Ορισμοί

- ***On-line* Αλγόριθμος**

  Κάθε αλγόριθμος, ο οποίος, δεχόμενος μία ακολουθία αιτήσεων, την επεξεργάζεται *σειριακά*, ώστε να περατώνει κάθε αίτηση *προτού αναλάβει την εξυπηρέτηση της επόμενης*
- Δεν ενδείκνυται κλασική ανάλυσή τους γιατί, ενδεχομένως, ένας κακόβουλος χρήστης μπορεί να εμφανίζει ακολουθίες που οδηγούν τον αλγόριθμο σε χείριστη συμπεριφορά

Ορισμοί (συν.)

- Αντ' αυτού, συγκρίνεται βάσει της επιδόσεως ενός *off-line* ιδεατού αλγορίθμου *O*, ο οποίος *γνωρίζει όλη την ακολουθία εξ αρχής*
- Η επίδοση του *off-line* αλγορίθμου *O* εκφράζει την δυσκολία (πολυπλοκότητα) της εισόδου
- Κατά συνέπεια, μιλούμε για *c*-ανταγωνιστικό αλγόριθμο ελαχιστοποιήσεως *A*, όταν, για οποιαδήποτε ακολουθία εισόδου σ, ισχύει:

$$f_A(\sigma) - cf_O(\sigma) \leq b, \quad b \text{ σταθερά, ίσως } f(c)$$

Ορισμοί (συν.)

- Ομοίως, μιλούμε για c-ανταγωνιστικό αλγόριθμο μεγιστοποιήσεως A, όταν, για οποιαδήποτε ακολουθία εισόδου σ, ισχύει:

  $f_A(\sigma) - c^{-1} f_O(\sigma) \geq b$, b σταθερά, ίσως $f(c)$

Είδη Αντιπάλων

- Συνήθως, ο *off-line* θεωρείται ως ***αντίπαλος,*** ο οποίος:
  - είτε ξέρει τα πάντα για τον *ντετερμινιστικό* αλγόριθμο *A* και, έτσι, διαλέγει την ακολουθία εισόδου, ώστε να τον παρασύρει σε χείριστη συμπεριφορά
  - είτε, καθώς ο *A* είναι τυχαίος, αλλά πάντοτε αποκαλύπτει τις επιλογές του, παρατηρεί την συμπεριφορά του *A* και, βάσει των *i*-1 επιλογών, διαλέγει την *i*-στή είσοδο (*προσαρμοζόμενος-adaptive*)
  - είτε, καθώς ο *A* είναι τυχαίος και δεν αποκαλύπτει τις επιλογές του, δεν μπορεί να κάνει τίποτε άλλο παρά να παρακολουθεί τον αλγόριθμο σε μία ακολουθία εισόδου (*επιλήσμων-oblivious*)

Αναζήτηση Τιμών

- Δοθείσης μίας ακολουθίας τιμών

  $\mathbf{p} = \langle p_1, p_2, \ldots, p_n \rangle, \qquad p_i \in [m, M],\ m, M > 0$

  αποκαλυπτόμενης τιμή-προς-τιμή, ο ‘παίκτης’ αναζητεί την μέγιστη (ή την ελάχιστη τιμή), με τους εξής κανόνες:
  - Αποδέχεται την τρέχουσα τιμή ως την μέγιστη (ελάχιστη)
  - Απορρίπτει την τιμή, και ζητά να δει την επόμενη
  - Ο χρονικός ορίζοντας του παιχνιδιού είναι πεπερασμένος:
    - Εάν δεν επιλέξει καμμία τιμή, μένει με την τελευταία p_n
- Εφαρμογές
  - Αναζήτηση του καλύτερου/ης κατά την πρόσληψη προσωπικού
  - Αναζήτηση χαμηλότερης τιμής αγοράς
  - Η-πλειστηριασμοί (e-auctions): αναζήτηση υψηλότερης τιμής προσφοράς

Ντετερμινιστικός Αλγόριθμος- Πολιτική Τιμής Εξασφαλίσεως (RRP)

- Αποδοχή της πρώτης προσφοράς μεγαλύτερης ή ίσης με:

$$\sqrt{mM}$$

ονομαζόμενης ως ***τιμή εξασφαλίσεως***

- Ο αλγόριθμος είναι $\sqrt{\varphi}$- ανταγωνιστικός ($\varphi = M/m$)
  - Έστω ότι στον παίκτη προσφέρεται η τιμή εξασφαλίσεως:
    - Μελλοντικά, θα του προσφερθεί M και, άρα, ο λόγος της επιδόσεως είναι:

$$M \,/ \sqrt{mM} = \sqrt{\varphi}$$

  - Του προσφέρεται τιμή *το πολύ* $\sqrt{mM}$ -1, οπότε θα μείνει με την τελευταία τιμή, η οποία, στην χειρότερη περίπτωση, είναι m:

$$(\sqrt{mM} - 1) / m < \sqrt{mM} / m = \sqrt{\varphi}$$

Πολιτική Τιμής Εξασφαλίσεως (RRP) (συν.)

- Ο λόγος είναι ο καλύτερος δυνατός
  - Θεωρήστε την ακολουθία $<p_1,p_2>$, με $p_1=x < M$
  - Εάν ο εκάστοτε αλγόριθμος επιλέξει x, τότε ο αντίπαλος προσφέρει $p_2 = M$, επιτυγχάνοντας λόγο M/x
  - Διαφορετικά, ο αντίπαλος προσφέρει $p_2 = m$, επιτυγχάνοντας λόγο x/m
  - Ισοσκελίζοντας τους δύο λόγους

    $$M/x = x/m,$$

    προκύπτει το αιτούμενο

Πιθανοτικός Αλγόριθμος

- Το όριο $\sqrt{\phi}$ δύναται να ξεπεραστεί με χρήση τυχαιότητας
- Έστω RRP(i) ο ντετερμινιστικός αλγόριθμος με τιμή εξασφαλίσεως $m2^i$. Τότε:

  Επιλέγουμε, ισοπίθανα (με πιθανότητα $1/k$), μία εκ των πολιτικών

  RRP(1), RRP(2), RRP(3),…, RRP(k)
- Θα δείξουμε ότι ο αλγόριθμος είναι ($l(\varphi)\log\varphi$)-ανταγωνιστικός, με

  $$\lim_{\varphi\to\infty} l(\varphi) = 1$$

  όπου, δίχως βλάβη της γενικότητας, $\varphi=2^k$, $k \in \mathrm{N}$

Πιθανοτικός Αλγόριθμος (συν.)

- Έστω ότι η μέγιστη τιμή που ο αντίπαλος επιλέγει είναι:

$$m2^{i+1}-\varepsilon \in [m2^i, m2^{i+1}), \qquad 0<\varepsilon<1$$

- Τότε

$$\mathbf{E}[f]=\sum_{j=1}^{i} (1/k)m2^j + \sum_{j>i}(1/k)m = m/k(2^{i+1}-2+k-i)$$

καθώς:

  - για $j \leq i$, η στρατηγική θα διαλέξει τουλάχιστον την τιμή εξασφαλίσεως $m2^j$
  - για $j > i$, θα πετύχει τουλάχιστον m (την ελάχιστη)

Πιθανοτικός Αλγόριθμος (συν.)

- Οπότε, ο λόγος είναι:

$$(m2^{i+1}-\varepsilon)/m/k(2^{i+1}-2+k-i)$$

$$< m2^{i+1}/m/k(2^{i+1}-2+k-i)$$

$$= k\,2^{i+1}/(2^{i+1}-2+k-i)$$

$$= \left(2^{i+1}/(2^{i+1}-2+k-i)\right)\log\varphi < l(\varphi)\log\varphi$$

με $l(\varphi) \sim 1$ για $i = (k-2)+1/\ln 2$

Κρυφή Μνήμη (Caching)

- Γρήγορη μνήμη *k* θέσεων (σελίδων) που παρεμβάλλεται μεταξύ κύριας μνήμης και εφαρμογής και αποθηκεύει τις τελευταίες προσπελάσεις
  - Εάν υπάρχει, hit! και αποφεύγουμε την προσπέλαση της μνήμης
  - Εάν δεν υπάρχει, miss! και την φέρνουμε από την κύρια μνήμη
    - Εάν υπάρχει χώρος στην cache, τότε την αποθηκεύουμε χωρίς πρόβλημα
    - Αλλιώς, πρέπει να δημιουργήσουμε ελεύθερο χώρο με την *έξωση (eviction) κάποιας*

Κρυφή Μνήμη (Caching) (συν.)

- Το πρόβλημα εμφανίζεται στα λειτουργικά, αλλά είναι παρόμοιο με αυτό των browsers.
- Κυρίαρχο σημείο: η πολιτική εξώσεως
  - LRU: *η παλαιότερα ανακτημένη*
  - LFU: *με την μικρότερη συχνότητα ανακτήσεως*
  - FiFo: *κατά σειρά τοποθετήσεως στην μνήμη*
  - Random Selection: *τυχαία επιλογή*

Δυσκολία caching

- Για οποιονδήποτε αλγόριθμο caching υπάρχουν αυθαίρετα μεγάλες ακολουθίες αιτήσεων σελίδων, τέτοιες ώστε ο λόγος ανταγωνιστικότητας να είναι τουλάχιστον όσο και το μέγεθος k της cache.
  - Έστω ότι οι cache του *on-line* και του *off-line* περιέχουν τις ίδιες k σελίδες, ενώ η πρώτη αίτηση είναι για σελίδα άγνωστη (miss)
  - Ονομάζουμε S το σύνολο αυτών των k+1 σελίδων
  - Η ακολουθία αιτήσεων σ:
    - αιτείται εκείνη η σελίδα από το S που δεν υπάρχει στην cache (πάντοτε υπάρχει τέτοια)

Δυσκολία caching (συν.)

- Ο *off-line* χωρίζει την *σ* σε μέγιστες υπακολουθίες (γύρους) *k διακριτών σελίδων*, και κάθε φορά διώχνει αυτήν που θα ζητηθεί πρώτη στον επόμενο γύρο. Πχ,
  - 4 2 3 |**1** 4 3 |**2** 4 1 |**3**...
  - {1,2,3},{1,4,3},{1,4,2}|{3,4,2},{3,1,2},{4,1,2}|{4,1,3}, … (*on-line*)
  - {1,2,3},{4,2,3},{4,2,3}|{4,1,3},{4,1,3},{4,1,3}| {4,1,2}, …(*off-line*)
- Άρα:
  - Ο *on-line* έχει μία αποτυχία ανά αίτηση
  - Ο *off-line* έχει μία αποτυχία ανά γύρο (δηλ., μία ανά *k* αιτήσεις)

LRU

- Προσπαθεί να εκμεταλλευτεί την χρονική «τοπικότητα» των σελίδων για να διαμορφώσει πολιτική εξώσεως
- Αιτεί μία ουρά προτεραιότητας, με αύξοντες αριθμούς κλειδιού-αντιπροσωπευτικούς της χρονικής στιγμής τελευταίας χρήσης (εύκολο)
- Έχει απόδοση: $f_{LRU} \leq kf_O + k$
  - Η σ χωρίζεται σε υπακολουθίες $\tau_0\tau_1\tau_2...\tau_m$ ώστε η τ_0 να περιέχει το πολύ k αποτυχίες LRU και οι υπόλοιπες ακριβώς k
  - Σε κάθε τ_i η *off-line* πρέπει να σημειώνει τουλάχιστον μία αποτυχία, με εξαίρεση ίσως την τ_0:
    - παρατηρήστε πως σε κάθε υπακολουθία τ_i , λόγω του τρόπου λειτουργίας της LRU, θα υπάρχουν ανακτήσεις ***τουλάχιστον*** k+1 διακεκριμένων σελίδων
    - Κατά την τ_0, εάν παρατηρηθεί διπλή ανάκτηση, τότε υπάρχουν ανακτήσεις ***τουλάχιστον*** k+1 διακεκριμένων σελίδων. Διαφορετικά, ίσως η *off-line* γλυτώσει την αποτυχία
  - Εάν οι δύο cache έχουν διαφορετικά περιεχόμενα, τότε η *off-line* πολιτική μπορεί να γλιτώσει το πολύ k ανακτήσεις

FiFo

- Εύκολη υλοποίηση με μία λίστα
- Ίδια πολυπλοκότητα με LRU (παρόμοιο επιχείρημα)

LFU

- Απαιτεί την δυσκολότερη υλοποίηση (ουρά μεγίστου)
- Δεν είναι k-ανταγωνιστική (!):

  Έστω η ακολουθία

  $$\tau_1 = \sigma_1\sigma_1\ldots\sigma_1\,\sigma_2\sigma_2\ldots\sigma_2\,\cdots\,\sigma_{k-1}\sigma_{k-1}\ldots\sigma_{k-1}$$

  των $l > k$ προσπελάσεων σε k-1 διαφορετικές σελίδες και

  $$\tau_2 = \sigma_k\sigma_{k+1}\sigma_k\,\sigma_{k+1}\ldots\,\sigma_k\,\sigma_{k+1}$$

  μία ακολουθία των $2l$-2 προσπελάσεων σε δύο νέες σελίδες.

  Τότε, η σύνθετη ακολουθία αιτήσεων

  $$\Sigma = \tau_1\tau_2$$

  - οδηγεί την LFU, μετά τις πρώτες $l(k$-1) ανακτήσεις, να αποτυγχάνει σε κάθε αίτηση ($2l$-2, συνολικά)
  - ο βέλτιστος να αποτυγχάνει μία φορά
  - *Άρα, κατά την* Σ*, ο λόγος* LFU/*off-line* = $2(l$-1) > $l > k$
  - Έστω Σ΄= ΣΣΣΣ···ΣΣ αυθαίρετα μεγάλη ακολουθία. Σε αυτήν, ο *λόγος* LFU/*off-line* > $l > k$

LiFo

- Δεν είναι k-ανταγωνιστική (!):
  Έστω η ακολουθία n ανακτήσεων

  $$\tau = \sigma_1\sigma_2 \ldots \sigma_{k-1}\sigma_k\sigma_{k+1}\,\sigma_k\sigma_{k+1}\sigma_k\,\sigma_{k+1}\ldots\sigma_k\,\sigma_{k+1}\ \ldots$$

  Τότε:
  - Η LiFo, μετά την k-στή αίτηση, αποτυγχάνει συνεχώς
  - Η *off-line* το πολύ k+1 φορές
  - $f_{\text{LiFo}}/f_O \geq (n\text{-}k)/k = n/k\text{-}1 > k$

Πολιτική Μαρκαδόρου (Marker)

- Τι ισχύει (χωρίς απόδειξη):
  Για κάθε τυχαίο αλγόριθμο caching, υπάρχει ακολουθία με λόγο ανταγωνιστικότητας:

$$c \geq H_k = \sum_{1 \leq k \leq j} 1/j$$

- Πολιτική Μαρκαδόρου
  - Προσπαθεί να εξομοιώσει την πολιτική LRU
    - Κάθε σελίδα έχει ένα μπιτ, αρχικά θεμένο στην τιμή ‘0’
    - Εάν ζητηθεί μία σελίδα, τότε το μπιτ της γίνεται ‘1’
    - Εάν πρέπει να εκδιωχθεί μία σελίδα, τότε διαλέγουμε μία από αυτές με μπιτ 0. Εάν όλες έχουν μπιτ 1, τότε
      - Μηδενίζουμε όλα τα μπιτ
      - Διαλέγουμε μία στην τύχη

Ανάλυση

- Η πολιτική μαρκαδόρου είναι (2log$k$)-ανταγωνιστική
  - Η ανάλυση γίνεται σε γύρους
  - Κάθε γύρος ξεκινά με όλες τις σελίδες να έχουν μηδενικά μπιτ, και τελειώνει όταν όλες έχουν σημειωθεί με αναμμένα μπιτ
  - Μία σελίδα χαρακτηρίζεται ***μπαγιάτικη*** ή ***έωλη (stale)*** εάν προϋπήρχε στην cache προ του γύρου
  - Αλλιώς ***νωπή (fresh)***

Ανάλυση (συν.)

- Εάν κατά τον *i*-στό γύρο ο μαρκαδόρος σημειώσει f_i προσπελάσεις νωπών σελίδων (f_i εξώσεις, δηλαδή) και b_i είναι οι σελίδες που υπάρχουν μόνον στην cache του βέλτιστου αλγορίθμου, τότε ο *off-line* θα κάνει τουλάχιστον $f_i - b_i$ εξώσεις
- Επίσης, ο *off-line* αλγόριθμος πρέπει να κάνει και b_{i+1} αντικαταστάσεις για να είναι έτοιμος για τον επόμενο γύρο
- Άρα, με το πέρας του *i*-στού γύρου, ο o*ff-line* έχει διενεργήσει τουλάχιστον:

$$\max\{f_i - b_i, b_{i+1}\} \geq (f_i - b_i + b_{i+1})/2$$

εξώσεις

Ανάλυση (συν.)

- Συνεπώς, σε όλους τους γύρους ο *off-line* έχει διενεργήσει ***τουλάχιστον***

$$\Sigma\left(\left(f_i - b_i + b_{i+1}\right)/2\right) = (b_{\kappa+1} - b_1)/2 + \Sigma\, f_i/2$$

  εξώσεις
- Ο μαρκαδόρος χρειάζεται $f_i + s_i$ εξώσεις, δηλ.,
  - Αυτές που δεν υπήρχαν προ του γύρου (φρέσκιες)
  - Συν το μ.ο. s_i των μπαγιάτικων που τις ξαναέφερε πίσω, αφότου εκδιώχτηκαν
- Με κάθε μπαγιάτικη σελίδα σ , έστω X_σ η τυχαία μεταβλητή δείκτης που είναι 1 εάν η σελίδα αιτείται μετά την εκδίωξή της, και 0 εάν όχι. Τότε:

$$s_i = \Sigma\, \mathbf{Prob}[X_\sigma = 1]$$

- **Prob**$[X_\sigma = 1] \leq$ # νωπών που υπάρχουν μέχρι της αιτήσεως της σ / # απροσπέλαστων έωλων (κάθε αίτηση φρέσκιας σελίδας, διώχνει και μία μπαγιάτικη) $\leq f_i$ / # απροσπέλαστων έωλων (μεγιστοποίηση όταν όλες έχουν μπει)
- Καθώς σημειώνονται k-f_i προσπελάσεις σε μπαγιάτικες σελίδες, προκύπτει πως:

$$s_i \leq f_i/k + f_i/(k\text{-}1) + \cdots + f_i/(f_i+1) \leq f_i \Sigma\, 1/j = f_i H_k$$

- Οπότε, $f_m = \Sigma_i \left(f_i H_k + f_i\right)$ είναι το αναμενόμενο πλήθος αντικαταστάσεων του μαρκαδόρου σε όλους τους γύρους
- Σχηματίζοντας τον λόγο $f_m / f_O = \Sigma_i \left(f_i H_k + f_i\right) / \left((b_{\kappa+1} - b_1)/2 + \Sigma\, f_i/2\right)$, μετά από πράξεις, βγαίνει μικρότερος ή ίσος του $c \log n$, c σταθερά