ΕΘΝΙΚΌ ΜΕΤΣΌΒΙΟ ΠΟΛΥΤΕΧΝΕΊΟ

ΣΧΟΛΉ ΗΛΕΚΤΡΟΛΌΓΩΝ ΜΗΧΑΝΙΚΏΝ ΚΑΙ ΜΗΧΑΝΙΚΏΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΑΛΓΌΡΙΘΜΟΙ & ΠΟΛΥΠΛΟΚΟΤΗΤΑ

(2020-2021)

2η Σειρά Γραπτών Ασκήσεων

Ονοματεπώνυμο:

- Χρήστος Τσούφης

Αριθμός Μητρώου:

- 03117176

1η Άσκηση – Δίσκοι και Σημεία

Δίνονται n διαφορετικά μεταξύ τους σημεία και μια ευθεία l στο επίπεδο. Κάθε ένα από τα σημεία βρίσκεται σε κάθετη απόσταση το πολύ r από την ευθεία l. Αντικειμενικός στόχος είναι να καλυφθούν όλα τα n σημεία με ελάχιστο πλήθος δίσκων ακτίνας r.

Απαραίτητη Μαθηματική Θεωρία:

Η εύρεση της ορθής τοποθέτησης του κέντρου του δίσκου πάνω στην ευθεία l ώστε να ανήκει στην περιφέρεια το σημείο R υπολογίζεται από σύγκριση του R με το R' που προκύπτει από την επίλυση του συστήματος που δημιουργείται όταν η κάθετη ευθεία που περνά από το σημείο R, τέμνει την ευθεία στην οποία πρέπει να ανήκει το κέντρο του δίσκου. Αν η απόσταση μεταξύ τους είναι μικρότερη της ακτίνας r, προκύπτει μια δευτεροβάθμια εξίσωση η οποία δίνει δύο λύσεις και από αυτές επιλέγεται η δεξιότερη ώστε να προκύψει το κέντρο του δίσκου. Ειδική περίπτωση αποτελεί το ενδεχόμενο η απόσταση μεταξύ τους να είναι ίση με r, οπότε το κέντρο του δίσκου θα είναι το R'.

Ο Αλγόριθμος είναι ο εξής:

- Έστω ότι η εξίσωση ευθείας έχει την μορφή $x = a$.
    - Τότε, πρώτα επιλέγονται τα σημεία και αυτά ταξινομούνται ως προς y.
- Έστω ότι η εξίσωση ευθείας έχει την μορφή $y = a \cdot x + b$.
    - Τότε, πρώτα επιλέγονται τα σημεία και αυτά ταξινομούνται ως προς x.
    - Έπειτα, διατρέχοντας την λίστα με τα σημεία, επιλέγεται το πρώτο που δεν καλύπτει ο δίσκος (δηλαδή το αριστερότερο, αφού αυτό έχει το μικρότερο x) και γίνεται προσπάθεια για δημιουργία κύκλου τέτοιου ώστε το σημείο αυτό να ανήκει στην περιφέρεια του κύκλου.
    - Στην συνέχεια, επειδή προκύπτει δευτεροβάθμια εξίσωση, αυτό συνεπάγεται δύο κέντρα κύκλων και κάθε φορά επιλέγεται ο δεξιότερος.
    - Σημειώνεται ότι, δεν δημιουργείται αμέσως αυτός ο κύκλος καθώς κάθε φορά δεσμεύεται προσωρινά (temporary variable) το εκάστοτε κέντρο (x, y) που προέκυψε.
    - Ύστερα, συνεχίζεται αυτή η διαδικασία για τα υπόλοιπα σημεία της λίστας και ελέγχεται κατά πόσο ανήκουν μέσα σε αυτόν τον κύκλο που δημιουργείται.
    - Μόλις βρεθεί κάποιο σημείο που δεν ανήκει μέσα στον κύκλο, τότε δημιουργείται ένας νέος κύκλος με τον ίδιο τρόπο ώστε το σημείο που δεν ανήκει στον προηγούμενο, να βρίσκεται στην περιφέρεια του νέου, και ελέγχεται αν ο νέος κύκλος έχει το κέντρο του αριστερότερα από του πρώτου.
        - Αν ναι, τότε αποδεσμεύεται ο πρώτος και διατηρείται ο νέος (new temporary variable).
        - Η μόνη περίπτωση οριστικοποίησης ενός κύκλου είναι η εύρεση ενός επόμενου κύκλου που να είναι δεξιότερα από τον προηγούμενο.

Έλεγχος Ορθότητας:

Έστω ότι υπάρχει ένας *Optimal* αλγόριθμος. Τότε, ισχύει ότι:

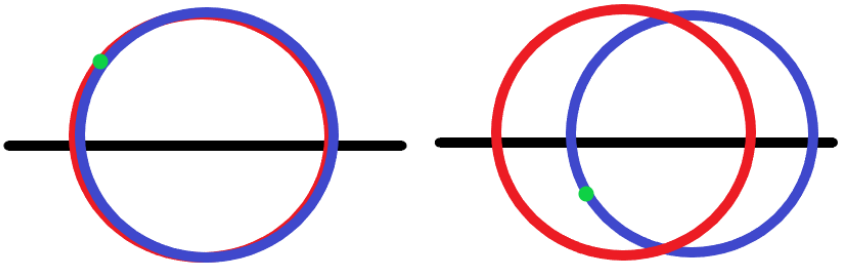
- Αν λάβει *n = 0* (καθόλου) σημεία, τότε δεν θα επιστρέψει κανένα δίσκο.
- Αν λάβει *n = 1* (ένα) σημείο, τότε θα επιστρέψει ένα δίσκο (με το σημείο είτε στην περιφέρειά του, είτε μέσα στον δίσκο).
- Αν λάβει *n* σημεία, περιγράφεται αναλυτικά παρακάτω.

Έστω ότι υπάρχει ένας *Greedy* αλγόριθμος. Τότε, ισχύει ότι:

- Αν λάβει *n = 0* (καθόλου) σημεία, τότε δεν θα επιστρέψει κανένα δίσκο.
- Αν λάβει *n = 1* (ένα) σημείο, τότε θα επιστρέψει ένα δίσκο (με το σημείο στην περιφέρειά του και όσο γίνεται δεξιότερα).
- Αν λάβει *n* σημεία, περιγράφεται αναλυτικά παρακάτω.

Ειδικότερα, αν λάβει *n* σημεία, τότε γίνεται χρήση του *Κριτήριου της Ανταλλαγής*. Με την προϋπόθεση ότι τα προηγούμενα σημεία [0, i-1] ανήκουν όλα σε τουλάχιστον ένα δίσκο που δημιουργήθηκαν από τον Optimal, υπολογίζεται το i-οστό σημείο. Συγκεκριμένα, επειδή το i-οστό σημείο δεν ανήκει σε κανένα από τους προηγούμενους δίσκους, γίνεται ανταλλαγή μεταξύ του δίσκου που δημιούργησε ο Optimal και καλύπτει το i-οστό σημείο, με ένα δίσκο που θα δημιουργούσε ο Greedy. Έτσι, προκύπτουν οι εξής καταστάσεις:

- Είτε θα δημιουργήσει ο Optimal ένα κύκλο (κόκκινο) και θα ανήκει πάνω στην περιφέρεια αυτό το σημείο (πράσινο) και τότε θα είναι ακριβώς ο ίδιος κύκλος που έχει δημιουργήσει και ο Greedy (μπλε) (Σχήμα 1).
- Είτε θα δημιουργήσει ο Optimal ένα κύκλο (κόκκινο) και θα ανήκει μέσα στον κύκλο αυτό το σημείο (πράσινο) και όχι στην περιφέρειά του (Σχήμα 2). Οπότε, όταν ο Greedy είναι δεξιότερα τότε αυτό σημαίνει ότι ο δίσκος του Greedy, συμπεριλαμβάνει όσα σημεία είχε και ο δίσκος του Optimal ή και παραπάνω.



Σχήμα 1

Σχήμα 2

Συνεπώς, σε αυτήν την άσκηση, τόσο ο Optimal, όσο και ο Greedy, αφού θα επιστρέφουν την βέλτιστη λύση.

Πολυπλοκότητα:

Η πολυπλοκότητα του αλγορίθμου θα είναι O(nlogn). Αυτό προκύπτει ως εξής:

Απαιτείται sorting για τα σημεία, δηλαδή O(nlogn).

Απαιτείται να τα διατρέξει, δηλαδή O(n).

Δηλαδή O(nlogn + n) = O(nlogn).

2η Άσκηση – Μεταφορά Δεμάτων

a) Παρακάτω υπολογίζεται κάθε περίπτωση στοίβαξης:

1. Στοίβαξη με βάση το βάρος:
   Ένα παράδειγμα ασφαλούς στοίβαξης αποτελούν τα ζεύγη: (7,12), (5,7), (6,3).
   Ένα παράδειγμα μη ασφαλούς στοίβαξης αποτελούν τα ζεύγη: (7,12), (6,3), (5,7).
   Συμπέρασμα: αυτό το είδος στοίβαξης δεν οδηγεί πάντα σε ασφαλή αποτελέσματα.

2. Στοίβαξη με βάση την αντοχή:
   Ένα παράδειγμα ασφαλούς στοίβαξης αποτελούν τα ζεύγη: (8,5), (2,9), (4,2).
   Ένα παράδειγμα μη ασφαλούς στοίβαξης αποτελούν τα ζεύγη: (2,9), (8,5), (4,2).
   Συμπέρασμα: αυτό το είδος στοίβαξης δεν οδηγεί πάντα σε ασφαλή αποτελέσματα.

3. Στοίβαξη με βάση το άθροισμα βάρους & αντοχής:
   Ένα παράδειγμα ασφαλούς στοίβαξης αποτελούν τα ζεύγη: (2,5), (3,4), (4,9).
   Ένα ακόμη παράδειγμα ασφαλούς στοίβαξης αποτελούν τα ζεύγη: (4,9), (2,5), (3,4).
   Συμπέρασμα: αυτό το είδος στοίβαξης οδηγεί πάντα σε ασφαλή αποτελέσματα.

   Απόδειξη:
   Για την απόδειξη, γίνεται χρήση του Κριτήριου Ανταλλαγής και θα δειχθεί ότι η στοίβαξη με το τελευταίο κριτήριο δίνει πάντα ασφαλή λύση, εφόσον αυτή υπάρχει.

   Προϋποθέσεις:
   - Έστω μια ασφαλής λύση για την οποία δεν ισχύει το τελευταίο κριτήριο.
   - Για κάθε πακέτο i με (w_i, d_i) ισχύει σίγουρα ότι $d_i \geq w_{i+1} + w_{i+2} + \ldots + w_n$.

   Θα δειχθεί ότι η στοίβαξη μπορεί να παραμείνει ασφαλής για όλα τα διαδοχικά ζεύγη που ισχύει $w_i + d_i \leq w_{i+1} + d_{i+1}$ και μπορούν να ανταλλάξουν θέση μεταξύ τους.
   Έστω, ότι αλλάζουν τα ζεύγη i και i+1 (οπότε $w_i + d_i \leq w_{i+1} + d_{i+1}$). Τότε, τόσο τα i+2, …, n ζεύγη που βρίσκονται πιο πάνω όσο και τα i-1, i-2, …, 1 ζεύγη που βρίσκονται πιο κάτω, παραμένουν ασφαλή διότι δεν έχει αλλάξει το βάρος που σηκώνουν.
   - Το i-οστό ζεύγος μπορεί να σηκώσει το βάρος διότι
     $d_i \geq w_{i+1} + w_{i+2} + \ldots + w_n > w_{i+2} + w_{i+3} + \ldots + w_n$
   - Το i+1-οστό ζεύγος μπορεί να σηκώσει το βάρος διότι:
     - $w_{i+1} + d_{i+1} \geq w_i + d_i$ (1)
     - $d_i \geq w_{i+1} + w_{i+2} + \ldots + w_n \rightarrow w_i + d_i \geq w_i + w_{i+1} + w_{i+2} + \ldots + w_n$ (2)
     - Από (1), (2): $w_{i+1} + d_{i+1} \geq w_i + w_{i+1} + w_{i+2} + \ldots + w_n$
     - Συνεπώς (1): $d_{i+1} \geq w_i + w_{i+2} + \ldots + w_n$

   Εν κατακλείδι, η ασφαλής λύση μπορεί να μετασχηματιστεί σε λύση που ικανοποιεί το τελευταίο κριτήριο, ότι δηλαδή $w_1 + d_1 \geq w_2 + d_2 \geq \ldots \geq w_n + d_n$.

   Πολυπλοκότητα:
   Η πολυπλοκότητα του αλγορίθμου θα είναι O(nlogn). Αυτό προκύπτει διότι απαιτείται sorting για τα ζεύγη, δηλαδή O(nlogn).

b) Ζητείται η εύρεση του υποσυνόλου των πακέτων που μπορούν να στοιβαχθούν ασφαλώς και να δίνουν το μέγιστο κέρδος.

Αλγόριθμος:
Προϋποθέσεις:

- Τα πακέτα ταξινομούνται σε φθίνουσα σειρά με βάση το τελευταίο κριτήριο.
- Θεωρείται ως P(i, j) το μέγιστο κέρδος από την στοίβαξη των πακέτων n μέχρι και το i-οστό, με συνολικό βάρος το πολύ d_j.
- Θεωρείται πως ισχύει $d_{i-1} \geq d_j$ (δηλαδή τις ασφαλείς διατάξεις).
- Κάθε φορά επιλέγεται ένα πακέτο και γίνεται προσπάθεια να τοποθετηθεί κάτω από τα υπόλοιπα n πακέτα, τα οποία ήδη βρίσκονται στην στοίβαξη. Αυτή η τοποθέτηση οδηγεί τελικά είτε στην προσθήκη του πακέτου είτε στην απόρριψή του.

Αποδεικνύεται ότι $P(i, j) = \begin{cases} 0 \\ P(i-1, dj) \\ \max\{P(i-1, \min\{di, dj - wi\}) + pi, P(i-1, dj)\} \end{cases}$

Ορθότητα:
Ο αλγόριθμος είναι ορθός επειδή σε κάθε επανάληψη ελέγχει αν το i-οστό στοιχείο μπορεί να τοποθετηθεί στην στοίβα για μέγιστο συνολικό βάρος ίσο με d_j. Παίρνει μόνο τις ασφαλείς διατάξεις καθώς πρέπει να ικανοποιείται πάντα το $d_{i-1} \geq d_j$. Συγκεκριμένα, είτε το πακέτο απορρίπτεται και δεν συμπεριλαμβάνεται στη στοίβα, είτε τοποθετείται σε μια ήδη υπάρχουσα στοίβα ως βάση της στοίβας, αν $d_i \geq d_j - w_i$ ή ως μια εκ των στοιβών που έχουν βάρος έως και d_i. Έτσι, μπορεί να συμπληρωθεί ο πίνακας όπως και στο Πρόβλημα Knapsack και να ληφθεί η σωστή απάντηση.

Πολυπλοκότητα:
Το πρόβλημα αυτό ανήκει στην κατηγορία του Δυναμικού Προγραμματισμού και λύνεται σε ψευδο-πολυωνυμικό χρόνο, δηλαδή $O(n\log n + nd_{max})$ το οποίο προκύπτει από sorting και μετά του Knapsack. Μοιάζει με το Διακριτό Πρόβλημα του Σακιδίου σε O(nB) όπου αντί για i διαθέσιμα αντικείμενα και b διαθέσιμο χώρο, υπάρχουν n πακέτα και d αντοχή.

3η Άσκηση – Τριγωνοποίηση Πολυγώνου

Αλγόριθμος:

Έστω ένα πολύγωνο με κορυφές (0, 1, 2,…, n – 1). Τότε, για κάθε κορυφή του πολυγώνου u_i θα ισχύει:

- Είτε θα συνδεθεί η κορυφή u_i με τις κορυφές u_{i+1} και u_{i-1} (σε κάθε πολύγωνο αυτό γίνεται μόνο για 2 κορυφές). Έτσι δημιουργούνται τα πολύγωνα $\{u_{i-1}, u_i, u_{i+1}\}$ και $\{u_{i+1}, u_{i+2}, \ldots, u_{i-2}, u_{i-1}\}$.
- Είτε θα συνδεθεί με την u_i κάποια κορυφή εκτός από τις u_{i+1} και u_{i-1} , έστω η κορυφή u_j. Έτσι δημιουργούνται τα πολύγωνα $\{u_i, u_{i+1}, \ldots, u_{j-1}, u_j\}$ και $\{u_j, u_{j+1}, \ldots, u_{i-1}, u_i\}$.

Για να βρεθεί λύση, θα πρέπει να διατηρείται σαν ελάχιστο κόστος τριγωνοποίησης το άθροισμα των άλλων δύο υποπολυγώνων που θα δημιουργεί και του τριγώνου. Από αρχή της βελτιστότητας είναι γνωστό ότι για να βρεθεί η βέλτιστη λύση αρκεί να εξεταστούν όλες οι πιθανές διαμερίσεις του πολυγώνου και να επιλεγεί αυτή με το ελάχιστο συνολικό μήκος πλευρών για τα τρίγωνα που προκύπτουν από την τριγωνοποίηση.

Στην πραγματικότητα, ο αλγόριθμος αυτός αποτελεί μια αναδρομική σχέση. Πιο συγκεκριμένα, για κάθε πολύγωνο, διατηρείται το ελάχιστο κόστος τριγωνοποίησης, το οποίο αποτελείται από το κόστος του τριγώνου που δημιουργήθηκε και το ελάχιστο κόστος τριγωνοποίησης των δύο υποπολυγώνων που δημιουργήθηκαν. Απαραίτητη προϋπόθεση ώστε να μπορεί να τριγωνοποιηθεί είναι να έχει τουλάχιστον τρεις κορυφές. Επίσης, χρησιμοποιείται και memorization ώστε να αποφευχθούν περιττοί υπολογισμοί. Αυτό γίνεται με χρήση ενός πίνακα με n – 1 στήλες και γραμμές και όλα τα κελιά του αρχικοποιημένα με –1 και κατά την εκτέλεση του αλγορίθμου για κάποιο πολυώνυμο, αυτή η τιμή θα αλλάζει και αναλόγως αν αυτή είναι ίση ή διάφορη του –1, θα πρέπει να γίνουν οι κατάλληλοι υπολογισμοί.

Αναδρομική Σχέση: $P(i, j) = \begin{cases} 0 \\ \min\{P(i,k) + P(k,j) + triangleCost(i,j,k)\} \end{cases}$

Και με πιο αναλυτική γραφή:

Για n = 3: $\Delta(u_i, u_j, u_k) = \left|\left|(u_i, u_j)\right|\right| + \left|\left|(u_j, u_k)\right|\right| + \left|\left|(u_k, u_i)\right|\right|$

Για n ≥ 3: $\Delta(u_1, u_2, \ldots, u_{i-1}, u_i, u_{i+1}, \ldots, u_{n-1}, u_n) =$

$$min\left\{\min_{j \neq \{u_{i-1}, u_i, u_{i+1}\}}\left[\Delta(u_i, u_{i+1}, \ldots, u_{j-1}, u_j) + \Delta(u_j, u_{j+1}, \ldots, u_{i-1}, u_i)\right],\ \Delta(u_{i-1}, u_i, u_{i+1}) + \Delta(u_{i+1}, u_{i+2}, \ldots, u_{i-2}, u_{i-1})\right\}$$

Ορθότητα:

Κάνοντας χρήση της αναδρομικής σχέσης, προκύπτει ότι το πρόβλημα διαιρείται σε μικρότερα προβλήματα και από το καθένα λαμβάνεται το μικρότερο κόστος. Επομένως, ο αλγόριθμος είναι ορθός αφού λύνει όλα τα υποπροβλήματα με βέλτιστο τρόπο.

Πολυπλοκότητα:

Η πολυπλοκότητα είναι $O(n^3)$ αφού συμπληρώνεται ένας πίνακας δυο διαστάσεων και παράλληλα δημιουργούνται όλα τα ενδεχόμενα υποπροβλήματα O(n).

4η Άσκηση – Τοποθέτηση Στεγάστρων (και Κυρτό Κάλυμμα)

Σημείωση:

Η επίλυση ακολούθησε την ίδια λογική με την 3η Άσκηση.

Αλγόριθμος:

- Δημιουργείται ένας πίνακας αποστάσεων που έχει n+1 θέσεις (η θέση 0 έχει το 0, και οι υπόλοιπες n έχουν τα σημεία x_0, …, x_n.
- Αρχίζει από το σημείο που βρίσκεται στην n+1 θέση του πίνακα ώστε να υπολογιστεί το κόστος κάλυψης των σημείων που βρίσκονται από τη θέση 0 μέχρι και τη θέση που βρίσκεται το υπό εξέταση σημείο.
- Το προηγούμενο βήμα λειτουργεί διαιρώντας το πρόβλημα σε υποπροβλήματα και παράλληλα κάθε φορά υπολογίζεται η βέλτιστη λύση για κάθε υποπρόβλημα. Ειδικότερα, αν το σημείο που ελέγχεται είναι στην θέση j τότε για κάθε ενδιάμεση θέση στο διάστημα [0, j] θα δημιουργείται το υποπρόβλημα c_i (ο δείκτης i υποδεικνύει την ενδιάμεση θέση) και θα πακετάρονται ενιαία τα σημεία που βρίσκονται από την θέση i+1 μέχρι και την θέση j.
- Σημειώνεται ότι, το κόστος ισούται με το ελάχιστο κόστος υποπροβλήματος αθροισμένο με το ενιαίο κόστος από την i+1 θέση μέχρι και την j. Οπότε, χρησιμοποιώντας τον πίνακα c που έχει μέγεθος n+1 θέσεις με αρχικοποιημένη την θέση 0 (ίση με 0) θα αποθηκεύεται στην θέση i ο λιγότερο δαπανηρός τρόπος κάλυψης των πρώτων i σημείων.

Παράδειγμα – Εφαρμογή:

Έστω ευθείας μήκους 10 (δέκα) και έστω οι ενδιάμεσοι αριθμοί [1, 3, 5, 8]. Επίσης, τοποθετείται στην αρχή ο αριθμός 0. Έτσι, δημιουργείται ένας πίνακας μήκους n όπου n είναι το πλήθος των υποδιαστημάτων συν το μηδέν. Κάνοντας χρήση της παρακάτω σχέσης, προκύπτουν:

$c[i] = \min\{c[j] + (x_i - x_{j+1})^2 + c\}$

οπότε για i = 0, c[0] = 0

για i = 1, $c[1] = \min\{5\} = 5$

για i = 2, $c[2] = \min\{9, 10\} = 9$

για i = 3, $c[3] = \min\{21, 14, 14\} = 14$

για i = 4, $c[4] = \min\{c[0] + (8-1)^2 + 5, c[1] + (8-3)^2 + 5, c[2] + (8-5)^2 + 5, c[3] + (8-8)^2 + 5\}=$

$= \min\{53, 35, 23, 19\} = 19$

Και πιο εποπτικά,

Σχόλιο: Πρακτικά, τα δεξιά κελιά "συστεγάζονται" διότι θα έχουν ήδη την καλύτερη τιμή για να στεγαστούν μιας και κάθε φορά επιλύονται υποπροβλήματα τα οποία λύνονται βέλτιστα και μετά θα λυθεί αυτό.

Ορθότητα:

Επειδή λύνεις τα υποπροβλήματα βέλτιστα, πρακτικά και το πρόβλημα θα είναι βέλτιστο, αφού παραλείπονται περιττοί υπολογισμοί καθώς μεγαλύτερα υποπροβλήματα θα έχουν σαν μέρος στις λύσεις τους το βέλτιστο κόστος κάποιου τουλάχιστον υποπροβλήματος και επομένως και το τελικό πρόβλημα θα είναι βέλτιστο.

Πολυπλοκότητα:

Η πολυπλοκότητα είναι $O(n^2)$ γιατί υπάρχει μια συνάρτηση n υπολογισμών και επίσης πραγματοποιούνται άλλες n κλήσεις.

5η Άσκηση – Καλύπτοντας ένα Δέντρο

a) Με βάση προηγούμενο σχέδιο λύσης, έχει γραφτεί ο παρακάτω αλγόριθμος.
Αλγόριθμος:

- Έστω το ελάχιστο κόστος κάλυψης, το οποίο συμβολίζεται με f(v,d,i), όπου i είναι το σύνολο μεγέθους του υποδέντρου, v είναι η ρίζα και d η απόσταση της ρίζας.
- Χρησιμοποιείται η αναδρομική σχέση:
  f(v,d,i) = min{A, B}, όπου A σημαίνει να ανήκει η κορυφή v στο K και B να μην ανήκει.
- Τότε, το κόστος του υποδέντρου με ρίζα την κορυφή v, αν αυτή ανήκει στο K, θα δίνεται ως εξής:
  A = min {max{ f(ch_l(v),1,j), f(ch_r(v),1,i–1–j) }}, για j=0,…,i–1
  Έτσι, για το A: η λύση για το δέντρο θα είναι το μέγιστο κόστος των δύο υποδέντρων για την καλύτερη μοιρασιά των i–1 στοιχείων που έχουν απομείνει από το σύνολο κάλυψης.
- Επιπλέον, το κόστος του υποδέντρου με ρίζα την κορυφή v, αν αυτή δεν ανήκει στο K, θα δίνεται ως εξής:
  B = min {max{ f(ch_l(v),d+1,j), f(ch_r(v),d+1,i–j), d }}, για j=0,…,i
  Έτσι, για το B: η λύση για το δέντρο θα είναι το μέγιστο κόστος μεταξύ της ρίζα και των λύσεων των δύο υποδέντρων για την καλύτερη μοιρασιά των i στοιχείων του συνόλου κάλυψης.
- Έπειτα, αρχικοποιείται το φύλλο v: f(v,d,i) = $\begin{cases} d, i = 0 \\ 0, i > 0 \end{cases}$
- Οπότε, ξεκινώντας από τα φύλλα, για κάθε επίπεδο υπολογίζεται το f για κάθε d και για κάθε i. Οπότε το ζητούμενο κόστος είναι f(r,0,k), όπου στην ρίζα εκτελείται μόνο ο κανόνας A.
- Για την επιστροφή του συνόλου κάλυψης, αρκεί να αποθηκευτεί και αν η καλύτερη επιλογή ήταν η κορυφή v να ανήκει στο σύνολο κάλυψης ή όχι καθώς και το καλύτερο μοίρασμα των i κορυφών στα υποδέντρα.

Πολυπλοκότητα:

Η πολυπλοκότητα θα είναι $O(n^2k^2)$

b) Αλγόριθμος:
Σημείωση: Greedy Αλγόριθμος
Έως ότου καλυφθεί ολόκληρο το δέντρο:

- Αρχίζοντας από ένα φύλλο στο χαμηλότερο επίπεδο, να ανεβαίνει στο δέντρο μέχρι την κορυφή-πρόγονο σε απόσταση z.
- Έπειτα, να προσθέσει την κορυφή αυτή στο σύνολο K.
- Τέλος, να διαγράψει το υποδέντρο με ρίζα την κορυφή αυτή.

Πολυπλοκότητα:

Η πολυπλοκότητα θα είναι O(n) διότι, διέρχεται από κάθε κορυφή σταθερό πλήθος φορών.

Ορθότητα:

Κάνοντας χρήση της Αρχής Βελτιστότητας Υπολύσεων, αν έχει βρεθεί μια βέλτιστη λύση K* για όλο το δέντρο T και αν οριστεί το T' ως το δέντρο που προκύπτει από το T αν αφαιρεθεί το υποδέντρο που βρίσκεται κάτω από μια κορυφή που ανήκει στο σύνολο K, τότε τα στοιχεία που απομένουν στο K καλύπτουν βέλτιστα το T'.

Επίσης, κάνοντας χρήση της Άπληστης Επιλογής, προκύπτει ότι αν v η κορυφή που διαλέγει πρώτη ο αλγόριθμος, τότε θα δειχθεί ότι υπάρχει βέλτιστη λύση που περιέχει την v:

Έστω ότι η K* είναι βέλτιστη λύση που δεν περιέχει την v. Τότε, αναγκαστικά θα περιέχει κάποια κορυφή-απόγονο της v. Έστω v' αυτή η κορυφή. Τότε, μπορεί να κατασκευαστεί λύση με ίσο (ή μικρότερο) πλήθος κορυφών κάλυψης απλά αντικαθιστώντας την v' με την v στο σύνολο κάλυψης.

Αν θεωρηθεί ως "μαύρο κουτί" ο αλγόριθμος A(z) του 2^{ου} ερωτήματος, προκύπτει η λύση και για το 1^{ο}:

Με δυαδική αναζήτηση ως προς z για την εύρεση του βέλτιστου κόστους z* για το οποίο ο αλγόριθμος του 2^{ου} ερωτήματος επιστρέφει σύνολο μεγέθους το πολύ k. Τυπικά, αναζητείται το z* τ.ω.: για κάθε $z \geq z^*$, $A(z) \leq k$ και για κάθε $z < z^*$, $A(z) > k$.

Αυτό είναι εφικτό γιατί το μέγεθος του βέλτιστου συνόλου είναι φθίνουσα συνάρτηση του z.

Πολυπλοκότητα:

Η πολυπλοκότητα θα είναι O(nlogn) διότι θα είναι O(n) για τον αλγόριθμο που τρέχει σε κάθε ένα από τα O(logn) βήματα της binary search.