ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

2η Γραπτή Άσκηση

# Αλγόριθμοι & Πολυπλοκότητα

*Σπουδαστής:*
Διονύσης ΖΗΝΔΡΟΣ (06601)
<dionyziz@gmail.com>

*Διδάσκοντες:*
Στάθης ΖΑΧΟΣ
Δημήτρης ΦΩΤΑΚΗΣ

31 Δεκεμβρίου 2011

# Άσκηση 1

**Algorithm 1** Άσκηση 1

**procedure** MOTORCYCLE($l, u, n, v, T$)
  **for** $i \leftarrow 1$ to $n$ **do**
    $A[i].u \leftarrow u[i]$
    $A[i].l \leftarrow l[i]$
    $A[i].i \leftarrow i$
  $A \leftarrow$ sort $A$ by $u$
  $S \leftarrow \emptyset$
  **for** $i \leftarrow 1$ to $n$ **do**
    $speed \leftarrow v + A[i].u$
    $t \leftarrow \frac{A[i].l}{speed}$
    **if** $T > t$ **then**
      $S \leftarrow S \cup \{(A[i].i, A[i].l)\}$
      $T \leftarrow T - t$
    **else**
      $S \leftarrow S \cup \{(A[i].i, T * speed)\}$
      *break*
  **return** $S$

# Άσκηση 2

# Άσκηση 3

## α)

Ένα παράδειγμα για το οποίο ο άπληστος αλγόριθμος δεν δίνει το βέλτιστο αποτέλεσμα είναι το εξής:

| | | |
|---|---|---|
| 0 | 4999 | 0 |
| 4999 | 5000 | 4999 |
| 0 | 4999 | 0 |
| 0 | 0 | 0 |

Ενώ ο άπληστος αλγόριθμος θα επιλέξει να τοποθετήσει ένα βότσαλο στο κουτί με αξία 5000, η βέλτιστη λύση επιτυγχάνεται επιλέγοντας τα 4 κουτάκια που είναι γείτονές του και δίνουν συνολική αξία 19,996.

## β)

Η λύση δυναμικού προγραμματισμού εντοπίζει τη βέλτιστη διάταξη σε γραμμικό χρόνο διασχίζοντας τη σκακιέρα από τα αριστερά προς τα δεξιά.

Σε κάθε βήμα, όταν επεξεργαζόμαστε την $i$-οστή στήλη, αποθηκεύουμε το βέλτιστο ποσό $J[i, \mu]$ που μπορούμε να κερδίσουμε από το μέρος της σκακιέρας που αποτελείται από τις στήλες 1 έως $i$ για κάθε πιθανό συνδυασμό βοτσάλων $\mu$ που μπορούμε να τοποθετήσουμε στην $i$-οστή στήλη. Έχοντας αυτά τα δεδομένα, ο υπολογισμός της βέλτιστης λύσης για την $(i+1)$-οστή στήλη προκύπτει από την εξής αναδρομική σχέση:

$$\Omega = \{x \in 2^{\{1,2,3,4\}} : \forall \alpha, \beta \in x : \alpha - 1 \neq \beta\}$$
$$J[i, \mu] = \max\{J[i-1, \mu'] + \sum_{q \in \mu} C[i, q] : \mu' \in \Omega \wedge \mu' \cap \mu = \emptyset\}$$

**Algorithm 2** Άσκηση 3β

**procedure** $\textsc{Chess}(C, n)$
    $M \leftarrow \{x \in 2^{\{1,2,3,4\}} : \forall \alpha, \beta \in x : \alpha - 1 \neq \beta\}$
    **for all** $\mu \in M$ **do**
        $J'[\mu] \leftarrow 0$
    **for** $i \leftarrow 0$ to $n$ **do**
        **for** $\mu \in M$ **do**
            $J[\mu] \leftarrow 0$
            **for** $\nu \in M$ **do**
                **if** $\nu \cap \mu = \emptyset$ **then**
                    $J[\mu] \leftarrow \max(J[\mu], J'[\nu] + \sum_{q \in \mu} C[i, q])$
        $J' \leftarrow J$
    **return** $\max(\{J[\mu] : \mu \in M\})$

# Άσκηση 4

**Algorithm 3** Άσκηση 4

**procedure** LineSplit($l, n, C$)
    $J[0] \leftarrow 0$
    **for** $i \leftarrow 1$ to $n$ **do**
        $J[i] \leftarrow \infty$
        $cost \leftarrow C + 1$
        **for** $j \leftarrow i$ downto 1 **do**
            $cost \leftarrow cost - (l[j] + 1)$
            **if** $cost < 0$ **then**
                *break*
            $J[i] \leftarrow \min(J[i], J[j-1] + cost^2)$
    **return** $J[n]$

## Άσκηση 5

**Algorithm 4** Άσκηση 5

```
procedure SERVERS(b, c, n)
    Ω ← {i : b[i] > 0}
    if Ω = ∅ then
        return ∅
    K ← max(Ω)
    JS[K] ← c[K]
    J[K] ← c[K]
    W[K] ← K
    for i ← K − 1 downto 1 do
        JS[i] ← J[i + 1] + C[i]
        J[i] ← ∞
        for j ← i to W[i + 1] do
            cost ← JS[j]
            for l = i to j do
                cost ← cost + (j − l)B[l]
                if cost >= J[i] then
                    break
            if cost < J[i] then
                J[i] ← cost
                W[i] ← j
    S ← {S_K}
    i ← W[1] + 1
    while i ≤ K do
        S ← S ∪ {S_{i−1}}
        i ← W[i] + 1
    return S
```

## Άσκηση 6

**α)**

**β)**

**γ)**