Αλγόριθμοι και πολυπλοκότητα
Εργασία: 2η Προγραμματιστική

Διδάσκοντες: Παγουρτζής Α., Σούλιου Δ.,Φωτάκης Δ.

Ονοματεπώνυμο: Σεβαστού Νικολέτα
Α.Μ.: 711514 22 00015
email:nikolesev@gmail.com

ΑΛΜΑ

2023-2024

Άσκηση 1

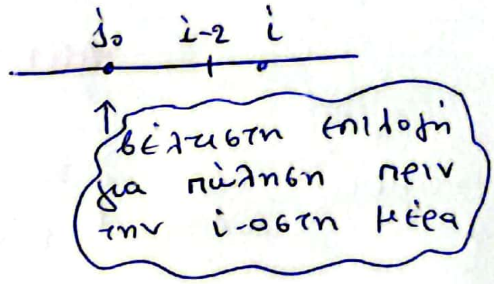
Έστω ότι βρίσκομαι στην i-οστή μέρα & έχω κάνει λ αγοραπωλησίες μέχρι. Συμβολίζω το βέλτιστο κόστος

$$PR(i,\lambda) = \max \begin{cases} PR(i-1,\lambda) \\ \max\limits_{1 \le j \le i-2} \{ P(i) - P(j+1) + profit(j, \lambda-1) \} \end{cases}$$

$i \in \{3, N\}$

$PR(1,\lambda) = 0 \qquad (\lambda = 0 < \lfloor i/2 \rfloor)$

$$PR(2,\lambda) = \begin{cases} 0, & \lambda = 0 \\ \max\{P(2)-P(1), 0\}, & \lambda = 1 \end{cases}$$




```
CHOKOLATE (N, k, P)
#P: πίνακας με τιμές ανά ημέρα
   D = new_matrix (N, k)   # διάστασης N×K, πίνακας με τα κέρδη
   D[1,0] = 0
   D[2,0] = 0
   D[2,1] = max {P(2)-P(1), 0}
   max ← -P(1)
   for  i=3 to N
        λ ← 0
        while (λ ≤ ⌊i/2⌋ & λ ≤ k)
              λ ← λ+1          # θα υπολογίσει έτσι & το
                               # D[i-1,λ]
              if D[i-2,λ-1] - P(i-1) > max
                    max ← D[i-2,λ-1] - P(i-1)
              end
              D[i,λ] ← D[i-1,λ]
              if  P(i)+max > D[i-1,λ]
                    D[i,λ] ← P(i)+max
              end
        end
   end
   return D(N,k)
```

Άσκηση 2

- Έστω κορυφή i (αντί για V_i, χάριν συντομίας)

Συμβολισμοί: **Parent(i)**: πατέρας της i στο DFS δέντρο

TIME(i): min / βέλτιστος χρόνος σε sec για παράδοση των γραμμάτων στην ρίζα

Dist(i,j): απόσταση (ίσως να είναι αποτέλεσμα αθροισμάτων) της i από τη j (με j πρόγονος της i ή i πρόγονος της j)

A(i): σύνολο προγόνων της i

$$TIME(i) = \min_{v \in A(i)} \left\{ S(i) \cdot Dist(i,1),\ P(v) + S(i) \cdot Dist(i,v) + S(v) \cdot Dist(v,1) \right\}$$

Θα χρησιμοποιήσω λίστα γειτνίασης για την αποθήκευση του γραφήματος & των αποστάσεων, ενώ για τα P(i), S(i) έναν πίνακα (για το καθένα) διάστασης 1x(N-1)

Για την αποθήκευση του TIME, θα χρησιμοποιήσω πίνακα 1x(N-1) πάλι.

Η αρχικοποίηση των πινάκων P, S θεωρώ ότι έχει γίνει ενώ του TIME γίνεται σε σταθερό χρόνο.

```
               λίστα γειτνίασης με αποστάσεις
DELIVERY (G, P, S, N)
    t ← 0   # καθολική μετ/τη
    Q ← new_stack ( )   # καθολική δομή
    TIME ← new_matrix (1, N-1)   # καθολική δομή
    for all v ∈ V do
        m[v] ← A   # v: ανεξερεύνητη
        A[v] ← NULL   # πατέρας της v : αρχικοποίηση
    end
    for all v ∈ V do
        if m[v] = A then
            DFS (v)
        end
    end
    return TIME
```

```
DFS (v, G, P, S, N)
    m[v] ← Y    # υπό εξερεύνηση
    t ← t+1
    d[v] ← t
    push (v, Q)
    for all i ∈ Adj[v] do
        dist ← 0
        if m[i] = A then
            A[i] ← v    # πατέρας της i είναι η v
            push (i, Q) , DFS[i, G, P, S, N]
            dist ← dist + D[i, v]
            if Adj[i] = A[i] # η λίστα με τους γείτονες
                             # της i, η οποία είναι φύλλο
                TIME[i] ← dist · S(i)
                dist 2 ← 0
                j = i
                pop(Q)                                  ← ο γονιός της i
                while IsEmpty (Q) = False
                    dist 2 ← dist 2 + D[j, top(Q)]
                    j = pop(Q)
                    dist ← dist - D[i, j]
                    if TIME[i] > [P(j-1) + S(j-1) · dist +
                    + S(i-1) · dist 2] # j-1, i-1 γιατί ο
                    # πίνακας έχει N-1 θέσεις
                          TIME[i] ← P(j-1) + S(j-1) · dist
                            + S(i-1) · dist 2
                    end_if
                end_while
            end_if
        end_if
    end for
  m[v] ← E    # εξερευνημένη
  t ← t+1
  f[v] ← t
```