ΕΚΠΑ, Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Αλγόριθμοι και Πολυπλοκότητα, Τμήμα Αρτιων
Εαρινό Εξάμηνο 2019-2020, Εργασία 3
Ιωάννης Ροβιθάκης, sdi1800164

1) Για συμβολοσειρές **X=(x1, x2, … xn)** και **Y=(y1, y2, … , ym)**, με μήκη n και m**:**
OPT(i,j) = το πλήθος χαρακτήρων συνεχόμενης κοινής υπακολουθίας μέχρι τα i,j

$$\mathrm{OPT(i,j)} = \begin{cases} \mathrm{OPT(\ i\text{-}1,\ j\text{-}1) + 1} \ \text{:όταν } \mathrm{X(i)=Y(i)} \\ 0 \ \text{:όταν } \mathrm{X(i)!=Y(i)} \end{cases}$$

Πιο αναλυτικά:
> Αν οι i, j **είναι ίδιοι**, βρήκαμε ταίριασμα οπότε προσθέτουμε 1 στο πλήθος ταιριασμάτων μέχρι τους προηγούμενους χαρακτήρες i-1, j-1 (**OPT(i-1,j-1)**)
> Αν οι i, j **δεν ταιριάζουν**, απλά βάζουμε 0, αφού θέλουμε υπο-ακολουθίες **συνεχόμενων** χαρακτήρων.

Στην πράξη γεμίζουμε ένα πίνακα **OPT** με διαστάσεις **(n+1) x (m+1),** με τους χαρακτήρες της κάθε συμβολοσειράς να αντιστοιχούν με την σειρα, στην ανάλογη γραμμή ή στήλη, και την μηδενική σειρά και στήλη αρχικοποιημένες σε 0 ώστε να δουλέψει σωστά ο τύπος. (Βλέπε μορφή πίνακα στο παράδειγμα παρακάτω)

Ο αλγόριθμος ξεκινάει από το OPT(1,1) και προσπελαύνει τα κελιά γραμμή γραμμή, μέχρι το κελι OPT(n,m) με πολυπλοκότητα **O(n x m).**
Στο τέλος αρκεί να βρούμε το **max** των στοιχείων του OPT, με πολυπλοκότητα **O(n x m),** το οποίο θα είναι και το τελικό μας αποτέλεσμα.
Συνεπώς, η **συνολική πολυπλοκότητα** του θα είναι **O(n x m).**

Ένα απλό παράδειγμα γεμάτου πίνακα για X=(a,b,c), Y=(c,a,b):

	0	**a**	**b**	**c**
0	0	0	0	0
c	0	0	0	1
a	0	1	0	0
b	0	0	2	0

Τελικα, με εύρεση μέγιστου, βρίσκουμε το μέγιστο μήκος υποακολουθίας = 2.

2) Για ακολουθία **X=(x1, x2, … xn)** θετικών ακεραίων:
Έστω **OPT(j)** το μέγιστο βάρος στοιχείων του συνόλου μέχρι το j στοιχείο

OPT(j) = max{ OPT(j-2) + w(j), OPT(j-1) }

Αναλυτικά:
> Συμπεριλάβουμε to **j** (γεγονος που **αποκλείει** να συμπεριληφθεί το **j-1**) οπότε στο μέγιστο βάρος του συνόλου μέχρι αυτό (**OPT(j-2)**), προσθέτουμε το βάρος προσθήκης του j που παίρνουμε από τον πίνακα w.
> Δεν συμπεριλαμβάνουμε το j, οπότε κρατάμε το μέγιστο βάρος μέχρι το j-1 (**OPT(j-1)**);

Σχόλια:
α. Η σχέση εφαρμόζεται σε έναν πίνακα μεγέθους **n+1**, από το **OPT(2)** μέχρι το **OPT(n),** (n για τα στοιχεία του αρχικού συνόλου + 1 μηδενικό στην αρχή) με πολυπλοκότητα **O(n)**.
β. Το μηδενικό κελί του πίνακα αντιστοιχεί στην επιλογή κανενός από τα στοιχεία, οπότε έχει βάρος 0 ώστε να λειτουργεί σωστά η σχέση. (**OPT(0)=0**)
γ. Το πρώτο κελί του πίνακα αντιστοιχεί στην επιλογή του πρώτου στοιχείου του πίνακα και έχει βάρος ακριβώς το βάρος του στοιχείου αυτού, αφού δεν υπάρχει προηγούμενο στοιχείο. (**OPT(1)=w(1)**)
δ. Πριν την εκτέλεση, γεμίζουμε τον πίνακα **w** με τα βάρη των στοιχείων του συνόλου, **σύμφωνα με τον τύπο που μας δόθηκε** ώστε να τα έχουμε έτοιμα, με πολυπλοκότητα **O(n)**. (Εδώ, βαρος στοιχείου j = άθροισμα όλων των στοιχείων της ακολουθίας X από 1 μέχρι j, αρα για το βάρος w(j) αρκεί να προσθέτουμε το xj με το βαρος w(j-1))

Στο τέλος μπορούμε πλέον να βρούμε τα στοιχεία που αποτελούν το μέγιστο ανεξάρτητο σύνολο με **backtracking,** με κόστος **O(n)**.
Η **συνολική πολυπλοκότητα** είναι **O(n)**.

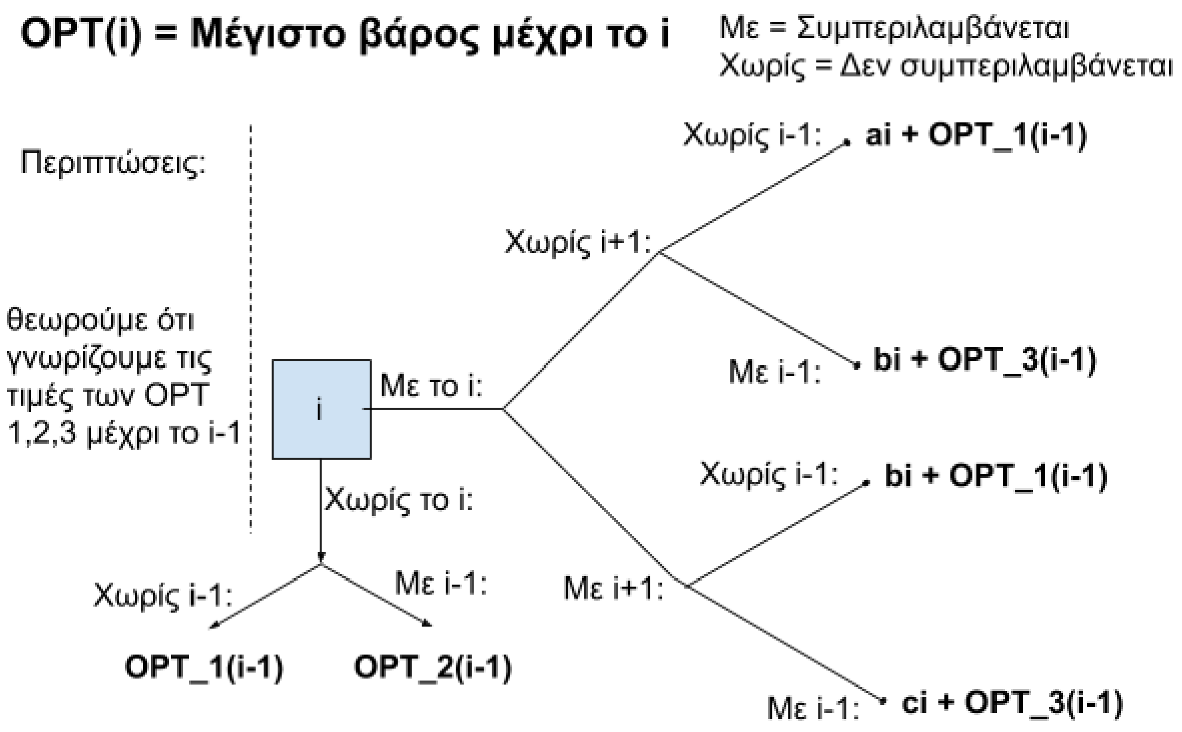
3) Για σύνολο από τριάδες (ai, bi, ci), i = 1,...,n:

Έστω **OPT(i)** = Το μέγιστο δυνατό κέρδος μέχρι την θέση i, και:
α. OPT_1(i)=Το κέρδος μέχρι τη θέση i, αν δεν ανοίξει κατάστημα στο i.
β. OPT_2(i)=Το κέρδος μέχρι τη θέση i, αν ανοίξει κατάστημα στο i, αλλά όχι στο i+1.
γ. OPT_3(i)=Το κέρδος μέχρι τη θέση i, αν ανοίξει κατάστημα και στο i και στο i+1.

Τότε η αναδρομική σχέση που θέλουμε απλά μεγιστοποιεί το δυνατό κέρδος για όλες τις πιθανές περιπτώσεις:
OPT(i) = max{OPT_1(i), OPT_2(i), OPT_3(i)}, με:

OPT_1(i)=max{ OPT_2(i-1), OPT_1(i-1) }
OPT_2(i)=max{ ai+OPT_1(i-1), bi+OPT_3(i-1) }
OPT_3(i)=max{ bi+OPT_1(i-1), ci+OPT_3(i-1) }

Η λογική της αναδρομικής σχέσης φαίνεται στο σχήμα:



Πιο αναλυτικά, έχουμε:
> OPT_1: Δεν ανοίγει μαγαζί στη θέση i οπότε έχουμε **μέγιστο** δυνατό κέρδος το μέγιστο κέρδος μέχρι την i-1 αν ανοίξει σε αυτή χωρίς επόμενο(**OPT_2**), ή το μέγιστο κέρδος αν δεν ανοίξει ούτε στην i-1 κατάστημα **(OPT_1).**
> OPT_2: Ανοίγει στην i αλλά όχι στην i+1 => θα πάρουμε το **μέγιστο** δυνατό κέρδος από τις δύο πιθανές περιπτώσεις που προκύπτουν: **α)** ο i δεν έχει προηγούμενο κόμβο i-1, οπότε το κέρδος του θα είναι **ai** + το κέρδος που θα υπάρχει μέχρι το i-1 χωρίς όμως το i-1 (**OPT_1), β)** ο i έχει i-1 οπότε το κέρδος του θα ειναι **bi** + το κέρδος μέχρι το i-1 αν αυτό συμπεριλαμβάνεται και συμπεριλαμβάνεται και το επόμενο του δηλαδή το i (**OPT_3).**
>OPT_3: Ανοίγει στην i και στην i+1 => θα πάρουμε το **μέγιστο** δυνατό κέρδος από τις δύο πιθανές περιπτώσεις που προκύπτουν: **α)** ο i δεν έχει προηγούμενο κόμβο i-1, οπότε το κέρδος του θα ειναι **bi** + το κέρδος μέχρι το i-1 χωρίς το i-1(**OPT_1**), **β)** ο i έχει i-1, οπότε έχει κέρδος **ci** + το κέρδος μέχρι το i-1 έχοντας επόμενο(**OPT_3)**

Σχόλια:
α. Η OPT(i) υπολογίζεται για κάθε i από 1 μέχρι n, εξου και η πολυπλοκότητα **O(n).**
β. Στο τέλος για να βρούμε τον ακριβή βέλτιστο συνδυασμό τοποθεσιών, απλά κάνουμε **backtracking** στις optimal επιλογές της OPT.
γ. Για **i=1 δεν ορίζεται το ci** οπότε από την OPT_3(i) υπολογίζουμε μόνο την περίπτωση **bi+OPT_1(i-1).**
δ. Για **i=n δεν ορίζεται η** OPT3(i) οπότε δεν την υπολογίζουμε
ε. Στις αναδρομικές μας σχέσεις, θεωρούμε ότι έχουν στην αρχή τους το μηδενικό στοιχείο (**OPT_1/2/3(0)=0**), ώστε να μπορεί να πάρει τιμή σωστα το 1ο στοιχείο αφού ο τύπος χρειάζεται την τιμή OPT…(i-1)

4) Έστω OPT(i) το ελάχιστο δυνατό κόστος από την αρχή μέχρι να κάνουμε και την i-οστή διαδρομή. (n-1 διαδρομές σύνολο)

Η αναδρομική σχέση που συνδέει τις λύσεις των προηγούμενων υποπροβλημάτων με την λύση του OPT(i) είναι η:
OPT(i) = min{ OPT(i-1) + ri , OPT(i-4) + B }

Αναλυτικά θελουμε την πιο οικονομική από τις εξής διαθέσιμες επιλογές:
> OPT(i-1) + ri: Το κόστος από την αρχή μέχρι την i-1 + το κόστος του ταξί ri για την i
> OPT(i-4) + B: Το κόστος από την αρχή μέχρι το σημείο από το οποίο πήραμε το Μουφερ + το στανταρ κόστος B του μούφερ για τις 4 διαδρομές μέχρι και την i

Σχόλια:
α. Εκτελούμε τον αλγόριθμο σε πίνακα μεγέθους n+4, αρχικοποιούμε τα πρώτα 4 κελιά με 0 ώστε να έχει πάντα τιμή το OPT(i-4), και αρχίζουμε την εκτέλεση από την 5η θέση η οποία αντιστοιχεί στο i=1, n φορές μέχρι το τέλος του πίνακα.

β. Αφού ο OPT(i) Εκτελείται n φορές λόγω της απομνημόνευσης και του τρόπου που γεμίζει ο πίνακας, η πολυπλοκότητα θα είναι **O(n).**

γ. Στο τέλος παίρνουμε την βέλτιστη (ελάχιστη - πιο οικονομική) λύση, δηλαδή τον πίνακα μεγέθους n-1 με τον βέλτιστο συνδυασμό ταξί και Μούφερ κάνοντας **backtracking** πάνω στο βέλτιστο αποτέλεσμα του OPT.

(Γενική σημείωση: Η κυρία Φουτρουνέλη μας είπε ότι δεν χρειάζεται να περιγράψουμε τα backtracking οπότε παρέλειψα την περιγραφή. Άμα τελικά χρειάζεται περιγραφή της μεθοδολογίας backtracking, μπορείτε να μου στείλετε email για διευκρινίσεις)