Αλγόριθμοι και Πολυπλοκότητα

2η Σειρά Γραπτών Ασκήσεων

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Αμπατζή Ναυσικά (03117198)

7ο Εξάμηνο - Ροή Λ

Ακαδημαϊκό Έτος 2020-2021

1 Άσκηση 1: Δίσκοι και Σημεία

Δεδομένα : n διαφορετικά σημεία, ευθεία l στο επίπεδο και δίσκοι ακτίνας r με κέντρα πάνω στην ευθεία
Ζητούμενο : Ελαχιστοποίηση των δίσκων ώστε να καλυφθούν τα n σημεία.

1.1 Διατύπωση του Greedy Αλγορίθμου

Θεωρούμε ότι για κάθε σημείο j στο επίπεδο που απέχει απόσταση d από την ευθεία l, με $d \leq r$, υπάρχουν δύο σημεία k_j και y_j πάνω στην ευθεία l, ώστε η μεταξύ τους απόσταση, αλλά και η απόσταση του καθενός από το j να ισούται με r. Για να περιέχεται το σημείο j μέσα σε έναν δίσκο θα πρέπει το κέντρο του δίσκου αυτού να βρίσκεται ανάμεσα στο διάστημα $[k_j, y_j]$ της ευθείας.

Επομένως ξεκινάμε την λύση του προβλήματος θεωρώντας ότι η ευθεία l είναι ο άξονας (0,x). Υπολογίζουμε τις θέσεις όλων των ζευγών k_j,y_j, για $j \in [1, n]$.
Έπειτα ταξινομούμε σε έναν πίνακα τα ζεύγη k_j, y_j, κατά αύξουσα απόστασή του k_j από την αρχή του άξονα x.

1.Επιλέγουμε το ζεύγος (k_j,y_j) που αποτελεί το πρώτο στοιχείο του πίνακα.
2.Τοποθετούμε δίκσο με κέντρο το y_j.
3.Προχωράμε στο επόμενο ζεύγος ($k_{j'}$,$y_{j'}$) του πίνακα.
4α.Εάν το $k_{j'}$ βρίσκεται μέσα στο διάστημα του άξονα που ορίζει ο προηγούμενος κύκλος και $k_{j'} < y_j$ αυτό σημαίνει ότι το σημείο j' καλύπτεται από τον ήδη υπάρχων δίσκο.
4β.Διαφορετικά τοποθετούμε δίσκο με κέντρο το $y_{j'}$.
5.Επαναλαμβάνουμε τα βήματα 3-4.β μέχρι να εηετάσουμε όλα τα ζεύγη $k_{j'}$,$y_{j'}$.

Για τον υπολογισμό των σημείων k_j,y_j απαιτείται χρόνος O(n) και για την ταξινόμηση των ζευγών σε πίνακα O(nlogn). Τέλος για την τοποθέτηση δίσκων θα χρειαστεί το πολύ χρόνος O(n). Άρα η πολυπλοκότητα του αλγορίθμου είναι O(nlogn).

1.2 Απόδειξη Ορθότητας

Θα εφαρμόσουμε επαγωγή στο πλήθος των δίσκων. Υποθέτουμε ότι τα ζεύγη (k_j, y_j), είναι πάντα ταξινομημένα κατά αύξουσα απόστασή του k_j από την αρχή του άξονα x.
Συμβολισμοί :
Έστω A το σύνολο των n ζευγών (k_j,y_j)(2n σημείων) και A_j το σύνολο των ζευγών που βρίσκεται δεξιότερα από το ζεύγος (k_j,y_j) του πίνακα.
Έστω c^*(A) το πλήθος των δίσκων της βέλτιστης λύσης και c(A) το πλήθος των δίσκων του greedy αλγορίθμου.

- Βάση Επαγωγής: Εάν υπάρχει μόνο 1 ζεύγος (k_j,y_j) τότε απαιτείται ένας δίσκος για να καλυφθεί.
- Υπόθεση : Έστω ότι $\forall A' <$A, ισχύει ότι $c(A') \leq c^*(A')$.
- Επαγωγικό Βήμα: Θα αποδείξουμε ότι για κάθε σύνολο A n δίσκων, ο greedy αλγόριθμος χρησιμοποιεί το πολύ τόσους δίσκους όσους ο βέλτιστος. Δηλαδή, $\forall A$, |A| = n ισχύει ότι $c(A) \leq c^*(A)$.

Για τον greedy αλγόριθμο ισχύει ότι c(A) = 1[k_1,y_1] + $c(A_1)$, δηλαδή επιλέγεται το πρώτο ζεύγος (k,y) και τοποθετείται δίσκος με κέντρο το y.
Για τον βέλτιστο αλγόριθμο ισχύει ότι c^*(A) = 1[k_j,y_j] + min(j) $c^*(A_j)$. Δηλαδή ο βέλτιστος αλγόριθμος διαλέγει ένα ζεύγος (k_j,y_j αρχικά, ώστε να ελαχιστοποιήσει το πλήθος των απαιτούμενων δίσκων.

Επιχείρημα ανταλλαγής: Στο πρόβλημα αυτό ισχύει η μονοτονικότητα, δηλαδή για $\forall A' < A''$, ισχύει ότι $c^*(A') \leq c^*(A'')$ και $c(A') \leq c(A'')$

Έχουμε ότι:
$c(A) = 1[k_1, y_1]+c(A_1) \leq 1+c^*(A_1)$ (από την επαγωγική υπόθεση) $\leq 1+min[j]c^*(A_j) = c^*(A)$ (από το επιχείρημα ανταλλαγής) $= c^*(A)$
Επομένως αποδείχθηκε ότι ο αλγόριθμος που περιγράφτηκε είναι ορθός.

2 Άσκηση 2: Μεταφορά δεμάτων

2.1 Εξέταση των πιθανών επιλογών

- Στοίβαξη με βάση το βάρος: Το κρίτηριο αυτό δεν εγγυάται τον υπολογισμό μιας ασφαλούς σύνταξης. Για παράδειγμα εάν έχουμε 3 πακέτα με $w_1 = 10$ και $d_1 = 9$, $w_2 = 8$ και $d_2 = 3$ και $w_1 = 6$ και $d_1 = 5$ παρατηρούμε ότι το πακέτο 2 το οποίο θα είναι στη μέση της στοίβας δε θα αντέξει το βάρος του πακέτου 1.
- Στοίβαξη με βάση την αντοχή: Το κρίτηριο αυτό δεν εγγυάται επίσης τον υπολογισμό μιας ασφαλούς σύνταξης.Για παράδειγμα εάν έχουμε ξανά τα 3 πακέτα με $w_1 = 10$ και $d_1 = 9$, $w_2 = 8$ και $d_2 = 3$ και $w_1 = 6$ και $d_1 = 5$ παρατηρούμε ότι το πακέτο 1 το οποίο θα είναι στη αρχή της στοίβας δε θα αντέξει το βάρος των άλλων δύο πακέτων, το οποίο είναι ίσο με 14.
- Στοίβαση με βάση το άθροισμα βάρους και αντοχής: Το κριτήριο αυτό μπορεί να εγγυηθεί τον υπολογισμό μίας ασφαλούς σύνταξης, εάν αυτή υπάρχει. Για να το αποδείξουμε αυτό αρκεί να δείξουμε ότι κάθε συνεχές ζεύγος δεμάτων για τα οποία ισχύει ότι $d(box_i + 1) + w(box_i + 1) < d(box_i) + w(box_i)$ (1).

Ισοδύναμα θέλουμε να αποδείξουμε ότι:

$$\sum_{j=1}^{i-1} w(box_j) + w(box_{i+1}) < d(box_i)(2)$$

Για την αρχική στοίβαξη ισχύει ότι:

$$\sum_{j=1}^{i-1} w(box_j) + w(box_i) \leq d(box_{i+1})(3)$$

Προσθέτουμε και στα δύο μέλη το $w(box_{i+1})$ και έτσι η (2) γίνεται :

$$\sum_{j=1}^{i-1} w(box_j) + w(box_i) + w(box_{i+1}) \leq d(box_{i+1}) + w(box_{i+1})(4)$$

Τώρα μπορούμε να αντικαταστήσουμε στην (1) το αριστερό μέρος ως εξής:

$$\sum_{j=1}^{i-1} w(box_j) + w(box_i) + w(box_{i+1}) \leq d(box_i) + w(box_i)$$

Αφαιρούμε το $w(box_i)$ και τελικά προκύπτει ότι:

$$\sum_{j=1}^{i-1} w(box_j) + w(box_{i+1}) \leq d(box_i)$$

2.2 Τελικός Αλγόριθμος

Από το προηγούμενο ερώτημα έχουμε, ότι ένα υποσύνολο των n πακέτων (έστω D), το οποίο μπορεί να στοιβαχθεί ασφαλώς, οδηγεί τουλάχιστον σε μία ταξινομιμένη κατά φθίνουσα σειρά βάρους και αντοχής στοίβαξη. Η στοίβαξη αυτή θα είναι ασφαλής. Έτσι για να μεγιστοποιήσουμε το κέρδος, αρκεί να ταξιμομήσουμε τα πακέτα σύμφωνα με την 3η λύση του ερωτήματος (α), να υπολογίσουμε τα κέρδη για κάθε στοίβαξη και να κρατήσουμε το μεγαλύτερο.

Ορίζουμε ως w_i, d_i, p_i, αντίστοιχα το βάρος, την αντοχή και το κόστος του i-οστού δέματος.

Εάν εφαρμόσουμε τη μέθοδο της εξαντλητικής αναζήτησης θα πρέπει να εξετάσουμε 2^n στοιβάξεις. Αυτό οδηγεί σε χρονική πολυπλοκότητα $\Omega(\mathrm{n}2^n)$. Η λύση αυτή, δηλαδή του προβλήματος είναι πάρα πολύ αργή.

Αρχή Βελτιστότητας

Σύμφωνα με τα παραπάνω η βέλτιστη λύση θα βρίσκεται στο σύνολο D* $\subseteq N = \{1, ..., n\}$. Εάν αγνοοήσουμε το αντικείμενο n από το αρχικό σύνολο με τα πακέτα τότε:

1. Εάν αυτό άνηκει στο σύνολο D*, τότε η βέλτιστη λύση θα προκύψει είτε για στοίβαξη χωρίς αυτό το πακέτο, εάν το κέρδος παραμένει το μεγαλύερο, είτε για κάποια άλλη στοίβαξη που πλέον έχει το μέγιστο κέρδος.
2. Εάν το πακέτο δεν ανήκει στο υποσύνολο D*, τότε το D*\{n} θα είναι η βέλτιστη

λύση για N\{n} πακέτα.

Εφόσον ισχύει η αρχή της βελτιστότητας, εάν γνωρίζουμε τη βέλτιστη αξία για πακέτα N\{n} και στοιβάξεις επιτρεπτού βάρους W και W$-w_n$, μπορούμε να αποφασίσουμε εάν το πακέτο θα προστεθεί στη στοίβαξη ή οχι θα προκύψει βέλτιστη λύση.

Διατύπωση αναδρομικής σχέσης

Θα χρησιμοποιήσουμε δυναμικό προγραμματισμό, κατ'αντιστοιχία με το πρόβλημα knapsack. Στοιβάζουμε τα δέματα σε φθίνουσα βάρους και αντοχής. Για κάθε ζεύγος (i,w), για το οποίο ισχύει ότι $d_{i-1} \geq w$ ορίζουμε ως C(i,w) το μέγιστο κέρδος μεταφορικών ύστερα από ασφαλή στοίβαξη των δεμάτων i έως n και συνολικού βάρους το πολύ w.

$$C(i,w) = \begin{cases} 0 & i > n \vee W \leq 0 \\ max\{C(i+1,w), C(i+1, min(w-w_i, d_i) + p_i)\} & i = 0,...,n \wedge w_i = 1,...,W \\ C(i+1,w) & w_i > W \end{cases}$$

Η μέγιστη δυνατή αξία που δύναται να προκύψει είναι η $C(0, \sum w_i)$ Παρατηρούμε ότι η αναδρομική σχέση λύνει επικαλυπτόμενα υποπροβλήματα. Έτσι εάν αυτά δεν αποθηκεύονται κάπου θα πρέπει να υπολογίζονται κάθε φορά από την αρχή. Το αποτέλεσμα θα είναι μεγάλο κόστος. Για την επίλυση του παραπάνω ζητήματος θα χρησιμοποιήσουμε έναν πίνακα Π διαστάσεων nx$sumw_i$. Κατ' αντιστοιχία με την αναδρομική σχέση στο πρόβλημα του knapsack, ο πίνακας Π συμπληρώνεται σε ένα πέρασμα και έχει χωρικό και χρονικό κόστος O(n * C), όπου C= $sumw_i$. Για την εύρεση του υποσυνόλου του πίνακα των πακέτων, ώστε να έχουμε στοίβαξη με μέγιστο κέρδος η διαδικασία είναι η ίδια με αυτή για το knapsack.

Η ορθότητα του αλγορίθμου προκύπτει από το γεγονός ότι ο πίνακας Π περιέχει όλες τις μέγιστες δυνατές στοιβάξεις πακέτων με κάποιο όριο βάρους. Για να λύσουμε το πρόβλημα επιλέγουμε τη στοίβαξη με το μέγιστο βάρος.

Πολυπλοκότητα

Η χρονική πολυπλοκότητα του αλγορίθμου είναι O(nlogn +n*C) (ψευδοπολυωνυμική). Το πιθανότερο είναι ότι δεν μπορούμε να τη βελτιώσουμε καθώς το πρόβλημα αυτό μπορεί να αναχθεί στο διακριτό πρόβλημα του σακιδίου,που είναι NP πλήρες.

3 Άσκηση 3: Τριγωνοποίηση πολυγώνου

Θεωρούμε ένα κυρτό πολύγωνο P = $(u_0, u_1, ..., u_{n-1})$ με n κορυφές, δίνοντας τις κορυφές του με φορά αντίθετη από αυτή των δεικτών του ρολογιού. Θέλουμε να υπολογίσουμε μία τριγωνοποίηση του T(P), ώστε το συνολικό μήκος των πλευρών των τριγώνων που προκύπτουν να είναι το ελάχιστο.

Συμβολίζουμε με $\Delta(u_i, u_j, u_k)$ το συνολικό μήκος των πλευρών ενός τριγώνου με κορυφές u_i, u_j, u_k που προκύπτει από την τριγωνοποίηση. Για τον υπολογισμό του $\Delta(u_i, u_j, u_k)$ ισχύει ότι :

$\Delta(u_i, u_j, u_k)$ = dist(i,j) + (i,k) + dist(j,k) και

dist(i,j) = $\sqrt{(x[i] - x[j])^2 + (y[i] - y[j])^2}$, όπου x,y οι κορυφές του πολυγώνου.

Γνωρίζουμε ότι κάθε τριγωνοποίηση ενός κυρτού πολυγώνου n πλευρών έχει n-3 χορδές και χωρίζει το τρίγωνο σε n-2 τρίγωνα.

Αρχή Βελτιστότητας

Έστω t ένα τρίγωνο που ανήκει στην βέλτιστη τριγωνοποίηση T(P). Το t "χωρίζει" το πολύγωνο σε δύο επιμέρους βέλτιστα τριγωνοποιημένα πολύγωνα P_1 και P_2 και ισχύει ότι: T(P) = $T(P_1) \cup t \cup T(P_2)$.

Για να αποδείξουμε τον ισχυρισμό αυτό υποθέτουμε ότι το T(P1) δεν είναι ένα βέλτιστα τριγωνοποιημένο πολύγωνο, αλλά το βέλτιστο είναι το T(P1'). Έτσι για να έχουμε βέλτιστη τριγωνοποίηση θα πρέπει να αντικαταστήσουμε το T(P1) με το T(P1') . Έτσι προκύπτει η βέλτριστη τριγωνοποίηση T(P'). Όμως αυτό έρχεται σε αντίθεση με την αρχική μας υπόθεση, το ότι δηλαδή η T(P) είναι η βέλτιστη λύση. Ομοίως και για το T(P2).

Η ιδέα για την επίλυση του προβλήματος είναι η αναγωγή του σε πρόβλημα με μικρότερα πολύγωνα.

Αλγόριθμος

1. Επιλέγουμε μία πλευρά x του πολυγώνου P. Η x είναι πλευρά ένος και μόνο τριγώνου t στην βέλτιστη τριγωνοποίηση, το οποίο μας είναι άγνωστο.
2. Για όλα τα n-2 πιθανά τρίγωνα με βάση την ακμή x, λύνουμε το πρόβλημα αναδρομικά για τα δύο υπο-πολύγωνα που προκύπτουν από τον διαχωρισμό τους από το τρίγωνο με βάση x και κρατάμε την καλύτερη λύση.

Ο χρόνος εκτέλεσης της αναδρομής προκύπτει ως εξής:

$$T(n) = \sum_{i=1}^{n-1}(O(1) + T(i) + T(n-i-1))$$

όπου O(1) ο χρόνος για να διαχωριστεί το πολύγωνο σε δύο υπο-πολύγωνα από το κάθε τρίγωνο, T(i) ο χρόνος που μένει για να εξεταστεί το αριστερό υπό-πολύγωνο και T(n-i+1) ο χρόνος που μένει για να εξεταστεί το δεξί υπό-πολύγωνο.
Έχουμε ότι $\mathrm{T(n)} = \mathrm{O(n)} + 2 * \sum 2^{n-1} T(i) > 2\mathrm{T(n\text{-}1)} = 2^n - 1\mathrm{T(1)} = \Omega(2^n)$ Είναι δηλαδή εκθετικός. Γι'αυτόν τον λόγο θα χρησιμοποιήσουμε δυναμικό πργραμματισμό.

Αναδρομική Σχέση

Έστω :

- $P_{ij} = (u_i, u_{i+1}, ...u_j)$, ένα μέρος του πολυγώνου P.
- L(i,j) το άθροισμα του μήκους των πλευρών του T(P_{ij}).

$$L(i,j) = \begin{cases} 0 & , j = i+1 \\ min_{i<k<j}\{u(i,j,k) + L(i,k) + L(k,j)\} & , otherwise \end{cases}$$

Η βέλτιστη τριγωνοποίηση προκύπτει από το L(1,n). Παρατηρούμε ότι η αναδρομική σχέση λύνει επικαλυπτόμενα υποπροβλήματα. Έτσι εάν αυτά δεν αποθηκεύονται κάπου θα πρέπει να υπολογίζονται κάθε φορά από την αρχή. Το αποτέλεσμα θα είναι μεγάλο κόστος. Για

την επίλυση του παραπάνω ζητήματος θα χρησιμοποιήσουμε έναν πίνακα S[i,j], ο οποίος συμπληρώνεται κατάλληλα.

Τώρα μέσω του πίνακα S μπορεί να προκύψει η βέλτιστη τριγωνοποίηση.Εκτελούμε την αναδρομική πράξη Δ(1,n,S[n]). Σε κάθε βήμα υπάρχουν δύο σημεία, έστω x και y, τα οποία ορίζουν τη βάση του τριγώνου και την τρίτη του κορυφή S[x,y].

Απόδειξη Ορθότητας

Τελικά προκύπτει ότι το L_{1n} είναι το άθροισμα όλων των ακμών του πολυγώνου συν 2 φορές το άθροισμα όλων των διαγωνίων. Εφόσον το άθροισμα όλων των ακμών είναι ανεξάρτητο της βελτιστοποίησης, το L_{1n} ελαχιστοποιείται αν και μόνο αν το άθροισμα όλων των διαγωνίων είναι ελάχιστο κάτι το οποίο ο αλγόριθμος βρίσκει.

Πολυπλοκότητα

Η κατασκευή του πίνακα S έχει χρονική πολυπλοκότητα $O(n^3)$ και η ανακατασκευή του T(P) μέσω του S, O(n), επομένως ο αλγόρθμος έχει χρονική πολυπλοκότητα $O(n^3)$.

4 Άσκηση 4: Τοποθέτηση Στεγάστρων(και Κυρτό Κάλυμμα)

4.1 Αποδοτικός Αλγόριθμος

Αρχή Βελτιστότητας

Θεωρούμε ότι η βέλτιστη λύση θα βρεθεί από τις επιμέρους βέλτιστες λύσεις των μικρότερων προβλημάτων, που προκύπτουν εάν χωρίσουμε κατάλληλα το σύνολο των n σημείων.
Θεωρούμε ότι η βέλτιστη λύση είναι η K και ότι το στέγαστρο που καλύπτει το n-οστό σημείο ξεκινάει από το σημείο j. Θεωρούμε ως K_1 την επιλογή στεγάστρων για τα σημεία [0,...,j-1] και K_2 την επιλογή στεγάστρων για τα υπόλοιπα. Έστω ότι η K_1 δεν είναι η βέλτριστη επιλογή και αντί γι'αυτή είναι η K_1'. Τότε εάν αντικαταστήσουμε την K_1 με την K_1' η συνολική λύση θα είναι μία διαφορετική από την K. Επομένως η K δεν ήταν η βέλτιστη, όπως αρχικά υποθέσαμε, κάτι το οποίο είναι άτοπο. Άρα οι K_1 και K_2 είναι οι βέλτιστες λύσεις για τα δύο αυτά υποπροβλήματα.

Επομένως,την κάλυψη με το μικρότερο δυνατό κόστος των σημείων $x_1, ..., x_n$ με στέγαστρα, θα πρέπει για κάθε σημείο x_i να βρούμε ένα σημείο x_j, με $x_j < x_i$, όπου "ακυρώνοντας" το στέγαστρο ανάμεσα στα σημεία x_{j-1} και x_j και τοποθετώντας το τελευταίο στέγαστρο ανάμεσα στα σημεία x_j και x_i να έχουμε το ελάχιστο κόστος.

Αναδρομική Σχέση

Ορίζουμε ως cost[i] το ελάχιστο κόστος κάλυψης των n σημείων με στέγαστρα.

$$cost(i) = \begin{cases} 0 & , i = 0 \\ C & , i = 1 \\ min_{i>j}\{cost[j-1] + (x[i] - x[j])^2 + C)\} & , otherwise \end{cases}$$

Το ελάχιστο κόστος που ψάχνουμε είναι το cost(n).

Δημιουργούμε έναν πίνακα Π όπου αποθηκεύουμε τα αποτελέσματα της αναδρομής κάθε φορα, ώστε να μην υπολογίζονται από την αρχή ζητούμενα που είχαν υπολογιστεί σε προηγούμενα στάδια.

Πολυπλοκότητα

Η χρονική πολυπλοκότητα του αλγορίθμου είναι $O(n^2)$.

5 Άσκηση 5: Καλύπτοντας ένα Δέντρο

5.1 Ερώτημα (α)

Έστω ένα υπόδεντρο(K) με ρίζα u και ένα τμήμα A αυτού με μέγεθος j. Έστω ότι η ρίζα του υποδέντρου απέχει l από τον κοντινότερο πρόγονό της στο K. Υποθέτουμε ότι το A έχει ελάχιστο κόστος κάλυψης ίσο με cover(dist,j,u). Υπάρχουν δύο περιπτώσεις για την κορυφή u:

1. Η κορυφή u να μην ανήκει στο K.
2. Η κορυφή u να ανήκει στο K.

Έτσι το κόστος cover(dist,j,u) ορίζεται ως cover(dist,j,u) = min(u$\in$ K,u $\notin$ K). Η περίπτωση 1. λύνεται με την εύρεση του μεγαλύτερου κόστους για τα δύο υπόδεντρα, των οποίων οι ρίζες root1 και root2 έχουν απόσταση 1 από το A, ώστε τα υπόλοιπα j-1 στοιχεία του A να διαμοιραστούν βέλτιστα. Η περίσπτωση 2. λύνεται με την εύρεση του μέγιστου κόστους ανάμεσα στα δύο υπόδεντρα με ρίζες root1 και root2, και στη ρίζα, ώστε τα j στοιχεία να χωριστούν βέλτιστα. Ακολουθούν οι αναδρομικές σχέσεις των λύσεων που περιγράφτηκαν:

1. Costin = min(i=0,...,j-1)[max [cover(dist,j,root1(u), cover(1,i-j-1,root2(u)]]
2. Costout = min(i=0,...,j-1)[max [cover(dist +1 ,j,root1(u), cover(dist+1,j-i,root2(u)], dist]

Αναζητούμε το κόστος cover(0,k,r). Για να αποθηκεύουμε τα ενδοιάμεσα βήματα της αναδρομής χρησιμοποιούμε έναν πίνακα. Αρχίζουμε από τα φύλλα. Για κάθε επίπεδο κάθε φορά Θ(n), βρίσουμε το κόστος για dist = (0,...,n-1) Θ(n) και για i = (0,...,k) Θ(n). Έτσι βρίσκουμε το μέγιστο κόστος κάλυψης. Για να βρούμε τελικά το σύνολο κάλυψης, αποθηκεύουμε σε έναν ακόμα πίνακα, ποια από τις περιπτώσεις 1 και 2 είναι η καλύτερη επιλογή και τον διαχωρισμό των j κορυφών Θ(k). Από τα παραπάνω βήματα και τα επιμέρους κόστη η πολυπλοκότητα προκύπτει ίση με $O(n^2k^2)$.

5.2 Ερώτημα (β)

Διατύπωση του αλγορίθμου Greedy(z) Για τη λύση αυτού του ερωτήματος θα χρησιμοποιήσουμε greedy αλγόριθμο. Αρχικά επιλέγουμε ένα φύλλο από το κατώτερο επίπεδο του δέντρου και βρίσκουμε την κορυφή που απέχει z από αυτό. Βάζουμε την κορυφή αυτή στο K. Τέλος βρίσκουμε ποιο υπόδεντρο έχει ως ρίζα του αυτή την κορυφή και το αφαιρούμε.

Απόδειξη Ορθότητας Αρχή Βελτιστότητας

Έστω S η βέλτιστη λύση για το πρόβλημα. Επιλέγουμε ένα υπόδεντρο που βρίσκεται κάτω από κάποια κορυφή του συνόλου K και το αφαιρούμε. Τότε το K (με όσα στοιχεία έχει πλέον διαθέσιμα) καλύπτει πλήρως το δέντρο που προέκυψε από την αφαίρεση.

Έστω ότι αρχικά επιλέχθηκε η κορυφή u. Για να δείξουμε ότι υπάρχει βέλτιστη λύση που περιέχει την u, θα υποθέσουμε ότι η S δεν περιέχει την u. Συνεπώς θα πρέπει να περιέχει κάποιον απόγονό της, έστω u'. Έτσι εάν αντικαταστήσουμε την u με την u',μπορεί να προκύψει λύση με τον ίδιο ή μικρότερο αριθμό κορυφών κάλυψης.

Τώρα θα χρησιμοποιήσουμε τον αλγόριθμο αυτόν για να λύσουμε το ερώτημα (α). Με binary search αναζητούμε το καλύτερο z, ώστε ο Greedy(z) να επιστρέψει ένα το πολύ k μεγέθους σύνολο. Αυτό μπορέι να συμβεί, διότι ισχύει η μονοτονικότητα στον αλγόριθμο. Η πολυπλοκότητα του αλγορίθμου είναι O(nlogn), δίοτι έχουμε κόστος O(logn) για την δυαδική αναζήτηση και θα γίνουν το πολύ n αναζητήσεις($z \leq n$).