Άπληστοι Αλγόριθμοι

Δημήτρης Φωτάκης

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

Πανεπιστήμιο Αιγαίου

Απληστία

- Αλγόριθμοι βελτιστοποίησης λειτουργούν σε βήματα. Βήμα: επιλογή για μορφή (βέλτιστης) λύσης.
- **Δυναμικός Προγραμματισμός**
  - Λύνει **όλα** τα υπο-προβλήματα.
  - Συνδυάζει «κατάλληλες» επιμέρους λύσεις για βέλτιστη.
  - Λύση **όλων** υπο-προβλημάτων εγγυάται βέλτιστη λύση αλλά κοστίζει σημαντικά σε υπολογιστικό χρόνο.
- **Απληστία**
  - (Άπληστη) επιλογή που φαίνεται καλύτερη με βάση τρέχουσα κατάσταση και κάποιο (απλό) κριτήριο.
  - Ίδια στρατηγική στο υπο-πρόβλημα που προκύπτει.
  - Λύση **αναγκαίων** υπο-προβλημάτων: αποδοτικό υπολογιστικά αλλά δεν δίνει πάντα τη βέλτιστη λύση.

Απληστία

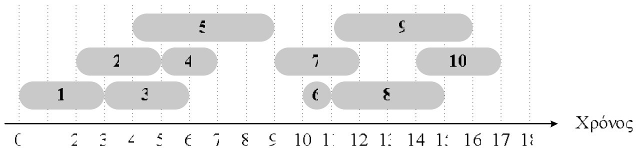
- Ταξινόμηση συνιστωσών με βάση κάποιο κριτήριο (π.χ. σακίδιο: αντικείμενα σε **φθίνουσα σειρά αξία / μέγεθος**).
- (Αμετάκλητη) επιλογή αν «καλύτερη» συνιστώσα θα συμπεριληφθεί στη λύση.
  - Επιλογή βασίζεται σε κάποιο απλό κριτήριο.
- Ίδια στρατηγική σε υπο-πρόβλημα που προκύπτει.
  - **Προσαρμοστικός**: αλλάζει ταξινόμηση σε κάθε βήμα.
  - **Μη-προσαρμοστικός**: ίδια ταξινόμηση σε όλα τα βήματα.
- Χρόνος εκτέλεσης καθορίζεται από χρόνο ταξινόμησης.
- Βέλτιστη λύση: **απόδειξη ορθότητας** με επαγωγή.
  - Ιδιότητα **άπληστης επιλογής**.
  - Ιδιότητα βέλτιστων επιμέρους λύσεων.

Επιλογή Δραστηριοτήτων

- *n* δραστηριότητες: αρχή και τέλος $[s_i, f_i) : \ f_i > s_i \geq 0$ (π.χ. μαθήματα, υπολογιστικές διεργασίες).
- Δρομολόγηση χωρίς χρονικές επικαλύψεις σε κοινό πόρο (π.χ. αίθουσα διδασκαλίας, επεξεργαστής).
- Ζητούμενο: δρομολόγηση **μέγιστου** #δραστηριοτήτων.

Παράδειγμα

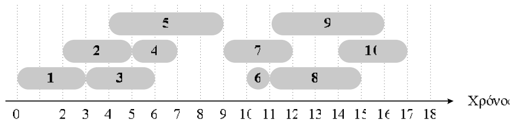
i	1	2	3	4	5	6	7	8	9	10
s_i	0	2	3	5	4	10	9	11	11	14
f_i	3	5	6	7	9	11	12	15	16	17



- Βέλτιστη λύση: **4** δραστηριότητες.
  Π.χ. {1, 3, 4, 6}, {2, 4, 7, 10}, {1, 4, 7, 10}, …

Άπληστος Αλγόριθμος

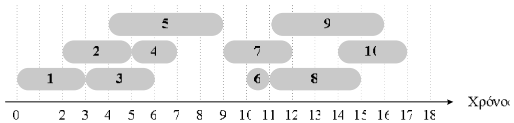
- Ταξινόμηση σε … ;
  - Αύξουσα σειρά **χρόνου τερματισμού.**
- Επόμενη δραστηριότητα
  - **Δρομολογείται** αν είναι εφικτό (πόρος είναι ελεύθερος).
  - **Αγνοείται** αν δρομολόγηση δεν είναι εφικτή.



Υλοποίηση

greedySelection$((s_1, f_1), \ldots, (s_n, f_n))$
  /* $f_1 \le f_2 \le \cdots \le f_n$ */
  $C \leftarrow \{1\};\ j \leftarrow 1;$
  for $i \leftarrow 2$ **to** n **do**
    if $s_i \ge f_j$ **then**
      $C \leftarrow C \cup \{i\};\ j \leftarrow i;$
  return(C);

Χρόνος **O(n log n)** (ταξινόμηση ως προς χρόνο τερματισμού).



Ορθότητα

- Επαγωγή στον #δραστηριοτήτων.
  Υποθέτουμε πάντα ότι $f_1 \le f_2 \le \cdots \le f_n$
- 1 δραστ. επιλέγεται πάντα: βέλτιστη λύση.
- Έστω αλγ. υπολογίζει βέλτιστη λύση για ≤ n – 1 δραστ.
  Θδο. υπολογίζει βέλτιστη λύση για σύνολο A με *n* δραστ.
  - Αλγ. επιλέγει 1 (f_1) και βέλτιστη λύση $A_1 = \{i : s_i \ge f_1\}$
  - C^*(A) βέλτιστη λύση και *j* δραστ. C^*(A) ολοκληρώνεται πρώτη.
  - Άπληστη επιλογή: $f_j \ge f_1 \Rightarrow (C^*(A) \setminus \{j\}) \cup \{1\}$ **εφικτή**.
  - $|C^*(A)| \le 1 + |C^*(A_1)|$ = #δραστηριοτήτων άπληστου αλγ.
- Άπληστος αλγόριθμος υπολογίζει βέλτιστη λύση για A.

Ορθότητα

- Αποδείξαμε ότι $|C^*(A)| = 1 + |C^*(A_1)|$
- Ιδιότητα **άπληστης επιλογής**:
  - (Άπληστη) επιλογή δραστ. με ελάχιστο χρόνο ολοκλήρωσης οδηγεί σε συνολικά βέλτιστη λύση.
- Ιδιότητα **βέλτιστων επιμέρους λύσεων**:
  - Βέλτιστη λύση περιέχει βέλτιστη λύση για υπο-πρόβλημα A_1 (δραστ. που δεν επικαλύπτονται με πρώτη).

Δρομολόγηση Εργασιών

- **Ένας** εξυπηρετητής (π.χ. επεξεργαστής, εκτυπωτής, ταμίας).
- Σύνολο *N* με *n* εργασίες: χρόνο εκτέλεσης $t_i > 0$ (π.χ. υπολογιστικές διεργασίες, εκτυπώσεις, συναλλαγές).
- Δρομολόγηση για ελαχιστοποίηση **συνολικού** (μέσου) χρόνου εξυπηρέτησης.
  - Δρομολόγηση: μετάθεση $\pi : \{1, \ldots, n\} \mapsto \{1, \ldots, n\}$
  - Χρόνος εξυπηρέτησης *i* : $c_i(\pi) = \sum_{j:\pi(j) \leq \pi(i)} t_j$
  - Συνολικός χρόνος εξυπηρέτησης: $T(\pi) = \sum_{i=1}^{n} c_i(\pi)$
- $t_1 = 8$, $t_2 = 7$, $t_3 = 2$, $t_4 = 5$.
  - 1, 2, 3, 4: 8 + 15 + 17 + 22 − **62**.
  - 3, 4, 2, 1: 2 + 7 + 14 + 22 − **45**.

Άπληστος Αλγόριθμος

- Δρομολόγηση σε αύξουσα σειρά χρόνου εκτέλεσης:
$$t_1 \leq t_2 \leq \cdots \leq t_n$$
- Συνολικός χρόνος εξυπηρέτησης:
$$\begin{aligned} T &= t_1 + (t_1 + t_2) + (t_1 + t_2 + t_3) + \cdots + (t_1 + \cdots + t_n) \\ &= n\,t_1 + (n-1)t_2 + (n-2)t_3 + \cdots + t_n \\ &= \textstyle\sum_{i=1}^{n} (n - i + 1)t_i \end{aligned}$$
  - Βέλτιστος γιατί **όσο μεγαλύτερος** χρόνος εκτέλεσης, **τόσο λιγότερες φορές** συνεισφέρει στο συνολικό.

Ορθότητα

- Έστω π* βέλτιστη δρομολόγηση, *T*(π*) συνολικός χρόνος. λ*(*j*) : σειρά εργασίας *j* στη βέλτιστη δρομολόγηση.
- Έστω π* διαφορετική από άπληστη:
  - *k* **πρώτη** που δρομολογείται αργότερα στην π*: λ*(*k*) > *k*
  - *j* αυτή που δρομολογείται *k*-οστή στην π*: λ*(*j*) = *k*
  - ... συμφωνούν σε *k* – 1 αρχικές: *k* < *j* και $t_k \leq t_j$
  - t_k και t_j στο *T*(π*): $(n - k + 1)t_j + \cdots + (n - \lambda^*(k) + 1)t_k$
  - Ανταλλαγή *k* και *j*: $(n - k + 1)t_k + \cdots + (n - \lambda^*(k) + 1)t_j$ (*k* πηγαίνει στη θέση που έχει στην άπληστη δρομολόγηση).
  - Διαφορά: $(\lambda^*(k) - k)(t_k - t_j) \leq 0$
- Έτσι π* γίνεται ίδια με άπληστη **χωρίς αύξηση** χρόνου.
  - **Άπληστη** δρομολόγηση είναι **βέλτιστη**.

Ιδιότητες

- Ιδιότητα **άπληστης επιλογής**:
  - Για κάθε *k*, βέλτιστη δρομολόγηση π* συμφωνεί με άπληστη στη σειρά των *k* πρώτων εργασιών.
  - (Άπληστη) επιλογή συντομότερης διαθέσιμης → βέλτιστη.
- Ιδιότητα **βέλτιστων επιμέρους λύσεων**:
  - $T = n\,t_1 + (n-1)t_2 + (n-2)t_3 + \cdots + t_n$
  - Αν αγνοήσουμε t_1, π* παραμένει βέλτιστη για υπόλοιπες.
- Απόδειξη **ορθότητας**: επαγωγική εφαρμογή ιδιότητας άπληστης επιλογής.

Κλασματικό Πρόβλημα Σακιδίου

- Δίνονται *n* αντικείμενα και **σακίδιο** μεγέθους B. Αντικείμενο *i* έχει **μέγεθος** και **αξία**: (s_i, p_i)
- Αντικείμενο *i* μπορεί να συμπεριληφθεί στο σακίδιο σε οποιοδήποτε ποσοστό.
- Ζητείται συλλογή μέγιστης αξίας που χωράει στο σακίδιο.

$$\begin{array}{rl} & \max \sum_{i=1}^n f_i p_i \\ \text{υπό περιορισμούς} & \sum_{i=1}^n f_i s_i \leq B \\ & f_i \in [0,1] \quad \forall i \in [n] \end{array}$$

  - Αντικείμενα: { (3, 5), (2, 7), (4, 4), (6, 8), (5, 4) }
    Μέγεθος σακιδίου: **10**.
  - Βέλτιστη λύση = { 1×(3, 5), 1×(2, 7), (5/6)×(6, 8) }
    Βέλτιστη αξία = 5 + 7 + (5/6)×8 = 18.3333

Άπληστος Αλγόριθμος

- Αντικείμενα N = {1, …, *n*}, σακίδιο μεγέθους B. Βέλτιστη λύση $F^* = (f_1^*, f_2^*, \ldots, f_n^*)$
- **Βέλτιστες Επιμέρους Λύσεις**. Αγνοούμε αντικείμενο *i* :
  - $F_{-i}^* = (f_1^*, \ldots, f_{i-1}^*, f_{i+1}^*, \ldots, f_n^*)$ **βέλτιστη λύση** για N \ { *i* } με σακίδιο $B - f_i^* s_i$
- Αντικείμενο *i* : $r_i = p_i / s_i$ (αξία / μονάδα μεγέθους)
- Αντικείμενα σε φθίνουσα σειρά r_i : $r_1 \geq r_2 \geq \cdots \geq r_n$
  - Όσο περισσότερο από *i* χωράει στο (διαθέσιμο) σακίδιο. $f_i = \begin{cases} 1 & \text{αν } B \geq s_i \\ s_i/B & \text{αν } B < s_i \end{cases}$
  - Αναπροσαρμογή διαθέσιμου σακιδίου και επόμενο αντικείμενο. $B = B - f_i s_i$

Υλοποίηση

```
greedyKnapsack(B, (s_1, p_1), ..., (s_n, p_n))
    for i ← 1 to n do
        r_i ← p_i/s_i; f_i ← 0;
    Ταξινόμηση r_1 ≥ r_2 ≥ ··· ≥ r_n
    for i ← 1 to n do
        if B ≤ 0 then f_i ← 0;
        else if B ≥ s_i then
                f_i ← 1; B ← B − s_i;
            else
                f_i ← B/s_i; B ← 0;
```

Χρόνος **O(n log n)** (ταξινόμηση ως προς λόγο αξίας / μέγεθος).

Άπληστη Επιλογή

- Έστω βέλτιστη λύση $F^* = (f_1^*, f_2^*, \ldots, f_n^*)$
  Έστω άπληστη λύση $F = (f_1, f_2, \ldots, f_n)$
- Ιδιότητα **άπληστης επιλογής**:
  - Υπάρχει βέλτιστη λύση: $f_1^* = f_1$
  - Άπληστη επιλογή: καμία λύση με περισσότερο από αντικ. 1.
  - Αν βέλτιστη $f_1^* < f_1$, αντικαθιστούμε $(f_1 - f_1^*)s_1$ μονάδες άλλου αντικ. (ή κενού) με αντικ. 1:
    - Αποδεκτή λύση γιατί $B \geq f_1 s_1$
    - Αξία **δεν** μειώνεται.
- Απόδειξη ορθότητας με επαγωγική εφαρμογή ιδιότητας άπληστης επιλογής.

Ορθότητα

- Επαγωγή στον #αντικειμένων.
  - Βάση: 1 αντικείμενο. Άπληστη επιλογή: $f_1^* = f_1$
  - Επαγωγική υπόθεση: #αντικ. < *n* – 1, άπληστη = βέλτιστη.
  - Θεωρούμε *n* αντικείμενα.
  - Άπληστη επιλογή: $f_1^* = f_1$
  - Στιγμιότυπο με *n* – 1 αντικ. και σακίδιο $B - f_1 s_1$
  - Επαγωγική υπόθεση: $F_{-1} = (f_2, \ldots, f_n)$ **βέλτιστη** λύση.
  - $F^*_{-1} = (f_2^*, \ldots, f_n^*)$ **εφικτή** λύση.
  - Συνολικά για *n* αντικείμενα:
    $$f_1 p_1 + \sum_{i=2}^{n} f_i p_i \geq f_1^* p_1 + \sum_{i=2}^{n} f_i^* p_i$$
- **Άπληστος** αλγόριθμος υπολογίζει **βέλτιστη λύση.**

Απληστία

- Ταξινόμηση συνιστωσών με βάση κάποιο κριτήριο (π.χ. σακίδιο: αντικείμενα σε **φθίνουσα σειρά αξία / μέγεθος**).
- (Αμετάκλητη) επιλογή αν «καλύτερη» συνιστώσα θα συμπεριληφθεί στη λύση.
  - Επιλογή βασίζεται σε κάποιο απλό κριτήριο.
- Ίδια στρατηγική σε υπο-πρόβλημα που προκύπτει.
  - **Προσαρμοστικός**: αλλάζει ταξινόμηση σε κάθε βήμα.
  - **Μη-προσαρμοστικός**: ίδια ταξινόμηση σε όλα τα βήματα.
- Χρόνος εκτέλεσης καθορίζεται από χρόνο ταξινόμησης.
- Βέλτιστη λύση: **απόδειξη ορθότητας** με επαγωγή.
  - Ιδιότητα **άπληστης επιλογής**.
  - Ιδιότητα βέλτιστων επιμέρους λύσεων.

Απληστία vs Δυναμικός Προγρ.

- 0-1 Πρόβλημα Σακιδίου: **όχι** ιδιότητα άπληστης επιλογής.
  - Π.χ. Αντικείμενα: {(1, 1+ε), (B, B)}. Σακίδιο μεγέθους B.
- **Δυναμικός Προγραμματισμός**:
  - Λύνουμε **όλα** υπο-προβ/τα για απόφαση αν *i* στο σακίδιο.
  - Χρονοβόρος αλλά εγγυάται βέλτιστη λύση.
- **Απληστία**:
  - Γρήγοροι και απλοί αλγόριθμοι: μόνο **αναγκαία** υπο-προβ/τα.
  - (Καλές) προσεγγιστικές λύσεις σε πολλά προβλήματα.
  - Βέλτιστη λύση μόνο όταν άπληστη επιλογή (ως προς συγκεκριμένο κριτήριο επιλογής).
- **Απληστία** και **Δυναμικός Προγραμματισμός**:
  - Βέλτιστες επιμέρους λύσεις.

Απληστία και Ρέστα

- Κέρματα αξίας 1, 5, και 20 λεπτών.
- Ρέστα ποσό *x* με **ελάχιστο** #κερμάτων.
- Αλγόριθμος:
  - Όσο περισσότερα 20λεπτα: $c_{20}(x) = \lfloor x/20 \rfloor$
  - Όσο περισσότερα 5λεπτα: $c_5(x) = \lfloor (x - 20\, c_{20}(x))/5 \rfloor$
  - Υπόλοιπα 1λεπτα: $c_1(x) = x - 20\, c_{20}(x) - 5\, c_5(x)$
- Βέλτιστη λύση χρησιμοποιεί ίδιο #κερμάτων:
  - 20λεπτα: Δεν μπορεί περισσότερα. Βελτιώνεται αν λιγότερα.
  - Αν ίδιο #20λέπτων, τότε ίδιο #5λέπτων.
  - … **επαγωγή** στα είδη κερμάτων.
- Δουλεύει αλγόριθμος αν κέρματα 1, 12, και 20 λεπτών;
  - Π.χ. ρέστα 24 λεπτά.