριστά έναν βέλτιστο απροθηματικό κώδικα έχει ακριβώς $|C|$ καταληκτικούς κόμβους, έναν για κάθε γράμμα του αλφαβήτου, και ακριβώς $|C| - 1$ εσωτερικούς κόμβους.

Απροθηματικοί Κώδικες (συν.)

- Αν γνωρίζουμε το δένδρο T που αναπαριστά έναν απροθηματικό κώδικα, μπορούμε να υπολογίσουμε εύκολα το πλήθος των δυφίων που απαιτούνται για την κωδικοποίηση ενός αρχείου.
- Για κάθε χαρακτήρα c του αλφαβήτου C, έστω $f(c)$ η συχνότητα εμφάνισής του στο αρχείο και $d_T(c)$ το βάθος του καταληκτικού κόμβου στο δέντρο. Σημειώστε ότι η ποσότητα $d_T(c)$ ισούται επίσης με το μήκος του κωδικού του χαρακτήρα c.
- Επομένως το πλήθος των δυφίων που απαιτούνται για την κωδικοποίηση του αρχείου είναι

$$B(T) = \sum_{c \in C} f(c) d_T(c)$$

Huffman

Κατασκευή ενός Κώδικα Huffman

- Ένας καθιερωμένος άπληστος αλγόριθμος για την κατασκευή ενός βέλτιστου απροθηματικού κώδικα είναι αυτός που ανέπτυξε ο Huffman (κώδικας Huffman).
- Η ορθότητα του αλγορίθμου βασίζεται στην ιδιότητα της άπληστης επιλογής και στη βέλτιστη υποδομή.
- Θα παρουσιάσουμε πρώτα τον ψευδοκώδικα για να αποσαφηνιστεί καλύτερα ο μηχανισμός με τον οποίο ο αλγόριθμος κάνει άπληστες επιλογές.

Huffman

Κατασκευή ενός Κώδικα Huffman (συν.)

- Υποθέτουμε ότι το C είναι ένα σύνολο από n χαρακτήρες και ότι ο κάθε χαρακτήρας $c \in C$ είναι ένα αντικείμενο με καθορισμένη συχνότητα $f[c]$.
- Ο αλγόριθμος κατασκευάζει το δέντρο T το οποίο αντιστοιχεί στον βέλτιστο κώδικα ακολουθώντας μια αναβιβαστική στατηγική.
- Ξεκινά με ένα σύνολο από $|C|$ καταληκτικούς κόμβους και δημιουργεί το τελικό δένδρο εκτελώντας μια ακολουθία $|C| - 1$ πράξεων «συγχώνευσης».
- Ο αλγόριθμος προσδιορίζει τα δύο αντικείμενα με τις μικρότερες συχνότητες εμφάνισης τα οποία θα συγχωνευθούν μέσω μιας ουράς προτεραιότητας ελαχίστου Q με κλειδιά τις συχνότητες f.
- Το προϊόν της συγχώνευσης των δυο αντικειμένων είναι ένα νέο αντικείμενο με συχνότητα το άθροισμα των δύο συγχωνευθέντων αντικειμένων.

Huffman

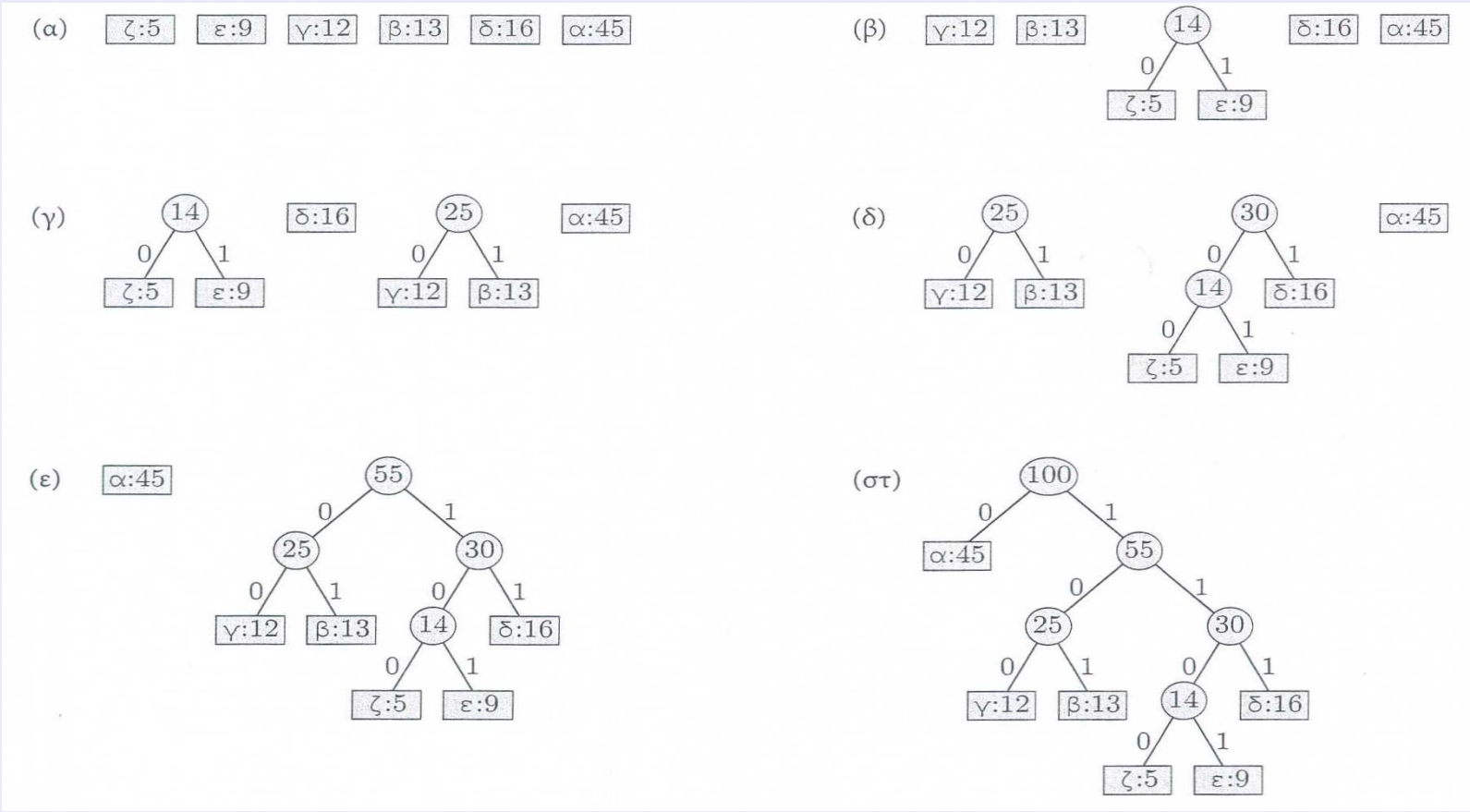
Κατασκευή ενός Κώδικα Huffman (συν.)

HUFFMAN(C)

1. $n \leftarrow |C|$
2. $Q \leftarrow |C|$
3. για $i \leftarrow 1$ έως $n-1$
4. δεσμεύουμε ένα νέο κόμβο z
5. αριστερός$[z] \leftarrow x \leftarrow$ ΕΞΑΓΩΓΗ ΕΛΑΧΙΣΤΟΥ(Q)
6. δεξιός$[z] \leftarrow y \leftarrow$ ΕΞΑΓΩΓΗ ΕΛΑΧΙΣΤΟΥ(Q)
7. $f[z] \leftarrow f[x] + f[y]$
8. ΕΙΣΑΓΩΓΗ(Q, z)
9. επιστροφή ΕΞΑΓΩΓΗ ΕΛΑΧΙΣΤΟΥ(Q) (επιστρέφει τη ρίζα του δέντρου)

Huffman

Κατασκευή ενός Κώδικα Huffman (συν.)



Ο συνολικός χρόνος εκτέλεσης της διαδικασίας Huffman για ένα σύνολο n χαρακτήρων είναι $O(n \log n)$

Huffman

Ορθότητα Αλγορίθμου Huffman

Για να αποδείξουμε ότι ο άπληστος αλγόριθμος Huffman είναι ορθός, θα δείξουμε ότι το πρόβλημα του προσδιορισμού ενός βέλτιστου απροθηματικού κώδικα διαθέτει τις ιδιότητες της άπληστης επιλογής και της βέλτιστης υποδομής.

Huffman

Λήμμα Άπληστης Επιλογής

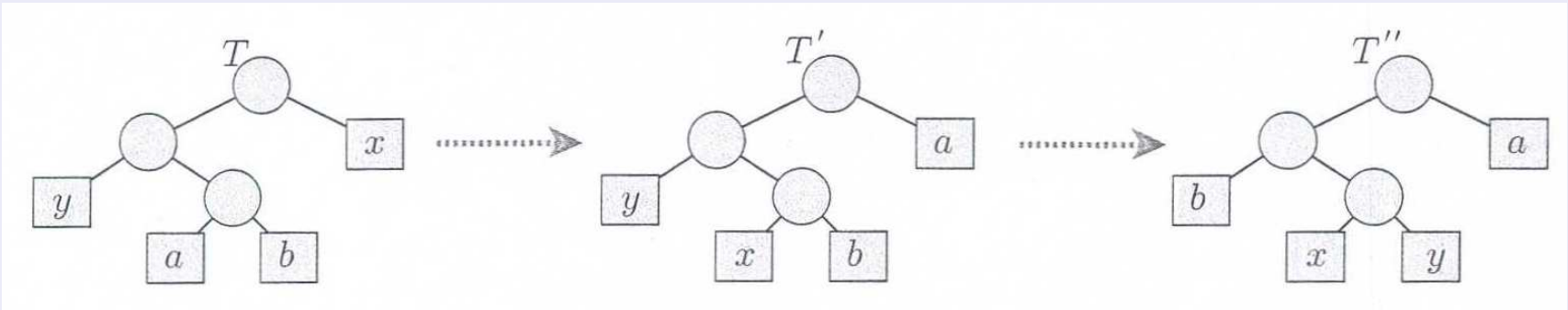
Λήμμα

Έστω ένα αλφάβητο του οποίου ο κάθε χαρακτήρας $c \in C$ έχει συχνότητα $f[c]$, και έστω x, y δυο χαρακτήρες του C οι οποίοι έχουν τις μικρότερες συχνότητες. Στην περίπτωση αυτή, υπάρχει βέλτιστος απροθηματικός κώδικας για το αλφάβητο C στον οποίο οι κωδικοί των x, y έχουν το ίδιο μήκος και διαφέρουν μόνο στο τελευταίο δυφίο.

Huffman

Λήμμα Άπληστης Επιλογής - Απόδειξη

Η κεντρική ιδέα της απόδειξης φαίνεται στο παρακάτω σχήμα.



Αν μπορούμε να κατασκευάσουμε ένα τέτοιο δένδρο, τότε οι κωδικοί των δυο χαρακτήρων θα έχουν το ίδιο μήκος και θα διαφέρουν μόνο στο τελευταίο δυφίο.

Λήμμα Άπληστης Επιλογής - Απόδειξη (συν.)

- Χωρίς απώλεια γενικότητας, μπορούμε να υποθέσουμε ότι

$$f[a] \leq f[b] \text{ και } f[x] \leq f[y]$$

- Αλλά

$$f[x] \leq f[a] \text{ και } f[c] \leq f[b]$$

- Εν συνεχεία εναλάσσουμε τις θέσεις των χαρακτήρων a και x στο δένδρο T και προκύπτει το δένδρο T'.

Απόδειξη (συν.)

Η διαφορά κόστους μεταξύ των δένδρων T, T' είναι

$$\begin{aligned} B(T) - B(T') &= \sum_{c \in C} f(c) d_T(c) - \sum_{c \in C} f(c) d_{T'}(c) \\ &= f[x] d_T(x) + f[a] d_T(a) - f[x] d_{T'}(x) - f[a] d_{T'}(a) \\ &= f[x] d_T(x) + f[a] d_T(a) - f[x] d_T(a) - f[a] d_T(x) \\ &= (f[a] - f[x])(d_T(a) - d_T(x)) \\ &\geq 0 \end{aligned}$$

Όμοια, η εναλλαγή των y, b δεν αυξάνει το κόστος, οπότε

$$B(T') - B(T'') \geq 0.$$

Απόδειξη (συν.)

Επομένως, από τις δυο τελευταίες σχέσεις προκύπτει ότι

$$B(T'') \leq B(T).$$

Αλλά το δένδρο T είναι βέλτιστο, που σημαίνει ότι

$$B(T) \leq B(T'').$$

Από τις δυο τελευταίες σχέσεις συνεπάγεται ότι

$$B(T'') = B(T).$$

Συνεπώς, το T'' είναι ένα βέλτιστο δένδρο στο οποίο οι χαρακτήρες x, y αναπαρίστανται από αδελφικούς καταληκτικούς κόμβους μέγιστου βάθους.

Huffman

Λήμμα Άπληστης Επιλογής

- Το Λήμμα μας εξασφαλίζει ότι η διαδικασία της κατασκευής ενός βέλτιστου δένδρου μέσω συγχωνεύσεων μπορεί να ξεκινήσει, χωρίς απώλεια της γενικότητας, με την άπληστη επιλογή της συγχώνευσης των δυο χαρακτήρων με τις ελάχιστες συχνότητες.
- Γιατί η επιλογή αυτή είναι άπληστη;
- Από όλες τις δυνατές συγχωνεύσεις σε κάθε βήμα η διαδικασία HUFFMAN επιλέγει εκείνη που επισύρει τη μικρότερη δαπάνη.

Huffman

Λήμμα Βέλτιστης Υποδομής

Λήμμα

- *Έστω* $C' = C - \{x, y\} \bigcup \{z\}$ *με* $f[z] = f[x] + f[y]$.
- *Έστω* T' *οποιοδήποτε βέλτιστο δένδρο για το αλφάβητο* C'.
- *Το δένδρο* T *που προκύπτει αν αντικαταστήσουμε στο* T' *τον καταληκτικό κόμβο του χαρακτήρα* z *με έναν εσωτερικό κόμβο ο οποίος έχει ως θυγατρικούς τους* x, y *αναπαριστά έναν βέλτιστο απροθηματικό κώδικα για το αλφάβητο* C.

Λήμμα Βέλτιστης Υποδομής-Απόδειξη

- Ως πρώτο βήμα, θα δείξουμε ότι το κόστος $B(T)$ του δένδρου T μπορεί να εκφραστεί συναρτήσει του κόστους $B(T')$ του δένδρου T'.
- Για κάθε χαρακτήρα $c \in C - \{x, y\}$ έχουμε $d_T(c) = d_{T'}(c)$ και επομένως $f[c]d_{T'}(c) = f[c]d_{T'}(c)$.
- Δεδομένου ότι $d_T(x) = d_T(y) = d_{T'}(z) + 1$ έχουμε

$$\begin{aligned} f[x]d_T(x) + f[y]d_T(y) &= (f[x] + f[y])(d_{T'}(z) + 1) \\ &= f[z]d_{T'}(z) + f[x] + f[y] \end{aligned}$$

απ' όπου έπεται ότι	$B(T) = B(T') + f[x] + f[y]$
ή ισοδύναμα	$B(T') = B(T) - f[x] - f[y]$

Λήμμα Βέλτιστης Υποδομής-Απόδειξη

Εν συνεχεία θα αποδείξουμε το λήμμα με εις άτοπον απαγωγή.

- Ας υποθέσουμε ότι το T δεν αντιπροσωπεύει έναν βέλτιστο απροθηματικό κώδικα για το αλφάβητο C.
- Στην περίπτωση αυτή υπάρχει ένα δένδρο T'' τέτοιο ώστε

$$B(T'') < B(T).$$

- Χωρίς απώλεια της γενικότητας (με βάση το Λήμμα της Απληστίας) τα x, y αναπαρίστανται από αδελφικούς κόμβους στο T''.
- Έστω T''' το δένδρο που προκύπτει αν αντικαταστήσουμε τον κοινό πατρικό κόμβο των x, y στο T'' από έναν καταληκτικό κόμβο z με συχνότητα

$$f[z] = f[x] + f[y].$$

Λήμμα Βέλτιστης Υποδομής-Απόδειξη

- Στην περίπτωση αυτή

$$\begin{aligned} B(T''') &= B(T'') - f[x] - f[y] \\ &< B(T) - f[x] - f[y] \\ &= B(T') \text{ αντίφαση!} \end{aligned}$$

διότι το T' αντιπροσωπεύει ένα βέλτιστο απροθηματικό κώδικα για το αλφάβητο C'.

- Επομένως το T αντιπροσωπεύει ένα βέλτιστο απροθηματικό κώδικα για το αλφάβητο C.

Huffman

Θεωρεμ

Η διαδικασία HUFFMAN *παράγει βέλτιστο απροθηματικό κώδικα.*

Απόδειξη.

Το θεώρημα έπεται άμεσα από τα Λήμματα άπληστης επιλογής και βέλτιστης υποδομής. □