Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Αλγόριθμοι και Πολυπλοκότητα
Διδάσκοντες: Δημήτρης Φωτάκης, Δώρα Σούλιου, Θανάσης Λιανέας
2η Σειρά Γραπτών Ασκήσεων - Ημ/νία Παράδοσης 3/12/2020

ΜΠΕΚΡΗΣ ΔΗΜΗΤΡΗΣ

ΑΜ 03117116

<u>1η Άσκηση:</u>

Για την τοποθέτηση του ελάχιστου αριθμού δίσκων, μελετήθηκε κι αποδείχθηκε ένας “Greedy” αλγόριθμος.

Αρχικά, προβάλουμε όλα τα σημεία στην ευθεία και μετρούμε τις αποστάσεις από την αρχή των αξόνων (0,0).

Έστω, **Xi** οι μετρήσεις κάθε προβολής. Με αλγόριθμο ταξινόμησης ταξινομούμε τα **Xi** σε χρόνο **O(nlogn)**.

<u>Βασική Ιδέα:</u>

Στόχος του αλγορίθμου είναι η τοποθέτηση των κεντρών των δίσκων όσο το δυνατό πιο δεξιά γίνεται μη αφήνοντας πιθανά αριστερά σημεία ακάλυπτα(σημεία αριστερά του άξονα **y**). Ισχυριζόματε ότι αυτή η σκέψη οδηγεί στη βέλτιστη λύση(το οποίο θα αποδειχθεί παρακάτω).

<u>Αλγόριθμος:</u>

1.1 Ξεκινάμε από το αριστερότερο σημείο και γίνεται υποψήφιο για την τοποθετησή του στην περιφέρεια του εκάστοτε δίσκου **D(κέντρο O, ακτίνα r)**.
1.2 Στη συνέχεια, για κάθε επόμενο στοιχείο Xi με τις ιδιότητες:
   - **a.a Xi > Xo (1)**
   - a.b **|Xi – Xo| < r (2)**, δηλαδή το **Xi** να ανήκει στα 2ο και 3ο τεταρτημόριο
1.3 **Η άπληστη επιλογή αποτελεί, το σημείο με την μέγιστη απόσταση από το κέντρο O.** Πράττουμε την άπληστη επιλογή, ως εξής:
   - **Xi** ανήκει στο **D**, όπου **D** η εξίσωση του δίσκου
     - i.a ΝΑΙ -> continue
     - i.b ΟΧΙ -> ανανεώνουμε το νέο υποψήφιο σημείο με μια σύγκριση**,** συνεχίζουμε
   - Οι επιλογές αυτές συνεχίζονται μέχρι να πάψει να ισχύει η συνθήκη **(2).**
1.4 Το άπληστο κριτήριο, λοιπόν, αποτελεί το σημείο, που θα βρεθεί στα 1ο και 3ο τεταρτημοριο (αφού μπορούμε να πάρουμε την απόλυτη τιμή της αρνητικής απόστασης) , με την μεγαλύτερη απόσταση από το **κέντρο O**, έχοντας ξεκινήσει την έρευνα από το εκάστοτε αριστερότερο εναπομείναν ακάλυπτο στοιχείο μετά το πέρας της προηγούμενης τοποθέτησης.

1.5 Μετά την τοποθέτηση του δίσκου, συνεχίζουμε το **iteration** από αριστερά προς τα δεξία, ψάχνοντας για το πρώτο στοιχείο που δεν καλύπτεται.
1.6 Επαναλαμβάνουμε τα παραπάμω βήματα μέχρι να μην υπάρχουν άλλα σημεία.

Σχολιασμός:

- Το κριτήριο αυτό, μας εγγυάται ότι, δεν αφήνει ακάλυπτα αριστερότερα σημεία μετά την τοποθέτηση του δίσκου.
- Επιχείρημα αποτελέι η παρακάτω ανάλυση:
  - Έστω ένα σημείο με προβολή **Xo**. Το **Xo** αποτελεί το αριστερότερο εναπομείναν ακάλυπτο στοιχείο της προηγούμενης τοποθέτησης.
  - Για κάθε στοιχείο **j** με **Xj, dj** με **|Xj - Xo| < r**, υπάρχουν 2 ενδεχόμενα:
    - **dj < do,** οπότε σίγουρα καλύπτεται κι αν τοποθετηθεί αυτό το σημείο στην περιφέρεια τότε τουλάχιστον ένα σημείο (το **Xo**) θα μείνει ακάλυπτο, αφού ο δίσκος θα μεταφερθεί δεξιότερα για να ανήκει στην περιφέρεια το **Xj** στοιχείο **(1)**
    - **dj > do,** καλύπτεται **(2)**
    - **dj > do,** δεν καλύπτεται **(3)**
  - Επομένως, με το παράπανω κριτήριο σκοπός μας είναι η ανίχνευση των προαναφερθέντων στοιχείων και η μελέτη τους.
  - Για τα στοιχεία με την ιδιότητα **(3)** , είμαστε σίγουροι ότι αν λάβουμε από αυτά το σημείο, το οποίο σχηματίζει τρίγωνο **Δ(O, j, Xj)** με την μεγαλύτερη υποτείνουσα, δεν θα μείνουν προήγουμενα ακάλυπτα.
  - Τα στοχεία με την ιδιότητα **(3)**, για να ανήκουν στην περιφέρεια, απαιτείται η μετατοπιση του **κέντρου O** αριστερά.
  - Επομένως, λαμβάνοντας ως σημείο περιφέρειας αυτό, το οποίο απέχει περισσότερο από το κέντρο εξασφαλίζουμε την μεγαλύτερη μετακίνηση προς τ΄αριστερά άρα και την κάλυψη όλων των αριστερών σημείων.

Χρονική Πολυπλόκοτητα:

Ξεκινώντας από το πρώτο σημείο, εκτελούμε τον αλγόριθμο. Η εύρεση του σημείου με την μεγαλύτερη απόσταση από το κέντρο γίνεται σε γραμμικό χρόνο ως προς το πλήθος των σημείων που ανήκουν στα 2ο, 3ο τεταρτημότιο. Μετά την τοποθέτηση του δίσκου, με O(1) για κάθε σημείο, που ελέγχουμε βρίσκουμε το σημείο που δεν καλύπτεται από τον τοποθετημένο δίσκο. Επαναλαμβανουμε την διαδικασία, μέχρι το τέλος. Ο αλγόριθμος εκτελείται σε ένα πέρασμα της εισόδου με ελέγχους O(1). Επομένως, ο χρόνος του αλγορίθμου είναι **O(n).** Η συνολική πολυπλοκότητα **O(nlogn).**

<u>Ορθότητα:</u>

Η ορθότητα αποδείχθηκε με επαγωγή:

Μετά την ταξινόμηση θα είναι:

S(A) = 1 + S(A(xi)) , [xi,di] , όπου A(xi) = {Xj: Xj δεν ανήκουν στο Dxi / Dxi : δίσκος με Xi στην περιφέρεια}

Έστω βέλτιστος αλγόριθμος:

S*(A) = min(Xj) {1 + S*(A(xj)} , 0 < Xj < Xf

A = { }, τότε ισχύει S(A) = S*(A) (1)

Έστω για κάθε $A' \subset A$ ισχύει S(A’) = S*(A’) (2)

Τότε S(A) = 1 + S*(A(xi)) (3)

Θ.δ.ο ισχύει S(A) = S*(A) (4)

Αν $X_j \neq X_i \rightarrow A_{(x_i)} \subset A_{(x_j)} \rightarrow S\left(A_{(x_i)}\right) \leqslant S\left(A_{(x_j)}\right)$ και $S*\left(A_{(x_i)}\right) \leqslant S*\left(A_{(x_j)}\right)$

Επομένως, η (3) γίνεται :

$S(A) \leqslant 1 + S*\left(A_{(x_i)}\right) \leqslant 1 + S*\left(A_{(x_j)}\right) i = j$

Άρα, αποδείχθηκε η (4). Το επιχείρημα ανταλλαγής εδώ λοιπόν αποτελεί η **μονοτονία** και περιγρέφεται με το γεγονός ότι, αν ανανταλλάξουμε το δικό μας σημείο περιφέρειας με το σημείο του βέλτιστου, τότε ο αλγοριθμός μας δεν βελτιώνεται λόγω, των μεγαλύτερων κατά πλήθος των υποσυνόλων. Επομένως, ο αλγόριθμος μας δίνει βέλτιστη λύση.

Άσκηση 2:

2.α.1)

Το κριτήριο καταρρέει, καθώς υπάρχει αντιπαράδειγμα για το οποίο γνωρίζουμε ότι, υπάρχει ασφαλής στίβαξη.

Δύο πακέτα με τιμές:

	W	D	P
A	3	1	5
B	**2**	**4**	**5**

Ενώ υπάρχει η ασφαλής στίβαξη ΒΑ, το παραπάνω κριτήριο μας δίνει μια λανθασμένη απάντηση ΑΒ.

2.α.2)

Όμοια υπάρχει αντιπαράδειγμα και για αυτό το κριτήριο.

	W	D	P
A	4	2	5
B	**1**	**3**	**5**

Ασφαλής στίβαξη αποτελεί η ακολουθία AB, ενώ το κριτήριο δίνει τη λανθασμένη απάντηση BA.

2α3)

Γνωρίζοντας την ύπαρξη ασφαλούς στίβαξης θα αποδείξουμε, ότι το συγκεκριμένο κριτήριο δίνει πάντα μια ασφαλή στίβαξη.

Έστω βέλτιστος αλγόριθμος με την ακολουθία $\pi_1*, \pi_2*, \pi_3*, \ldots \pi_n*$. Και η ακολουθία του άπληστου $\alpha_1, \alpha_2, \ldots \alpha_n$.

Έστω α_k το πρώτο στοιχείο, που διαφέρουν οι δυο αλγόριθμοι στίβαξης. Τότε θα ισχύει

$$w_k + d_k < w_{(k+1)} + d_{(k+1)} (1)$$

Έστω ότι, ανταλάσσουμε τα δύο αυτά στοιχεία, τότε για ασφαλή στίβαξη θα πρέπει να ισχύουν:

$$d_k \geq w_{(k+2)} + w_{(k+3)} + w_{(k+4)} \ldots . w_n (2)$$

$$d_{(k+1)} \geq w_k + w_{(k+2)} + w_{(k+3)} + w_{(k+4)} \ldots . w_n (3)$$

Εξαρχής, αφού είχαμε ασφαλή στίβαξη θα ισχύει:

$$d_k \geq w_{(k+1)} + w_{(k+2)} + w_{(k+3)} + w_{(k+4)} \ldots . w_n$$

Άρα η (2) ισχύει.

Από (1) θα έχουμε:

$$d_{(k+1)} + w_{(k+1)} > d_k + w_k \geq w_k + w_{(k+1)} + w_{(k+2)} \ldots + w_n \rightarrow$$

$$d_{(k+1)} > d_k + w_k \geq w_k + w_{(k+2)} \ldots + w_n$$

Επομένως, καταλήξαμε πάλι σε αφαλή στίβαξη. Εκτελώντας την ίδια διαδικασία σε κάθε στοιχείο, διαφέρουν οι δυο αλγόριθμοι, θα καταλήγουμε σε ασφαλή στίβαξη. Συνεπώς, ο αλγόριθμος είναι βέλτιστος.

2.β)

Το πρόβλημα θυμίζει knapsack, με τη διαφορά ότι, στην προκειμένη περίπτωση παίζει ρόλο η σειρά τοποθέτησης των αντικεμένων. Προς αποφυγή αυτού του περιορισμού και του εκθετικού χρόνου αναζήτησης, θα χρησιμοποιήσουμε το Greedy κριτήριο που αποδείχθηκε στο 2.α.3, το οποίο μας δίνει για κάθε σύνολο n στοιχείων, πάντα μια ασφαλής στίβαξη με την προϋπόθεση ότι υπάρχει κάποια.

Με το παρπάνω κριτήριο εξασφαλίζουμε μια σχετική σειρά τοποθέτησης των αντικειμένων, με την εγγύηση ότι, αν υπάρχει κάποια ασφαλής στίβαξη αυτών ή κάποιου υποσυνόλου τους, θα ακολουθεί το greedy κριτήριο.

Στη συνέχεια με DP ερευνούμε κάθε πιθανη στίβαξη, που μεγιστοποιεί το συνολικό κέρδος για το εκάστοτε υποσύνολο. Ως space state χρησιμοποιείται ο συμβολισμός (i,w) , όπου i το εκάστοτε στοιχείο προς μελέτη και w, το βάρος που υπάρχει από πάνω εκείνη τη στιγμή. Όπως, είναι φανερό επιλέχθηκε η κατασκευή της στίβας από την κορυφή προς τη βάση, για την αποφυγή του γραμμικού χρόνου, τον οποίο θα χρεωνόμασταν αν χτίζαμε την στίβα από τη βάση προς την κορυφή, ελέγχοντας για την τοποθέτηση του i-οστου στοιχείου τις αντοχές των i-1 στοιχείων.

Το w εκτείνεται στο διάστημα [0,dmax], καθώς αν w > dmax, μη ασφαλής στίβαξη.

Η μαθηματική σχέση που περιγράφει τον παραπάνω αλγοριθμο είναι η εξής:

$$P[i,w]=\begin{cases} min\{P[i-1,w+w_i]+p_i, P[i-1,w]\}, i>1, (d_i \geq w \; και \; w+w_i \leq d_{max}) \\ P[i-1,w], i>1, (d_i < w \; ή \; w+w_i > dmax) \\ p_1, i=1 \; και \; d_1 \geq w \\ 0, (i=1 \; και \; d_1 < w) \; ή \; 0=0 \end{cases}$$

Δίνεται ο παρακάτω ψευδοκώδικας που περιγράφει την παραπάνω αναδρομική σχέση.

```
*initialize*
for w ← 0 to d_max :
    P[0,w] ← 0
    if d_1 >= w do:
        P[1,w] ← p_1
    else:
        P[1,w] ← 0
*regression*
for i ← 2 to n:
    for w ← o to d_max :
        P[i,w] ← P[i-1, w]
        if d_i >= w and w+ w_i <= d_max do:
            q ← P[i-1, w+ w_i ] + p_i
            if q > P[i,w] do:
                P[i,w] ← q
return P[n,0]
```

Ορθότητα:

- Με DP ερευνούμε κάθε πιθανή (ασφαλής) στίβαξη, η οποία δίνει το μέγιστο κέρδος για κάθε υποπρόβημα στοιχείων. Επομένως, λύνοντας τα υποπροβήματα, με υλοποίηση bottom up, βρίσκουμε για κάθε υποσύνολο την μέγιστη κερδοφόρα ασφαλή στίβαξη και την αποθηκεύουμε στην αντίστοιχη θέση του πίνακα. Επικαλούμενοι την αρχή της βελτιστότητας, το τελικό βέλτιστο αποτέλεσμα θα βρίσκεται στη θέση P[n,0] του πίνακα, καθώς έχουν λυθεί προηγουμμένως τα υποπροβλήματα.

<u>Πολυπλοκότητα:</u>

Αρχικά χρεωνόμαστε O(nlogn) για την ταξινόμηση. Στη συνέχεια η χρονική πολυπλοκότητα, του κυρίους αλγορίθμου, είναι παρόμοια με του knapsack. Έχουμε πολυωνυμική πολυπλοκότητα ως προς την είσοδο $O(n*d_{max})$ και εκθετική πολυπλοκότητα ως προς το $\log_2(d_{max})$. Επομένως, αλγόριθμος **ψευδο-πολυωνυμικού** χρόνου.

Για τη χωρική πολυπλοκότητα ισχύουν τα ίδια $O(n*d_{max})$

Ακολουθεί ένα σχετικό παράδειγμα της υλοποίησης του παραπάνω αλγορίθμου:

Έστω, n στοιχεία:

	W	D	P	D+W
Α	3	2	2	5
Β	4	2	4	6
Γ	1	2	1	3
Δ	3	5	2	8
Ε	**2**	**2**	**1**	**4**

Ταξινομώντας ως προς τα αθροισματα:

$Δ \geq B \geq A \geq E \geq Γ$

Ο πίνακας θα γεμίσει ως εξής:

	i/w	0	1	2	3	4	5
	0	0	0	0	0	0	0
Δ	1	2	2	2	2	2	2
Β	2	6	6	2	2	2	2
Α	3	6	6	4	2	2	2
Ε	4	6	6	4	2	2	2
Γ	**5**	**7**	**6**	**4**	**2**	**2**	**2**

Το αποτέλεσμα βρίσκεται στη θέση P[5,0]. Για την εύρεση της στίβαξης θα ακολουθήσουμε την εξής διαδικασία(backtracking):

- Αρχίζουμε από τη θέση P[n,0] και κινούμαστε προς τα πάνω μέχρι να βρούμε την πρώτη ασυνέχεια.
- Σταματάμε, κοίταμε το index της προηγούμενης γραμμής(πριν αλλάξει η τιμή) και βάζουμε το αντίστοιχο στοιχείο στην στίβα ξεκινώντας από την κορυφή.
- Έστω λοιπόν ότι, βρήκαμε ότι, προστέθηκε το στοιχείο j. Για την συνέχεια του backtracking, πηγαίνουμε στη θέση $P[i-1, w+w_j]$
- Επαναλμβάνουμε τα παραπάνω βήματα μέχρι να φτάσουμε στο i=0.

Για το συγκεκριμένο παράδειγμα, θα γίνουν τα επακόλουθα βήματα:

- Ξεκινάμε από το i = 5
- Ανιχνεύουμε ασυνέχεια, οπότε το Γ στοιχείο στην κορυφή
- Πηγαίνουμε στη θέση P[4,1]
- Κινούμαστε προς τα πάνω μέχρι την πρώτη ασυνέχεια, η οποία βρίσκεται στο i=2, τότε το Β κάτω από το Γ
- Πηγαίνουμε στη θέση P[1,5]
- Ανιχνεύουμε ασυνέχεια, άρα το Δ κάτω από το Β
- Φτάσαμε στο i=0, ΤΕΛΟΣ

Η ζητούμενη στίβαξη είναι ΔΒΓ, με μέγιστο κέρδος 7.

<u>Άσκηση 3:</u>

Το πρόβλημα μοιάζει με το matrix multiplication. Η λύση του προβήματος ανάγεται στην εύρεση της καλύτερης διαμέρισης του ολικού πολυγώνου, η οποία θα δώσει τη βέλτιστη λύση αναδρομικά.

Έστω space state (i,j), όπου i η αρχική και j η τελική κορυφή του εκάστοτε πολύγωνου, κινούμενοι σύμφωνα με την αντίθετη φορά των δεικτών του ρολογιού. Ορίζουμε m[i,j] μια αναδρομική συνάρτηση, που επιστρέφει κάθε φορά το ελάχιστο δυνατό μήκος της αντίστοιχης τριγωνοποίησης, την οποία υφίσταται το πολύγωνο.

Ο αναδρομικός τύπος που την περιγράφει είναι ο εξής:

- $m[i,j] = \left\{ \begin{matrix} min_{(i<k<j)} m[i,k] + m[k,j] + \Delta(u_i u_k u_j), |(i-j)| \geq 2 \\ 0, |(i-j)| \leq 1 \end{matrix} \right\}$

Ακολουθεί ο παρακάτω ψευδοκώδικας που περιγράφει την παραπάνω αναδρομική σχέση. Η υλοποίηση έγινε με τη μέθοδο bottom-up.

```
triangular(P):
for i ← 1 to n-1:      *initialize*
    for j ←1 to n-1:
        m[i,j] ← 0
m[n,m] ← 0
*regression*
for p ←2 to n-1:
    for I ← 1 to n-p:
        j ← i+p ; m[i,j] ← ∞ *initialize*
        for k ← i+1 to j-1:
            q ← m[i,k] + m[k,j] + || u_i u_k || + || u_k u_j || + || u_j u_i ||
            if q < m[i,j] do:
                m[i,j] ← q
return m[1,n]
```

<u>Παρατήρηση:</u>

Η υλοποίηση απαιτεί μόνο το άνω μέρος του πίνακα(άνω τριγωνικός), καθώς τα υπόλοιπα υποπροβλήματα δεν έχει νόημα να εξεταστούν, άρα αποφεύγεται η χρήση άσκοπου χώρου.

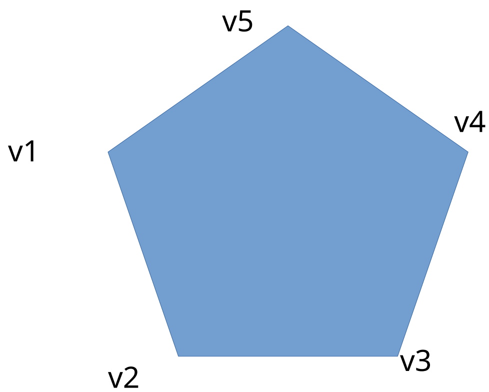
<u>Πολυπλοκότητα:</u>

Ο αλγόριθμος έχει χρονική πολυπλοκότητα **O(n^3)**, η οποία εύκολα υπολογίζεται ως το γινόμενο του space state και την πολυπλοκότητας του κάθε βήματος(εδώ είνα γραμμική). Τέλος, η χωρική πολυπλοκότητα **O(n^2)**.

<u>Ορθότητα:</u>

Ο αλγόριθμος εγγυάται την βέλτιστη λύση, καθώς επικαλείται την αρχή της βελτιστότητας. Με την μέθοδο bottom up βρίσκουμε το ελάχιστο μήκος για τα υποπρόβληματα, εξετάζοντας κάθε πιθανή διαμέριση και υπολογίζοντας για την κάθε μία το μήκος της εκάστοτε τριγωνοποίησης. Συγκρίνοντας τις τιμές σε γραμμικό χρόνο αποθηκεύουμε την μικρότερη τιμή στον πίνακα καταστάσεων για να αποφευχθεί ο ίδιος υπολογισμός. Πρακτικά, είναι ο κλασικός αλγόριθμος, που υπολογίζει αναδρομικά όλες τις πιθανές τριγωνοποιήσεις, **αλλά** χρησιμοποιούμε τον πίνακα για την αποφυγή του εκθετικού χρόνου.

Δίνεται παρακάτω ένα παράδειγμα του παραπάνω αλγορίθμου:



Για το παράδειγμα θα υποθέσουμε τις παρακάτω πλευρές:

	v1	v2	v3	v4	v5
V1	0	2	5	5	2
v2	x	0	3	4	3
v3	x	x	0	2	4
v4	x	x	x	0	2
v5	**x**	**x**	**x**	**x**	**0**

Ο πίνακας του αλγορίθμου θα γεμίσει ως εξής:

i/j	1	2	3	4	5
1	0	0	10	20	25
2	x	0	0	9	18
3	x	x	0	0	8
4	x	x	x	0	0
5	**x**	**x**	**x**	**x**	**0**

Παρατηρούμε ότι στην θέση m[1,5] του πίνακα βρίσκεται το σωστό αποτέλεσμα.

Άσκηση 4:

α) Το πρόβλημα είναι ένα ζήτημα ελαχιστοπόιησης. Θα χρησιμοποιήσουμε DP για την λύση του προβλήματος.

Αρχικά ορίζουμε το space state ως την μεταβλητή i, η οποία συμβολίζει τα πρώτα i σημεία του πεζοδρόμου.

Ως c[i] θα συμβολίζουμε το ελάχιστο κόστος, που είναι δυνατόν, για την κάλυψη των i πρώτων σημείων του πεζοδρόμου.

Η αναδρομική σχέση που λύνει το πρόβλημα είναι η εξής:

$$c[i]=\left\{\begin{matrix} min_{(0\leq j<i)} c[j]+\left(X_{(j+1)}-X_i\right)^2+C, i>1 \\ C, i=1 \\ 0, i=0 \end{matrix}\right\}$$

Πολυπλοκότητα:

Η χρονικλη πολυπλοκότητα του αλγορίθμου είναι ασφαλώς τετραγωνική (**O(n^2)**), καθώς προκύπτει εύκολα από το γινόμενο του state space επί του γραμμικού ελέγχου για κάθε στοιχείο. Η χωρική πολυπλοκότητα είναι γραμμική ως προς την είσοδο, δηλαδή **O(n)**.

Ορθότητα:

Η ορθότητα του αλγορίθμου εξασφαλίζεται, αν κανείς σκεφτεί το πρόβλημα σαν ένα σύνολο υποπροβλημάτων, που αντιμετωπίζει κάθε φορά. Μπορούμε να αρχίσουμε να σκεφτόμαστε το πρόβλημα ως εξής:

- Αχικά έχουμε ένα σήμειο προς κάλυψη, επομένως κοστίζει C
- Προστίθεται 2ο. Έχουμε τις εξής επιλογές:
    - Ξεχωριστό σκέπαστρο
    - Ένα ενιαίο.
    - Συγκρίνουμε και πάιρνουμε το καλύτερο αποτέλεσμα και το αποθηκεύουμε στην κατάλληλη θέση του πίνακα.
- Προστίθεται 3ο. Τρεις επιλογές:
    - Μονό σκέπαστρο
    - Διπλό σκέπαστρο
    - Τριπλό σκέπαστρο

Αυτή η διαδικασία συνεχίζεται για το i-οστο σημείο του πεζόδρομου.

Για το i-οστο σημείο για κάθε πιθανή διαμέριση, προστίθεται ο όρος $(X_{(j+1)} - X_i)^2 + C$, ο οποίος ανάγεται σε δύο πιθανές επιλογές:

- Μονό σκέπαστρο
- Ένωση με κάποια από τα **i- (j+1)** προηγούμενα σημεία

Θα μπορούσε κανείς να παρατηρήσει ότι, από το **(j+1)**-οστο σημείο έως το **i**-οστο σημείο, δεν ελέγχεται κανένας άλλος πιθανός συνδυασμός παρα μονο η συνένωση των στεγάστρων. Αυτό γίνεται, διότι κάθε άλλος πιθανός συνδυασμός μπορεί να αναχθεί σε προηγούμενο υποπρόβλημα.

Για παράδειγμα ο συνδυασμός $c[j] + (X_{(j+1)} - X_i)^2 + C$, μπορεί να ανλυθεί σε δύο επιμέρους στέγαστρα για τα σημεία από το **j+1** μέχρι το **i ,** ως

$$c[j] + (X_{(j+1)} - X_k)^2 + C + (X_{(k+1)} - X_i)^2 + C$$

το οπίο όμως ανάγεται στο υποπρόβλημα

$$c[k] + (X_{(k+1)} - X_i)^2 + C$$

το οποίο έχει υπολογιστεί και συμπεριεληφθεί στη λυση.

Επομένως, με μια απλή υλοποίηση bottom up καταληγουμε στην ορθή λύση στη θέση του πίνακα **c[n]**.

Ακολουθεί ψευδοκώδικας, που περιγράφει τον παράπανω αλγόριθμο:

```
cover (spots[]):
*initialize*
c[0] ← 0
c[1] ← 1
for I ← 2 to n-1:
    c[i] ← ∞
    for j ← 0 to i-1:
        q ← c[j] + (X_(j+1) − X_i)^2 + C
        if q < c[i] do:
            c[i] ← q
return c[n]
```

Ένα σχετικό παράδειγμα του παραπάνου αλγορίθμου είναι το παρακάτω:

X_1	X_2	X_3	X_4	**C**
2	**3**	**5**	**6**	**12**

Ο πίνακας **c** θα είναι:

i	**0**	**1**	**2**	**3**	**4**
c[i]	**0**	**12**	**13**	**22**	**26**

Το αποτέλσμα του αλγορίθμου βρίσκεται στην τελευταία θέση του πίνακα.

Άσκηση 5:

5α) Για το πωτο ερώτημα θα χρησιμοποιήσουμε DP.

Έστω f(v,d,i) το ελάχιστο κόστος κάλυψης από σύνολο μεγέθους "i" του υποδέντρου με ρίζα "v", η οποία έχει απόσταση "d". Προκύτπει η παρακάστω μαθηματική αναδρομική σχέση;

$f(v,d,i)=min\{A,B\}$ όπου A: να ανήκει η κορυφή "v" στο K,

B : να μην ανήνει η κορυφή "v" στο K.

Επίσης, το κόστος του υποδέντρου με ρίζα την κορυφή "v" αν αυτή ανήκει στο K, δίνεται ως εξλής:

$$A=min_{(j=0,\ldots i-1)}\{max\{f(ch_l(v),1,j),f(ch_r(v),1,i-1-j)\}\}$$

Για το A ισχύει: Η λύση για το δέντρο είναι το μέγιστο κόστος των δύο υποδέντρων(όπου οι ρίζες τους απέχουν 1 από το σύνολο) για την καλύτερη διαμέριση των i-1 εναπομηνάντων στοιχείων του συνόλου κάλυψης.

Για το B το κόστος του υποδέντρου με ρίζα την κορυφή "v" αν αυτή ΔΕΝ ανήκει στο K:

$$B=min_{(j=0,\ldots,I)}\{max\{f(ch_l(v),d+1,j),f(ch_r(v),d+1,i-j),d\}\}$$

Για το B θα ισχύει ότι, η λύση για το δέντρο είναι το μέγιστο κόστος μεταξύ της ρίζας(δηλαδή d) και των λύσεων των δύο υποδέντρων(όπου οι ρίζες τους απέχουν d+1 από το σύνολο), για την καλύτερη διαμέριση των i στοιχείων του συνόλου κάλυψης.

Συνοψίζοντας:

$$f(v,d,i)=min\{A,B\}$$
$$A=min_{(j=0,\ldots i-1)}\{max\{f(ch_l(v),1,j),f(ch_r(v),1,i-1-j)\}\}$$
$$B=min_{(j=0,\ldots,I)}\{max\{f(ch_l(v),d+1,j),f(ch_r(v),d+1,i-j),d\}\}$$

Αρχικοποίηση: Για κάθε φύλλο θα έχουμε: $f(v,d,i) = \begin{matrix} d, i=0 \\ 0, i>0 \end{matrix}$

Αλγόριθμος:

- Ξεκινώντας από τα φύλλα, αρχικοποιούμε σύμφωνα με την παραπάνω σχέση
- Για κάθε επίπεδο υπολογίζουμε το f, για κάθε d in {0,1,...n-1} και για κάθε i in {0,1,...k}.
- Το ζητούμενο κόστος είναι f(r,0,k), όπου στη ρίζα εκτελείται μόνο ο κανόνας A

Backtracking:

Για να επιστρέψουμε το σύνολο κάλυψης αρκεί να αποθηκεύουμε επιπλεόν, σε ένα δεύτερο πίνακα g(v,d,i) 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:

Ο αλγοριθμός μας, ακολούθει τους εξής βρόγχους επανάληψης:

$$\begin{matrix} \text{Για κάθε επίπεδό του δέντρου} \\ \text{Για κάθε κορυφή “}v\text{” στο επίπεδο} \end{matrix} \rightarrow \Theta(n)$$
$$\text{Για κάθε } d \in 0,1,\ldots,k \rightarrow \Theta(n)$$
$$\text{Για κάθε } I \in 0,1,..,k \rightarrow \Theta(k)$$
$$\text{Υπόλογισε } f(v,d,i) \rightarrow \Theta(n)$$

Άρα εκτελείται σε $O(n^2 k^2)$.

5β) Παρατίθεται παρακάτω ο Greedy αλγόριθμος που θα ακολουθήσουμε για την επίλυση.

Μέχρι να καλύψουμε το δέντρο:

- Ξεκινώντας από ένα φύλλο στο χαμηλότερο επίπεδο ανεβαίνουμε στο δέντρο μέχρι τηνκορυφή-πρόγονο σε απόσταση z.
- Προσθέτουμε την κορυφή αυτή στο σύνολο K
- Διαγράφουμε το υπόδεντρο με ρίζα την κορυφή αυτή.

Πολυπλοκότητα:

Διασχίζουμε κάθε κορυφή σταθερό πλος φορών, επομένως έχουμε χρονική πολυπλοκότητα O(n).

Ορθότητα:

Η απόδειξη θα γίνει με τη χρήση της μαθηματικής επαγωγής. Αρχικά, λόγω της αρχής της βελτιστότητας, αν έχουμε μια βέλτιστη λύση K* για όλο το δέντρο T και ορίσουμε το T’ ως το δέντρο, που προκύπτει από το T, αν αφαιρέσουμε το υπόδεντρο, που βρίσκεται κάτω από μια κορυφή, που ανήκει στο σύνολο K, τότε τα εναπομέναντα στοιχεία του K καλύπτουν βέλτιστα το T’.

Έστω “v”η κορυφή που διαλέγεται πρώτη από τον άπληστο. Θα δείξω ότι υπάρχει βέλτιστη λύση που περιέχει τη “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, ,μπορούμε να κατασκευάσουμε λύση με ίσο(ή μικρότερο) πλήθος κορυφών κάλυψης απλά αντικαθιστώντας την “ v’ “ με την “v” στο σύνολο κάλυψης.

Βελτίωση αλγορίθμου (α)ερωτήματος:

Κάνοντας χρήση του αλγορίθμου A(z) από το (β) ερώτημα, ακολουθούμε την εξής βελτιστοποίηση:

- Κάνουμε δυαδική αναζήτηση ως προς z για να βρούμε το βέλτιστο κόστος z*, για το οποίο ο αλγόριθμος του δευτέρου ερωτήματος επιστρέφει σύνολο μεγέθους το πολύ k.
- Ουσιαστικά ψάχνουμε το z* τ.ω:
  - $Για\, κάθε\, z \geq z* A(z) \leq k$
  - $Για\, κάθε\, z < z* A(z) > k$
- Αυτό είναι εφικτό, διότι το μέγεθος του βέλτιστου συνόλου είναι φθίνουσα συνάρτηση του z.

Πολυπλοκότητα:

Καταλήγουμε σε πολυπλοκότητα O(nlogn), για το λόγο ότι, έχουμε O(n) για τον αλγόριθμο που τρέχει σε καθένα από τα O(logn)βήματα της δυαδικής αναζήτησης.