Προγμένα Θέματα Αλγορίθμων

Διαλέξη 3: Αντισταθμιστική Ανάλυση Ι

Βασίλειος Νάκος

Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Πανεπιστήμιο Αθηνών

Συντομότερα Μονοπάτια ως Κινητήριο Παράδειγμα.

Δοθέντος γράφου $G(V, E)$ με μη αρνητικά βάρη w_e στις ακμές, να βρεθεί το συντομότερο μονοπάτι από ένα κόμβο s σε έναν κόμβο t.

Συντομότερα Μονοπάτια ως Κινητήριο Παράδειγμα.

Δοθέντος γράφου $G(V, E)$ με μη αρνητικά βάρη w_e στις ακμές, να βρεθεί το συντομότερο μονοπάτι από ένα κόμβο s σε έναν κόμβο t.

Αλγόριθμος Dijkstra

1: Διατήρησε ένα σύνολο S, αρχικοποιημένο στο $\{s\}$
2: Κράτησε πίνακα d για κάθε $u \in V$
3: $d_s \leftarrow 0$ και $d_u \leftarrow \infty, \forall u \in V \setminus \{s\}$ (εισαγωγή κόμβου)
4: $d_u \leftarrow w_{(s,v)}, \forall (s, v) \in E$
5: Όσο $t \notin S$
6: $\quad u^\star := \text{argmin}_{u \in \bar{S}} d_u$ (εξαγωγή ελαχίστου)
7: $\quad S \leftarrow S \cup \{u^\star\}$
8: $\quad$ Για κάθε v γείτονα του $u^\star$
9: $\quad\quad d_v \leftarrow \min\{d_v, d_{u^\star} + w_{(u^\star,v)}\}$ (μείωση κλειδιού)
10: Επέστρεψε το d_t

Συντομότερα Μονοπάτια ως Κινητήριο Παράδειγμα.

Αλγόριθμος Dijkstra

1: Διατήρησε ένα σύνολο S, αρχικοποιημένο στο $\{s\}$
2: Κράτησε πίνακα d για κάθε $u \in V$
3: $d_s \leftarrow 0$ και $d_u \leftarrow \infty, \forall u \in V \setminus \{s\}$ (εισαγωγή κόμβου)
4: $d_u \leftarrow w_{(s,v)}, \forall (s,v) \in E$
5: Όσο $t \notin S$
6: $\quad u^\star := \text{argmin}_{u \in \bar{S}} d_u$ (εξαγωγή ελαχίστου)
7: $\quad S \leftarrow S \cup \{u^\star\}$
8: $\quad$ Για κάθε v γείτονα του $u^\star$
9: $\quad\quad d_v \leftarrow \min\{d_v, d_{u^\star} + w_{(u^\star,v)}\}$ (μείωση κλειδιού)
10: Επέστρεψε το d_t

Έστω $t_{\text{in}}, t_{\text{emin}}, t_{\text{decKey}}$ τα κόστη να γίνουν οι αντίστοιχες πράξεις.

Συντομότερα Μονοπάτια ως Κινητήριο Παράδειγμα.

Αλγόριθμος Dijkstra

1: Διατήρησε ένα σύνολο S, αρχικοποιημένο στο $\{s\}$
2: Κράτησε πίνακα d για κάθε $u \in V$
3: $d_s \leftarrow 0$ και $d_u \leftarrow \infty, \forall u \in V \setminus \{s\}$ (εισαγωγή κόμβου)
4: $d_u \leftarrow w_{(s,v)}, \forall (s,v) \in E$
5: Όσο $t \notin S$
6: $\quad u^\star := \text{argmin}_{u \in \bar{S}} d_u$ (εξαγωγή ελαχίστου)
7: $\quad S \leftarrow S \cup \{u^\star\}$
8: $\quad$ Για κάθε v γείτονα του $u^\star$
9: $\quad\quad d_v \leftarrow \min\{d_v, d_{u^\star} + w_{(u^\star,v)}\}$ (μείωση κλειδιού)
10: Επέστρεψε το d_t

Έστω $t_{\text{in}}, t_{\text{emin}}, t_{\text{decKey}}$ τα κόστη να γίνουν οι αντίστοιχες πράξεις.

Χρόνος εκτέλεσης: $O\left(n \cdot t_{\text{emin}} + n \cdot t_{\text{in}} + m \cdot t_{\text{decKey}}\right)$

Δομή Δεδομένων: σωρός.

Άρα ο χρόνος εκτέλεσης του αλγόριθμου εξαρτάται από τη δομή δεδομένων που θα χρησιμοποιήσουμε: πόσο γρήγορα υποστηρίζει τις τρεις πράξεις εισαγωγή, εξαγωγή ελαχίστου, μείωση κλειδιού.

Δομή Δεδομένων: σωρός.

Άρα ο χρόνος εκτέλεσης του αλγόριθμου εξαρτάται από τη δομή δεδομένων που θα χρησιμοποιήσουμε: πόσο γρήγορα υποστηρίζει τις τρεις πράξεις εισαγωγή, εξαγωγή ελαχίστου, μείωση κλειδιού.

👉 Δυαδικός σωρός: $t_{\text{emin}} = O(1), t_{\text{in}}, t_{\text{decKey}} = O(\log n)$.

Δομή Δεδομένων: σωρός.

Άρα ο χρόνος εκτέλεσης του αλγόριθμου εξαρτάται από τη δομή δεδομένων που θα χρησιμοποιήσουμε: πόσο γρήγορα υποστηρίζει τις τρεις πράξεις εισαγωγή, εξαγωγή ελαχίστου, μείωση κλειδιού.

👉 Δυαδικός σωρός: $t_{\text{emin}} = O(1), t_{\text{in}}, t_{\text{decKey}} = O(\log n)$.

Αυτό πετυχαίνει υλοποίηση Dijkstra σε χρόνο $O((n+m)\log n)$.

Δομή Δεδομένων: σωρός.

Άρα ο χρόνος εκτέλεσης του αλγόριθμου εξαρτάται από τη δομή δεδομένων που θα χρησιμοποιήσουμε: πόσο γρήγορα υποστηρίζει τις τρεις πράξεις εισαγωγή, εξαγωγή ελαχίστου, μείωση κλειδιού.

👉 Δυαδικός σωρός: $t_{\text{emin}} = O(1), t_{\text{in}}, t_{\text{decKey}} = O(\log n)$.

Αυτό πετυχαίνει υλοποίηση Dijkstra σε χρόνο $O((n+m)\log n)$. Αλλά μήπως μπορούμε καλύτερα; Μήπως μπορούμε $O(n \log n + m)$;

Δομή Δεδομένων: σωρός.

Άρα ο χρόνος εκτέλεσης του αλγόριθμου εξαρτάται από τη δομή δεδομένων που θα χρησιμοποιήσουμε: πόσο γρήγορα υποστηρίζει τις τρεις πράξεις εισαγωγή, εξαγωγή ελαχίστου, μείωση κλειδιού.

👉 Δυαδικός σωρός: $t_{\text{emin}} = O(1), t_{\text{in}}, t_{\text{decKey}} = O(\log n)$.

Αυτό πετυχαίνει υλοποίηση Dijkstra σε χρόνο $O((n+m)\log n)$. Αλλά μήπως μπορούμε καλύτερα; Μήπως μπορούμε $O(n \log n + m)$;

👁 Χαλάρωση απαίτησης: δεν χρειάζεται η πράξη t_{decKey} να γίνεται **πάντα** σε χρόνο $O(1)$,

Δομή Δεδομένων: σωρός.

Άρα ο χρόνος εκτέλεσης του αλγόριθμου εξαρτάται από τη δομή δεδομένων που θα χρησιμοποιήσουμε: πόσο γρήγορα υποστηρίζει τις τρεις πράξεις εισαγωγή, εξαγωγή ελαχίστου, μείωση κλειδιού.

👉 Δυαδικός σωρός: $t_{\text{emin}} = O(1), t_{\text{in}}, t_{\text{decKey}} = O(\log n)$.

Αυτό πετυχαίνει υλοποίηση Dijkstra σε χρόνο $O((n+m)\log n)$. Αλλά μήπως μπορούμε καλύτερα; Μήπως μπορούμε $O(n \log n + m)$;

👁 Χαλάρωση απαίτησης: δεν χρειάζεται η πράξη t_{decKey} να γίνεται **πάντα** σε χρόνο $O(1)$, αρκεί να γίνεται 'κατά μέρο όρο' σε χρόνο $O(1)$

Δομή Δεδομένων: σωρός.

Άρα ο χρόνος εκτέλεσης του αλγόριθμου εξαρτάται από τη δομή δεδομένων που θα χρησιμοποιήσουμε: πόσο γρήγορα υποστηρίζει τις τρεις πράξεις εισαγωγή, εξαγωγή ελαχίστου, μείωση κλειδιού.

👉 Δυαδικός σωρός: $t_{\text{emin}} = O(1), t_{\text{in}}, t_{\text{decKey}} = O(\log n)$.

Αυτό πετυχαίνει υλοποίηση Dijkstra σε χρόνο $O((n+m)\log n)$. Αλλά μήπως μπορούμε καλύτερα; Μήπως μπορούμε $O(n \log n + m)$;

👁 Χαλάρωση απαίτησης: δεν χρειάζεται η πράξη t_{decKey} να γίνεται **πάντα** σε χρόνο $O(1)$, αρκεί να γίνεται 'κατά μέρο όρο' σε χρόνο $O(1) \Rightarrow$ **συνολικός** χρόνος για m πράξεις να είναι $O(m)$.

Δομή Δεδομένων: σωρός.

Άρα ο χρόνος εκτέλεσης του αλγόριθμου εξαρτάται από τη δομή δεδομένων που θα χρησιμοποιήσουμε: πόσο γρήγορα υποστηρίζει τις τρεις πράξεις εισαγωγή, εξαγωγή ελαχίστου, μείωση κλειδιού.

👉 Δυαδικός σωρός: $t_{\text{emin}} = O(1), t_{\text{in}}, t_{\text{decKey}} = O(\log n)$.

Αυτό πετυχαίνει υλοποίηση Dijkstra σε χρόνο $O((n+m)\log n)$. Αλλά μήπως μπορούμε καλύτερα; Μήπως μπορούμε $O(n \log n + m)$;

👁 Χαλάρωση απαίτησης: δεν χρειάζεται η πράξη t_{decKey} να γίνεται **πάντα** σε χρόνο $O(1)$, αρκεί να γίνεται 'κατά μέρο όρο' σε χρόνο $O(1) \Rightarrow$ **συνολικός** χρόνος για m πράξεις να είναι $O(m)$.

Αντισταθμιστική ανάλυση.

Ένα πιο απλό παράδειγμα.

Πριν σχεδιάσουμε τη ζητούμενη δομή, τον διάσημο σωρό Fibonacci, θα δούμε μερικά πιο απλά παραδείγματα.

Ένα πιο απλό παράδειγμα.

Πριν σχεδιάσουμε τη ζητούμενη δομή, τον διάσημο σωρό Fibonacci, θα δούμε μερικά πιο απλά παραδείγματα.

Πρόβλημα του δυναμικά μεταβαλλόμενου πίνακα: Θέλουμε να υλοποιήσουμε μια δομή δεδομένων που υποστηρίζει τις βασικές λειτουργίες ενός πίνακα — δηλαδή τυχαία πρόσβαση στα στοιχεία
-

Ένα πιο απλό παράδειγμα.

Πριν σχεδιάσουμε τη ζητούμενη δομή, τον διάσημο σωρό Fibonacci, θα δούμε μερικά πιο απλά παραδείγματα.

Πρόβλημα του δυναμικά μεταβαλλόμενου πίνακα: Θέλουμε να υλοποιήσουμε μια δομή δεδομένων που υποστηρίζει τις βασικές λειτουργίες ενός πίνακα — δηλαδή τυχαία πρόσβαση στα στοιχεία -αλλά ταυτόχρονα

Ένα πιο απλό παράδειγμα.

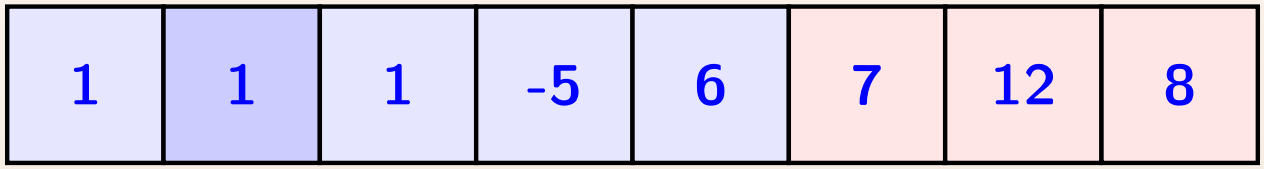
Πριν σχεδιάσουμε τη ζητούμενη δομή, τον διάσημο σωρό Fibonacci, θα δούμε μερικά πιο απλά παραδείγματα.

Πρόβλημα του δυναμικά μεταβαλλόμενου πίνακα: Θέλουμε να υλοποιήσουμε μια δομή δεδομένων που υποστηρίζει τις βασικές λειτουργίες ενός πίνακα — δηλαδή τυχαία πρόσβαση στα στοιχεία -αλλά ταυτόχρονα

Ένα πιο απλό παράδειγμα.

Πριν σχεδιάσουμε τη ζητούμενη δομή, τον διάσημο σωρό Fibonacci, θα δούμε μερικά πιο απλά παραδείγματα.

Πρόβλημα του δυναμικά μεταβαλλόμενου πίνακα: Θέλουμε να υλοποιήσουμε μια δομή δεδομένων που υποστηρίζει τις βασικές λειτουργίες ενός πίνακα — δηλαδή τυχαία πρόσβαση στα στοιχεία -αλλά ταυτόχρονα



1. Να προσθέτει (ή να αφαιρεί) ένα στοιχείο κάθε φορά
2. Να κάνει αποδοτικές ανακατανομές μνήμες.

Δυναμικά μεταβαλλόμενος πίνακας με εισαγωγή.

Ας δούμε την περίπτωση όπου έχουμε μόνο εισαγωγή στοιχείου και όχι διαγραφή.

1: Διατηρούμε το τρέχον μέγεθος του πίνακα στο size
2: Διατηρούμε το πόσα στοιχεία έχουν έρθει στο num.
3: Αρχικά $\text{size} = 1, \text{num} = 0$
4: Όταν έρχεται ένα νέο στοιχείο i
5: $\quad A[\text{num}] = i$
6: $\quad \text{num} = \text{num} + 1$
7: $\quad$ Αν $\text{num} = \text{size}$
8: $\quad\quad$ Ξαναφτιάξε τον A από την αρχή με μέγεθος $\text{size} + 1$.

Κακός αλγόριθμος:

Δυναμικά μεταβαλλόμενος πίνακας με εισαγωγή.

Ας δούμε την περίπτωση όπου έχουμε μόνο εισαγωγή στοιχείου και όχι διαγραφή.

1: Διατηρούμε το τρέχον μέγεθος του πίνακα στο size
2: Διατηρούμε το πόσα στοιχεία έχουν έρθει στο num.
3: Αρχικά $\text{size} = 1, \text{num} = 0$
4: Όταν έρχεται ένα νέο στοιχείο i
5: $\quad A[\text{num}] = i$
6: $\quad \text{num} = \text{num} + 1$
7: $\quad$ Αν $\text{num} = \text{size}$
8: $\quad\quad$ Ξαναφτιάξε τον A από την αρχή με μέγεθος $\text{size} + 1$.

Κακός αλγόριθμος: Κάθε νέα εισαγωγή στοιχείου ξαναφτιάχνει τον A από την αρχή, άρα $1 + 2 + \ldots + n = \Theta(n^2)$ χρόνος.

Δυναμικά μεταβαλλόμενος πίνακας με εισαγωγή.

1: Διατηρούμε το τρέχον μέγεθος του πίνακα στο size
2: Διατηρούμε το πόσα στοιχεία έχουν έρθει στο num.
3: Αρχικά $\mathrm{size} = 1, \mathrm{num} = 0$
4: Όταν έρχεται ένα νέο στοιχείο i
5: $\quad A[\mathrm{num}] = i$
6: $\quad \mathrm{num} = \mathrm{num} + 1$
7: $\quad$ Αν $\mathrm{num} = \mathrm{size}$
8: $\quad\quad$ Ξαναφτιάξε τον A από την αρχή με διπλάσιο μέγεθος

👉 Τώρα υπάρει χώρος:

Δυναμικά μεταβαλλόμενος πίνακας με εισαγωγή.

1: Διατηρούμε το τρέχον μέγεθος του πίνακα στο size
2: Διατηρούμε το πόσα στοιχεία έχουν έρθει στο num.
3: Αρχικά $\mathrm{size} = 1, \mathrm{num} = 0$
4: Όταν έρχεται ένα νέο στοιχείο i
5: $\quad A[\mathrm{num}] = i$
6: $\quad \mathrm{num} = \mathrm{num} + 1$
7: $\quad$ Αν $\mathrm{num} = \mathrm{size}$
8: $\quad\quad$ Ξαναφτιάξε τον A από την αρχή με διπλάσιο μέγεθος

👉 Τώρα υπάρει χώρος: πλέον ο αλγόριθμος πληρώνει πολύ πιο σπάνια: μόνο κάθε φορά που το μέγεθος του A μεγαλώνει από 2^ℓ σε $2^{\ell+1}$.

Δυναμικά μεταβαλλόμενος πίνακας με εισαγωγή.

1: Διατηρούμε το τρέχον μέγεθος του πίνακα στο size
2: Διατηρούμε το πόσα στοιχεία έχουν έρθει στο num.
3: Αρχικά $\mathrm{size} = 1, \mathrm{num} = 0$
4: Όταν έρχεται ένα νέο στοιχείο i
5: $\quad A[\mathrm{num}] = i$
6: $\quad \mathrm{num} = \mathrm{num} + 1$
7: $\quad$ Αν $\mathrm{num} = \mathrm{size}$
8: $\quad\quad$ Ξαναφτιάξε τον A από την αρχή με διπλάσιο μέγεθος

👉 Τώρα υπάρει χώρος: πλέον ο αλγόριθμος πληρώνει πολύ πιο σπάνια: μόνο κάθε φορά που το μέγεθος του A μεγαλώνει από 2^ℓ σε $2^{\ell+1}$.
Όποτε δεν μεγαλώνει πληρώνει 1.

Δυναμικά μεταβαλλόμενος πίνακας με εισαγωγή.

1: Διατηρούμε το τρέχον μέγεθος του πίνακα στο size
2: Διατηρούμε το πόσα στοιχεία έχουν έρθει στο num.
3: Αρχικά $\mathrm{size} = 1, \mathrm{num} = 0$
4: Όταν έρχεται ένα νέο στοιχείο i
5: $\quad A[\mathrm{num}] = i$
6: $\quad \mathrm{num} = \mathrm{num} + 1$
7: $\quad$ Αν $\mathrm{num} = \mathrm{size}$
8: $\quad\quad$ Ξαναφτιάξε τον A από την αρχή με διπλάσιο μέγεθος

👉 Τώρα υπάρει χώρος: πλέον ο αλγόριθμος πληρώνει πολύ πιο σπάνια: μόνο κάθε φορά που το μέγεθος του A μεγαλώνει από 2^ℓ σε $2^{\ell+1}$. Όποτε δεν μεγαλώνει πληρώνει 1. Συνολικό κόστος για n πράξεις

$$\underbrace{1+1+\ldots+1}_{n}+\sum_{\ell:0\leq\ell\leq\log(2n)} 2^\ell$$

Δυναμικά μεταβαλλόμενος πίνακας με εισαγωγή.

1: Διατηρούμε το τρέχον μέγεθος του πίνακα στο size
2: Διατηρούμε το πόσα στοιχεία έχουν έρθει στο num.
3: Αρχικά $\mathrm{size} = 1, \mathrm{num} = 0$
4: Όταν έρχεται ένα νέο στοιχείο i
5: $\quad A[\mathrm{num}] = i$
6: $\quad \mathrm{num} = \mathrm{num} + 1$
7: $\quad$ Αν $\mathrm{num} = \mathrm{size}$
8: $\quad\quad$ Ξαναφτιάξε τον A από την αρχή με διπλάσιο μέγεθος

👉 Τώρα υπάρει χώρος: πλέον ο αλγόριθμος πληρώνει πολύ πιο σπάνια: μόνο κάθε φορά που το μέγεθος του A μεγαλώνει από 2^ℓ σε $2^{\ell+1}$. Όποτε δεν μεγαλώνει πληρώνει 1. Συνολικό κόστος για n πράξεις

$$\underbrace{1+1+\ldots+1}_{n}+\sum_{\ell:0\leq\ell\leq\log(2n)} 2^\ell = O(n).$$

Εναλλακτική Ανάλυση.

1: Αρχικά $\text{size} = 1, \text{num} = 0$
2: Όταν έρχεται ένα νέο στοιχείο i
3: $\quad A[\text{num}] = i$
4: $\quad \text{num} = \text{num} + 1$
5: $\quad$ Αν $\text{num} = \text{size}$
6: $\quad\quad$ Ξαναφτιάξε τον A από την αρχή με διπλάσιο μέγεθος

Στο διάστημα $[2^\ell + 1, 2^{\ell+1}]$ υπάρχουν $2^{\ell+1} - (2^\ell + 1) + 1 = 2^\ell$ αριθμοί.

Εναλλακτική Ανάλυση.

1: Αρχικά $\text{size} = 1, \text{num} = 0$
2: Όταν έρχεται ένα νέο στοιχείο i
3: $\quad A[\text{num}] = i$
4: $\quad \text{num} = \text{num} + 1$
5: $\quad$ Αν $\text{num} = \text{size}$
6: $\quad\quad$ Ξαναφτιάξε τον A από την αρχή με διπλάσιο μέγεθος

Στο διάστημα $[2^\ell + 1, 2^{\ell+1}]$ υπάρχουν $2^{\ell+1} - (2^\ell + 1) + 1 = 2^\ell$ αριθμοί.

👁 Μπορούμε να χρεώσουμε την *ακριβή* πράξη ανακατανομής με κόστος $O(2^\ell)$ σε 2^ℓ πράξεις κόστους $O(1)$ η καθεμία, οι οποίες μεσολάβησαν μεταξύ της τρέχουσας και της προηγούμενης ανακατανομής.

Εναλλακτική Ανάλυση.

1: Αρχικά $\text{size} = 1, \text{num} = 0$
2: Όταν έρχεται ένα νέο στοιχείο i
3: $\quad A[\text{num}] = i$
4: $\quad \text{num} = \text{num} + 1$
5: $\quad$ Αν $\text{num} = \text{size}$
6: $\quad\quad$ Ξαναφτιάξε τον A από την αρχή με διπλάσιο μέγεθος

Στο διάστημα $[2^\ell + 1, 2^{\ell+1}]$ υπάρχουν $2^{\ell+1} - (2^\ell + 1) + 1 = 2^\ell$ αριθμοί.

👁 Μπορούμε να χρεώσουμε την *ακριβή* πράξη ανακατανομής με κόστος $O(2^\ell)$ σε 2^ℓ πράξεις κόστους $O(1)$ η καθεμία, οι οποίες μεσολάβησαν μεταξύ της τρέχουσας και της προηγούμενης ανακατανομής.

👉 Αντισταθμιστική ανάλυση: Ουδέν κακό αμιγές καλού ή 'ένα κακό μπορεί να χρεωθεί σε πολλά καλά'.

Ανάλυση μέσω συνάρτησης δυναμικού.

👉 Έστω ότι έχουμε μία συνάρτηση $\Phi(t)$

Ανάλυση μέσω συνάρτησης δυναμικού.

👉 Έστω ότι έχουμε μία συνάρτηση $\Phi(t)$ η οποία επιστρέφει έναν αριθμό σαν *αξιολόγηση* της κατάστασης της δομής δεδομένων τη στιγμή t.

Ανάλυση μέσω συνάρτησης δυναμικού.

👉 Έστω ότι έχουμε μία συνάρτηση $\Phi(t)$ η οποία επιστρέφει έναν αριθμό σαν *αξιολόγηση* της κατάστασης της δομής δεδομένων τη στιγμή t.

- Αν τη στιγμή t η δομή (ή ο αλγόριθμος) πληρώνει c_t ορίζουμε το *αντισταθμιστικό κόστος* $\hat{c}_t = c_t + \Phi(t) - \Phi(t-1)$

1. Απαιτούμε $\Phi(1) = 0$ (ή $\Phi(0) = 0$).
2. Συνηθώς προτιμούμε ένα καλό *ομοιόμορφο* φράγμα στο $\hat{c}_t$,

Ανάλυση μέσω συνάρτησης δυναμικού.

👉 Έστω ότι έχουμε μία συνάρτηση $\Phi(t)$ η οποία επιστρέφει έναν αριθμό σαν *αξιολόγηση* της κατάστασης της δομής δεδομένων τη στιγμή t.

- Αν τη στιγμή t η δομή (ή ο αλγόριθμος) πληρώνει c_t ορίζουμε το *αντισταθμιστικό κόστος* $\hat{c}_t = c_t + \Phi(t) - \Phi(t-1)$

1. Απαιτούμε $\Phi(1) = 0$ (ή $\Phi(0) = 0$).
2. Συνηθώς προτιμούμε ένα καλό *ομοιόμορφο* φράγμα στο $\hat{c}_t$, δηλαδή $\hat{c}_t \leq L$ για κάθε t.
3. $\sum_{t=1}^{n} \hat{c}_t = \sum_{t=1}^{n} c_t + \Phi(n) - \Phi(0)$

Ανάλυση μέσω συνάρτησης δυναμικού.

👉 Έστω ότι έχουμε μία συνάρτηση $\Phi(t)$ η οποία επιστρέφει έναν αριθμό σαν *αξιολόγηση* της κατάστασης της δομής δεδομένων τη στιγμή t.

- Αν τη στιγμή t η δομή (ή ο αλγόριθμος) πληρώνει c_t ορίζουμε το *αντισταθμιστικό κόστος* $\hat{c}_t = c_t + \Phi(t) - \Phi(t-1)$

1. Απαιτούμε $\Phi(1) = 0$ (ή $\Phi(0) = 0$).
2. Συνηθώς προτιμούμε ένα καλό *ομοιόμορφο* φράγμα στο $\hat{c}_t$, δηλαδή $\hat{c}_t \leq L$ για κάθε t.
3. $\sum_{t=1}^{n} \hat{c}_t = \sum_{t=1}^{n} c_t + \Phi(n) - \Phi(0)$
4. Μπορούμε να πάρουμε $\sum_{t=1}^{n} c_t + \Phi(n) \leq L \cdot n$

Ανάλυση μέσω συνάρτησης δυναμικού.

👉 Έστω ότι έχουμε μία συνάρτηση $\Phi(t)$ η οποία επιστρέφει έναν αριθμό σαν *αξιολόγηση* της κατάστασης της δομής δεδομένων τη στιγμή t.

- Αν τη στιγμή t η δομή (ή ο αλγόριθμος) πληρώνει c_t ορίζουμε το *αντισταθμιστικό κόστος* $\hat{c}_t = c_t + \Phi(t) - \Phi(t-1)$

1. Απαιτούμε $\Phi(1) = 0$ (ή $\Phi(0) = 0$).
2. Συνηθώς προτιμούμε ένα καλό *ομοιόμορφο* φράγμα στο $\hat{c}_t$, δηλαδή $\hat{c}_t \leq L$ για κάθε t.
3. $\sum_{t=1}^{n} \hat{c}_t = \sum_{t=1}^{n} c_t + \Phi(n) - \Phi(0)$
4. Μπορούμε να πάρουμε $\sum_{t=1}^{n} c_t + \Phi(n) \leq L \cdot n \Rightarrow \sum_{t=1}^{n} c_t \leq \ldots$.

★ Όλη η μαγεία βρίσκεται στην εύρεση της Φ για την αντισταθμιστική ανάλυση ενός αλγόριθμου (δομής).

Ανάλυση μέσω συνάρτησης δυναμικού.

Έστω num_t το πλήθος των στοιχείων και size_t το μέγεθος του πίνακα μετά την εισαγωγή του στοιχείου t.

Ανάλυση μέσω συνάρτησης δυναμικού.

Έστω num_t το πλήθος των στοιχείων και size_t το μέγεθος του πίνακα μετά την εισαγωγή του στοιχείου t. Ορίζουμε τη συνάρτηση $\Phi(t) = 4 \cdot \mathrm{num}_t - 2\mathrm{size}_t$.

Ανάλυση μέσω συνάρτησης δυναμικού.

Έστω num_t το πλήθος των στοιχείων και size_t το μέγεθος του πίνακα μετά την εισαγωγή του στοιχείου t. Ορίζουμε τη συνάρτηση $\Phi(t) = 4 \cdot \mathrm{num}_t - 2\mathrm{size}_t$.

Έχουμε $\Phi(1) = 0$ και
$\hat{c}_t = c_t + \Phi(t) - \Phi(t-1) = 4\left(\mathrm{num}_t - \mathrm{num}_{t-1}\right) - 2\left(\mathrm{size}_t - \mathrm{size}_{t-1}\right)$.

Ανάλυση μέσω συνάρτησης δυναμικού.

Έστω num_t το πλήθος των στοιχείων και size_t το μέγεθος του πίνακα μετά την εισαγωγή του στοιχείου t. Ορίζουμε τη συνάρτηση $\Phi(t) = 4 \cdot \text{num}_t - 2\text{size}_t$.

Έχουμε $\Phi(1) = 0$ και
$\hat{c}_t = c_t + \Phi(t) - \Phi(t-1) = 4\left(\text{num}_t - \text{num}_{t-1}\right) - 2\left(\text{size}_t - \text{size}_{t-1}\right)$.

1. Αν η εισαγωγή δεν οδήγησε σε επαναδημιουργία του $A \Rightarrow$ τότε $\hat{c}_t = 1 + 4 \cdot 1 + 0 = 4$.

Ανάλυση μέσω συνάρτησης δυναμικού.

Έστω num_t το πλήθος των στοιχείων και size_t το μέγεθος του πίνακα μετά την εισαγωγή του στοιχείου t. Ορίζουμε τη συνάρτηση $\Phi(t) = 4 \cdot \text{num}_t - 2\text{size}_t$.

Έχουμε $\Phi(1) = 0$ και
$\hat{c}_t = c_t + \Phi(t) - \Phi(t-1) = 4\left(\text{num}_t - \text{num}_{t-1}\right) - 2\left(\text{size}_t - \text{size}_{t-1}\right)$.

1. Αν η εισαγωγή δεν οδήγησε σε επαναδημιουργία του $A \Rightarrow$ τότε $\hat{c}_t = 1 + 4 \cdot 1 + 0 = 4$.
2. Αν η εισαγωγή οδήγησε σε επαναδημιουργία του A τότε

Ανάλυση μέσω συνάρτησης δυναμικού.

Έστω num_t το πλήθος των στοιχείων και size_t το μέγεθος του πίνακα μετά την εισαγωγή του στοιχείου t. Ορίζουμε τη συνάρτηση $\Phi(t) = 4 \cdot \text{num}_t - 2\text{size}_t$.

Έχουμε $\Phi(1) = 0$ και
$\hat{c}_t = c_t + \Phi(t) - \Phi(t-1) = 4\left(\text{num}_t - \text{num}_{t-1}\right) - 2\left(\text{size}_t - \text{size}_{t-1}\right)$.

1. Αν η εισαγωγή δεν οδήγησε σε επαναδημιουργία του $A \Rightarrow$ τότε $\hat{c}_t = 1 + 4 \cdot 1 + 0 = 4$.
2. Αν η εισαγωγή οδήγησε σε επαναδημιουργία του A τότε $\hat{c}_t = \text{size}_t + 4 \cdot 1 - 2 \cdot (\text{size}_t - \text{size}_{t-1}) = \underbrace{2\text{size}_{t-1} - \text{size}_t}_{0} + 4 = 4$

Άρα $\hat{c}_t \leq 4 \Rightarrow$

Ανάλυση μέσω συνάρτησης δυναμικού.

Έστω num_t το πλήθος των στοιχείων και size_t το μέγεθος του πίνακα μετά την εισαγωγή του στοιχείου t. Ορίζουμε τη συνάρτηση $\Phi(t) = 4 \cdot \text{num}_t - 2\text{size}_t$.

Έχουμε $\Phi(1) = 0$ και
$\hat{c}_t = c_t + \Phi(t) - \Phi(t-1) = 4\left(\text{num}_t - \text{num}_{t-1}\right) - 2\left(\text{size}_t - \text{size}_{t-1}\right).$

1. Αν η εισαγωγή δεν οδήγησε σε επαναδημιουργία του $A \Rightarrow$ τότε $\hat{c}_t = 1 + 4 \cdot 1 + 0 = 4.$
2. Αν η εισαγωγή οδήγησε σε επαναδημιουργία του A τότε $\hat{c}_t = \text{size}_t + 4 \cdot 1 - 2 \cdot (\text{size}_t - \text{size}_{t-1}) = \underbrace{2\text{size}_{t-1} - \text{size}_t}_{0} + 4 = 4$

Άρα $\hat{c}_t \leq 4 \Rightarrow \sum_{t=1}^{n} c_t \leq \sum_{t=1}^{n} \hat{c}_t - \Phi(t)$

Ανάλυση μέσω συνάρτησης δυναμικού.

Έστω num_t το πλήθος των στοιχείων και size_t το μέγεθος του πίνακα μετά την εισαγωγή του στοιχείου t. Ορίζουμε τη συνάρτηση $\Phi(t) = 4 \cdot \text{num}_t - 2\text{size}_t$.

Έχουμε $\Phi(1) = 0$ και
$\hat{c}_t = c_t + \Phi(t) - \Phi(t-1) = 4\left(\text{num}_t - \text{num}_{t-1}\right) - 2\left(\text{size}_t - \text{size}_{t-1}\right).$

1. Αν η εισαγωγή δεν οδήγησε σε επαναδημιουργία του $A \Rightarrow$ τότε $\hat{c}_t = 1 + 4 \cdot 1 + 0 = 4.$
2. Αν η εισαγωγή οδήγησε σε επαναδημιουργία του A τότε $\hat{c}_t = \text{size}_t + 4 \cdot 1 - 2 \cdot (\text{size}_t - \text{size}_{t-1}) = \underbrace{2\text{size}_{t-1} - \text{size}_t}_{0} + 4 = 4$

Άρα $\hat{c}_t \leq 4 \Rightarrow \sum_{t=1}^{n} c_t \leq \sum_{t=1}^{n} \hat{c}_t - \Phi(t) \leq 4n - 2n = 2n.$✓

Δυναμικά μεταβαλλόμενος πίνακας με εισαγωγή και διαγραφή.

1: Διατηρούμε το τρέχον μέγεθος του πίνακα στο size
2: Διατηρούμε το πόσα στοιχεία έχουν έρθει στο num.
3: Αρχικά $\mathrm{size} = 1, \mathrm{num} = 0$
4: Όταν έρχεται εισαγωγή νέου στοιχείο i κάνε ό,τι και πριν.
5: Όταν απαιτείται διαγραφή (από το τέλος)
6: $\quad \mathrm{num} = \mathrm{num} - 1$
7: $\quad$ Αν $\mathrm{num} = \frac{1}{2}\mathrm{size} - 1$
8: $\quad$ Ξαναφτιάξε τον A από την αρχή με μέγεθος $\frac{1}{2}\mathrm{size}$.

Τι συμβαίνει με αυτόν τον αλγόριθμο;

Δυναμικά μεταβαλλόμενος πίνακας με εισαγωγή και διαγραφή.

1: Διατηρούμε το τρέχον μέγεθος του πίνακα στο size
2: Διατηρούμε το πόσα στοιχεία έχουν έρθει στο num.
3: Αρχικά $\mathrm{size} = 1, \mathrm{num} = 0$
4: Όταν έρχεται εισαγωγή νέου στοιχείο i κάνε ό,τι και πριν.
5: Όταν απαιτείται διαγραφή (από το τέλος)
6: $\quad \mathrm{num} = \mathrm{num} - 1$
7: $\quad$ Αν $\mathrm{num} = \frac{1}{2}\mathrm{size} - 1$
8: $\quad$ Ξαναφτιάξε τον A από την αρχή με μέγεθος $\frac{1}{2}\mathrm{size}$.

Τι συμβαίνει με αυτόν τον αλγόριθμο; Κακός: Μπορεί να γίνουν $\frac{n}{2}$ εισαγωγές,

Δυναμικά μεταβαλλόμενος πίνακας με εισαγωγή και διαγραφή.

1: Διατηρούμε το τρέχον μέγεθος του πίνακα στο size
2: Διατηρούμε το πόσα στοιχεία έχουν έρθει στο num.
3: Αρχικά $\mathrm{size} = 1, \mathrm{num} = 0$
4: Όταν έρχεται εισαγωγή νέου στοιχείο i κάνε ό,τι και πριν.
5: Όταν απαιτείται διαγραφή (από το τέλος)
6: $\quad \mathrm{num} = \mathrm{num} - 1$
7: $\quad$ Αν $\mathrm{num} = \frac{1}{2}\mathrm{size} - 1$
8: $\quad$ Ξαναφτιάξε τον A από την αρχή με μέγεθος $\frac{1}{2}\mathrm{size}$.

Τι συμβαίνει με αυτόν τον αλγόριθμο; Κακός: Μπορεί να γίνουν $\frac{n}{2}$ εισαγωγές, και οι υπόλοιπες πράξεις να είναι εναλλάξ διαγραφές/εισαγωγές:

Δυναμικά μεταβαλλόμενος πίνακας με εισαγωγή και διαγραφή.

1: Διατηρούμε το τρέχον μέγεθος του πίνακα στο size
2: Διατηρούμε το πόσα στοιχεία έχουν έρθει στο num.
3: Αρχικά $\mathrm{size} = 1, \mathrm{num} = 0$
4: Όταν έρχεται εισαγωγή νέου στοιχείο i κάνε ό,τι και πριν.
5: Όταν απαιτείται διαγραφή (από το τέλος)
6: $\quad \mathrm{num} = \mathrm{num} - 1$
7: $\quad$ Αν $\mathrm{num} = \frac{1}{2}\mathrm{size} - 1$
8: $\quad$ Ξαναφτιάξε τον A από την αρχή με μέγεθος $\frac{1}{2}\mathrm{size}$.

Τι συμβαίνει με αυτόν τον αλγόριθμο; Κακός: Μπορεί να γίνουν $\frac{n}{2}$ εισαγωγές, και οι υπόλοιπες πράξεις να είναι εναλλάξ διαγραφές/εισαγωγές: σε κάθε βήμα ξαναφτιάχνεται ο πίνακας από την αρχή $\Rightarrow \Theta(n^2)$.

Δυναμικά μεταβαλλόμενος πίνακας με εισαγωγή και διαγραφή.

1: Διατηρούμε το τρέχον μέγεθος του πίνακα στο size
2: Διατηρούμε το πόσα στοιχεία έχουν έρθει στο num.
3: Αρχικά $\mathrm{size} = 1, \mathrm{num} = 0$
4: Όταν έρχεται εισαγωγή νέου στοιχείο i κάνε ό,τι και πριν.
5: Όταν απαιτείται διαγραφή (από το τέλος)
6: $\quad \mathrm{num} = \mathrm{num} - 1$
7: $\quad$ Αν $\mathrm{num} = \frac{1}{2}\mathrm{size} - 1$
8: $\quad$ Ξαναφτιάξε τον A από την αρχή με μέγεθος $\frac{3}{4}\mathrm{size}$.

Το Θέμα είναι ότι μπορεί να γίνονται εισαγωγές, διαγραφές, εισαγωγές.
Πως θα ελέγξουμε τον συνολικό χρόνο εκτέλεσης αν έχουμε n πράξεις;

Εποπτεία του αλγόριθμου.

Ας δούμε λίγο ένα πιθανό σενάριο μεταβολής της δυάδας $(\text{num}, \text{size})$

$(0, 1) \Rightarrow \ldots \Rightarrow$

Εποπτεία του αλγόριθμου.

Ας δούμε λίγο ένα πιθανό σενάριο μεταβολής της δυάδας $(\text{num}, \text{size})$

$(0, 1) \Rightarrow \ldots \Rightarrow (X-1, X) \Rightarrow$

Εποπτεία του αλγόριθμου.

Ας δούμε λίγο ένα πιθανό σενάριο μεταβολής της δυάδας $(\mathrm{num}, \mathrm{size})$

$$(0,1) \Rightarrow \ldots \Rightarrow (X-1, X) \Rightarrow \left(\frac{1}{2}X - 1, X\right) \Rightarrow$$

Εποπτεία του αλγόριθμου.

Ας δούμε λίγο ένα πιθανό σενάριο μεταβολής της δυάδας $(\text{num}, \text{size})$

$$(0,1) \Rightarrow \ldots \Rightarrow (X-1, X) \Rightarrow \left(\frac{1}{2}X - 1, X\right) \Rightarrow \left(\frac{1}{2}X, \frac{3}{4}X\right)$$

Εποπτεία του αλγόριθμου.

Ας δούμε λίγο ένα πιθανό σενάριο μεταβολής της δυάδας $(\text{num}, \text{size})$

$$(0,1) \Rightarrow \ldots \Rightarrow (X-1, X) \Rightarrow \left(\frac{1}{2}X - 1, X\right) \Rightarrow \left(\frac{1}{2}X, \frac{3}{4}X\right) \Rightarrow \left(\frac{3}{8}X, \frac{3}{4}X\right)$$

Εποπτεία του αλγόριθμου.

Ας δούμε λίγο ένα πιθανό σενάριο μεταβολής της δυάδας $(\text{num}, \text{size})$

$$(0,1) \Rightarrow \ldots \Rightarrow (X-1, X) \Rightarrow \left(\frac{1}{2}X - 1, X\right) \Rightarrow \left(\frac{1}{2}X, \frac{3}{4}X\right) \Rightarrow \left(\frac{3}{8}X, \frac{3}{4}X\right)$$

$$\Rightarrow \left(\frac{3}{8}X - 1, \frac{3}{4} \cdot \frac{3}{8}X\right)$$

Εποπτεία του αλγόριθμου.

Ας δούμε λίγο ένα πιθανό σενάριο μεταβολής της δυάδας $(\text{num}, \text{size})$

$$(0,1) \Rightarrow \ldots \Rightarrow (X-1, X) \Rightarrow \left(\frac{1}{2}X - 1, X\right) \Rightarrow \left(\frac{1}{2}X, \frac{3}{4}X\right) \Rightarrow \left(\frac{3}{8}X, \frac{3}{4}X\right)$$

$$\Rightarrow \left(\frac{3}{8}X - 1, \frac{3}{4} \cdot \frac{3}{8}X\right) \Rightarrow \left(\frac{3}{4} \cdot \frac{3}{8}X, \frac{3}{4} \cdot \frac{3}{8}X\right) \Rightarrow \ldots$$

Εποπτεία του αλγόριθμου.

Ας δούμε λίγο ένα πιθανό σενάριο μεταβολής της δυάδας $(\text{num}, \text{size})$

$$(0,1) \Rightarrow \ldots \Rightarrow (X-1, X) \Rightarrow \left(\frac{1}{2}X - 1, X\right) \Rightarrow \left(\frac{1}{2}X, \frac{3}{4}X\right) \Rightarrow \left(\frac{3}{8}X, \frac{3}{4}X\right)$$

$$\Rightarrow \left(\frac{3}{8}X - 1, \frac{3}{4} \cdot \frac{3}{8}X\right) \Rightarrow \left(\frac{3}{4} \cdot \frac{3}{8}X, \frac{3}{4} \cdot \frac{3}{8}X\right) \Rightarrow \ldots$$

- Ανάμεσα μεταξύ δύο οποιωνδήποτε δύο αναδομήσεων, πχ $\left(\frac{1}{2}X, \frac{3}{4}X\right) \Rightarrow \left(\frac{3}{8}X, \frac{3}{4}X\right)$

Εποπτεία του αλγόριθμου.

Ας δούμε λίγο ένα πιθανό σενάριο μεταβολής της δυάδας $(\text{num}, \text{size})$

$$(0,1) \Rightarrow \ldots \Rightarrow (X-1, X) \Rightarrow \left(\frac{1}{2}X - 1, X\right) \Rightarrow \left(\frac{1}{2}X, \frac{3}{4}X\right) \Rightarrow \left(\frac{3}{8}X, \frac{3}{4}X\right)$$

$$\Rightarrow \left(\frac{3}{8}X - 1, \frac{3}{4} \cdot \frac{3}{8}X\right) \Rightarrow \left(\frac{3}{4} \cdot \frac{3}{8}X, \frac{3}{4} \cdot \frac{3}{8}X\right) \Rightarrow \ldots$$

👁 Ανάμεσα μεταξύ δύο οποιωνδήποτε δύο αναδομήσεων, πχ $\left(\frac{1}{2}X, \frac{3}{4}X\right) \Rightarrow \left(\frac{3}{8}X, \frac{3}{4}X\right)$ έχουν μεσολαβήσει αρκετές διαγραφές και εισαγωγές:

Εποπτεία του αλγόριθμου.

Ας δούμε λίγο ένα πιθανό σενάριο μεταβολής της δυάδας $(\text{num}, \text{size})$

$$(0,1) \Rightarrow \ldots \Rightarrow (X-1, X) \Rightarrow \left(\frac{1}{2}X - 1, X\right) \Rightarrow \left(\frac{1}{2}X, \frac{3}{4}X\right) \Rightarrow \left(\frac{3}{8}X, \frac{3}{4}X\right)$$

$$\Rightarrow \left(\frac{3}{8}X - 1, \frac{3}{4} \cdot \frac{3}{8}X\right) \Rightarrow \left(\frac{3}{4} \cdot \frac{3}{8}X, \frac{3}{4} \cdot \frac{3}{8}X\right) \Rightarrow \ldots$$

👁 Ανάμεσα μεταξύ δύο οποιωνδήποτε δύο αναδομήσεων, πχ $\left(\frac{1}{2}X, \frac{3}{4}X\right) \Rightarrow \left(\frac{3}{8}X, \frac{3}{4}X\right)$ έχουν μεσολαβήσει αρκετές διαγραφές και εισαγωγές: υπάρχουν τουλάχιστον $\frac{1}{2}X - \frac{3}{8}X = \Omega(X)$ 'φτηνές ανανεώσεις'

Εποπτεία του αλγόριθμου.

Ας δούμε λίγο ένα πιθανό σενάριο μεταβολής της δυάδας $(\text{num}, \text{size})$

$$(0,1) \Rightarrow \ldots \Rightarrow (X-1, X) \Rightarrow \left(\frac{1}{2}X - 1, X\right) \Rightarrow \left(\frac{1}{2}X, \frac{3}{4}X\right) \Rightarrow \left(\frac{3}{8}X, \frac{3}{4}X\right)$$

$$\Rightarrow \left(\frac{3}{8}X - 1, \frac{3}{4} \cdot \frac{3}{8}X\right) \Rightarrow \left(\frac{3}{4} \cdot \frac{3}{8}X, \frac{3}{4} \cdot \frac{3}{8}X\right) \Rightarrow \ldots$$

👁 Ανάμεσα μεταξύ δύο οποιωνδήποτε δύο αναδομήσεων, πχ $\left(\frac{1}{2}X, \frac{3}{4}X\right) \Rightarrow \left(\frac{3}{8}X, \frac{3}{4}X\right)$ έχουν μεσολαβήσει αρκετές διαγραφές και εισαγωγές: υπάρχουν τουλάχιστον $\frac{1}{2}X - \frac{3}{8}X = \Omega(X)$ 'φτηνές ανανεώσεις' $\Rightarrow$ μπορώ να χρεώσω το κόστος της αναδόμησης σε αυτές τις ανανέωσεις.

Εποπτεία του αλγόριθμου.

Ας δούμε λίγο ένα πιθανό σενάριο μεταβολής της δυάδας $(\text{num}, \text{size})$

$$(0,1) \Rightarrow \ldots \Rightarrow (X-1, X) \Rightarrow \left(\frac{1}{2}X - 1, X\right) \Rightarrow \left(\frac{1}{2}X, \frac{3}{4}X\right) \Rightarrow \left(\frac{3}{8}X, \frac{3}{4}X\right)$$

$$\Rightarrow \left(\frac{3}{8}X - 1, \frac{3}{4} \cdot \frac{3}{8}X\right) \Rightarrow \left(\frac{3}{4} \cdot \frac{3}{8}X, \frac{3}{4} \cdot \frac{3}{8}X\right) \Rightarrow \ldots$$

👁 Ανάμεσα μεταξύ δύο οποιωνδήποτε δύο αναδομήσεων, πχ $\left(\frac{1}{2}X, \frac{3}{4}X\right) \Rightarrow \left(\frac{3}{8}X, \frac{3}{4}X\right)$ έχουν μεσολαβήσει αρκετές διαγραφές και εισαγωγές: υπάρχουν τουλάχιστον $\frac{1}{2}X - \frac{3}{8}X = \Omega(X)$ 'φτηνές ανανεώσεις' $\Rightarrow$ μπορώ να χρεώσω το κόστος της αναδόμησης σε αυτές τις ανανέωσεις.

👉 Γενικότερα: πλέον ανάμεσα μεταξύ οποιωνδήποτε δύο διαδοχικών αναδομήσεων του A μεσολαβούν αρκετές πράξεις ✓

Πρόβλημα Ανανέωσης Λίστας.

Πρόβλημα Ανανέωσης Λίστας

1. Ξεκινάμε με μία λίστα με τους αριθμούς $1, 2, \ldots, n$.
2. Κάθε στιγμή ένα αντικείμενο $i \in [n]$ ζητείται να προσπελαστεί. Ο χρόνος προσπέλασης είναι η *θέση* του στη λίστα.
3. Ο αλγόριθμος μπορεί να αναδιατάξει τη λύση εναλλάσσοντας διαδοχικά σημεία. Το κόστος εναλλαγής δύο σημείων είναι 1.

Θέλουμε να σχεδιάσουμε έναν αλγόριθμο *ελάχιστου* κόστους:

Πρόβλημα Ανανέωσης Λίστας.

Πρόβλημα Ανανέωσης Λίστας

1. Ξεκινάμε με μία λίστα με τους αριθμούς $1, 2, \ldots, n$.
2. Κάθε στιγμή ένα αντικείμενο $i \in [n]$ ζητείται να προσπελαστεί. Ο χρόνος προσπέλασης είναι η *θέση* του στη λίστα.
3. Ο αλγόριθμος μπορεί να αναδιατάξει τη λύση εναλλάσσοντας διαδοχικά σημεία. Το κόστος εναλλαγής δύο σημείων είναι 1.

Θέλουμε να σχεδιάσουμε έναν αλγόριθμο *ελάχιστου* κόστους: θα συγκριθούμε με τον βέλτιστο αλγόριθμο ο οποίος ξέρει *εκ των προτέρων* τη σειρά των αντικειμένων που θα ζητηθούν.

Πρόβλημα Ανανέωσης Λίστας.

Πρόβλημα Ανανέωσης Λίστας

1. Ξεκινάμε με μία λίστα με τους αριθμούς $1, 2, \ldots, n$.
2. Κάθε στιγμή ένα αντικείμενο $i \in [n]$ ζητείται να προσπελαστεί. Ο χρόνος προσπέλασης είναι η *θέση* του στη λίστα.
3. Ο αλγόριθμος μπορεί να αναδιατάξει τη λύση εναλλάσσοντας διαδοχικά σημεία. Το κόστος εναλλαγής δύο σημείων είναι 1.

Θέλουμε να σχεδιάσουμε έναν αλγόριθμο *ελάχιστου* κόστους: θα συγκριθούμε με τον βέλτιστο αλγόριθμο ο οποίος ξέρει *εκ των προτέρων* τη σειρά των αντικειμένων που θα ζητηθούν.

Έστω OPT αυτός ο αλγόριθμος, και έστω k το κόστος του.

Πρόβλημα Ανανέωσης Λίστας.

Αλγόριθμος Προτεραιοποίησης ALG

Κάθε φορά που ζητείται ένα στοιχείο $i \in [n]$, φέρε το πρώτο στη λίστα.

Πρόβλημα Ανανέωσης Λίστας.

Αλγόριθμος Προτεραιοποίησης ALG

Κάθε φορά που ζητείται ένα στοιχείο $i \in [n]$, φέρε το πρώτο στη λίστα.

Θεώρημα

Το κόστος του αλγορίθμου ALG *είναι το πολύ* $4k$.

Πρόβλημα Ανανέωσης Λίστας.

Αλγόριθμος Προτεραιοποίησης ALG

Κάθε φορά που ζητείται ένα στοιχείο $i \in [n]$, φέρε το πρώτο στη λίστα.

Θεώρημα

Το κόστος του αλγορίθμου ALG *είναι το πολύ* $4k$.

Σε κάθε βήμα είτε ο αλγόριθμος

1. είτε πληρώνει κάτι συγκρίσιμο με το τι πληρώνει ο OPT

Πρόβλημα Ανανέωσης Λίστας.

Αλγόριθμος Προτεραιοποίησης ALG

Κάθε φορά που ζητείται ένα στοιχείο $i \in [n]$, φέρε το πρώτο στη λίστα.

Θεώρημα

Το κόστος του αλγορίθμου ALG *είναι το πολύ* $4k$.

Σε κάθε βήμα είτε ο αλγόριθμος

1. είτε πληρώνει κάτι συγκρίσιμο με το τι πληρώνει ο OPT
2. είτε πληρώνει πολύ,

Πρόβλημα Ανανέωσης Λίστας.

Αλγόριθμος Προτεραιοποίησης ALG

Κάθε φορά που ζητείται ένα στοιχείο $i \in [n]$, φέρε το πρώτο στη λίστα.

Θεώρημα

Το κόστος του αλγορίθμου ALG *είναι το πολύ* $4k$.

Σε κάθε βήμα είτε ο αλγόριθμος

1. είτε πληρώνει κάτι συγκρίσιμο με το τι πληρώνει ο OPT
2. είτε πληρώνει πολύ, αλλά τότε κάπως πρέπει να δείξουμε ότι η λίστα του του έρχεται 'πιο κοντά' στη λίστα του OPT.

Πρόβλημα Ανανέωσης Λίστας.

Αλγόριθμος Προτεραιοποίησης ALG

Κάθε φορά που ζητείται ένα στοιχείο $i \in [n]$, φέρε το πρώτο στη λίστα.

Θεώρημα

Το κόστος του αλγορίθμου ALG *είναι το πολύ* $4k$.

Σε κάθε βήμα είτε ο αλγόριθμος

1. είτε πληρώνει κάτι συγκρίσιμο με το τι πληρώνει ο OPT
2. είτε πληρώνει πολύ, αλλά τότε κάπως πρέπει να δείξουμε ότι η λίστα του του έρχεται 'πιο κοντά' στη λίστα του OPT.

★ Θα χρειαστούμε μία συνάρτηση δυναμικού για να εκφράσουμε αυτό το 'πιο κοντά'.

Πρόβλημα Ανανέωσης Λίστας.

Αλγόριθμος Προτεραιοποίησης ALG

Κάθε φορά που ζητείται ένα στοιχείο $i \in [n]$, φέρε το πρώτο στη λίστα.

Θεώρημα

Το κόστος του αλγορίθμου ALG *είναι το πολύ* $4k$.

Σε κάθε βήμα είτε ο αλγόριθμος

1. είτε πληρώνει κάτι συγκρίσιμο με το τι πληρώνει ο OPT
2. είτε πληρώνει πολύ, αλλά τότε κάπως πρέπει να δείξουμε ότι η λίστα του του έρχεται 'πιο κοντά' στη λίστα του OPT.

★ Θα χρειαστούμε μία συνάρτηση δυναμικού για να εκφράσουμε αυτό το 'πιο κοντά'.

Επιλογή συνάρτησης δυναμικού.

$\star$ Ένα ζεύγος (i, j) λέγεται αντιστροφή για δύο πίνακες A, B αν $i < j$ (αντίστοιχα $i > j$) και η θέση που εμφανίζεται το i εμφανίζεται στην λίστα του A είναι **μεγαλύτερη** (αντίστοιχα **μικρότερη**) από την θέση μου εμφανίζεται το j στη θέση του B.

Επιλογή συνάρτησης δυναμικού.

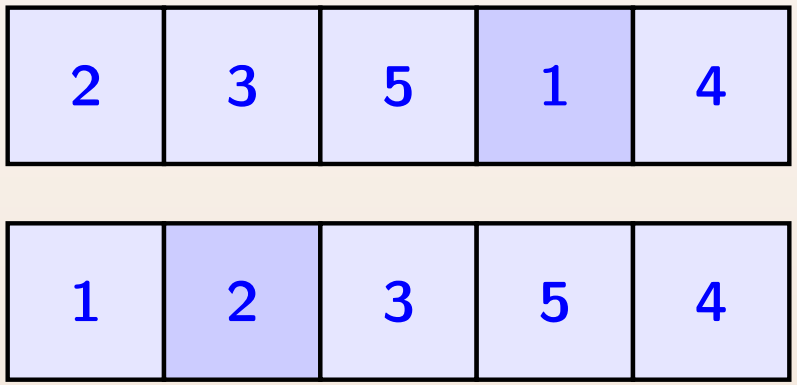
$\star$ Ένα ζεύγος (i, j) λέγεται αντιστροφή για δύο πίνακες A, B αν $i < j$ (αντίστοιχα $i > j$) και η θέση που εμφανίζεται το i εμφανίζεται στην λίστα του A είναι **μεγαλύτερη** (αντίστοιχα **μικρότερη**) από την θέση μου εμφανίζεται το j στη θέση του B.

$\Phi(t) = 2$ (#αντιστροφών μεταξύ της λίστας του ALG και του OPT τη στιγμή t).

Επιλογή συνάρτησης δυναμικού.

$\star$ Ένα ζεύγος (i, j) λέγεται αντιστροφή για δύο πίνακες A, B αν $i < j$ (αντίστοιχα $i > j$) και η θέση που εμφανίζεται το i εμφανίζεται στην λίστα του A είναι **μεγαλύτερη** (αντίστοιχα **μικρότερη**) από την θέση μου εμφανίζεται το j στη θέση του B.

$\Phi(t) = 2$ (#αντιστροφών μεταξύ της λίστας του ALG και του OPT τη στιγμή t).

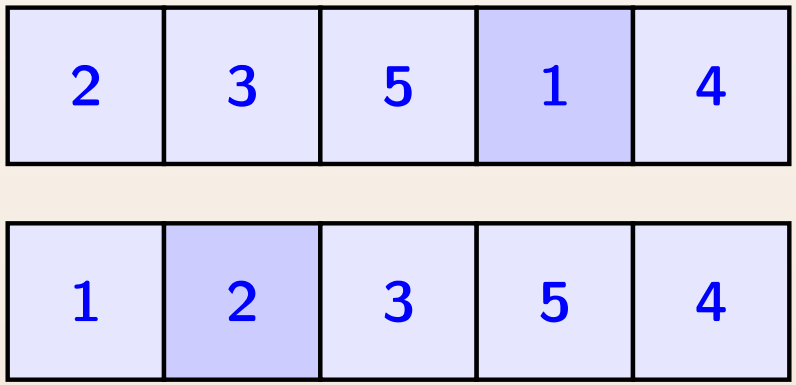


Το $(1, 2)$ είναι αντιστροφή.

Επιλογή συνάρτησης δυναμικού.

$\star$ Ένα ζεύγος (i, j) λέγεται αντιστροφή για δύο πίνακες A, B αν $i < j$ (αντίστοιχα $i > j$) και η θέση που εμφανίζεται το i εμφανίζεται στην λίστα του A είναι **μεγαλύτερη** (αντίστοιχα **μικρότερη**) από την θέση μου εμφανίζεται το j στη θέση του B.

$\Phi(t) = 2$ (#αντιστροφών μεταξύ της λίστας του ALG και του OPT τη στιγμή t).

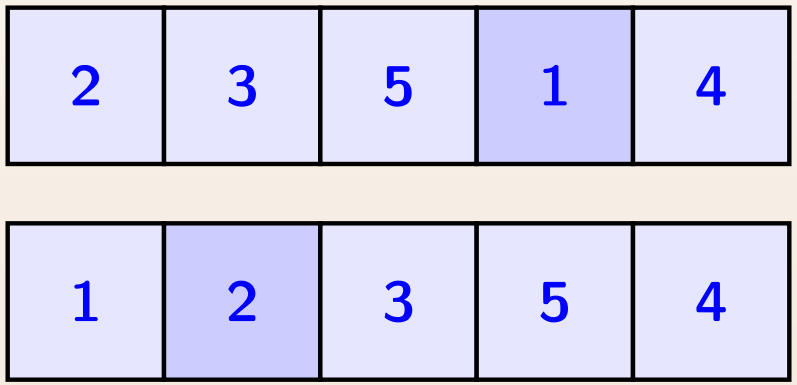


Το $(1, 2)$ είναι αντιστροφή. Όπως και το $(5, 4)$.

Επιλογή συνάρτησης δυναμικού.

$\star$ Ένα ζεύγος (i, j) λέγεται αντιστροφή για δύο πίνακες A, B αν $i < j$ (αντίστοιχα $i > j$) και η θέση που εμφανίζεται το i εμφανίζεται στην λίστα του A είναι **μεγαλύτερη** (αντίστοιχα **μικρότερη**) από την θέση μου εμφανίζεται το j στη θέση του B.

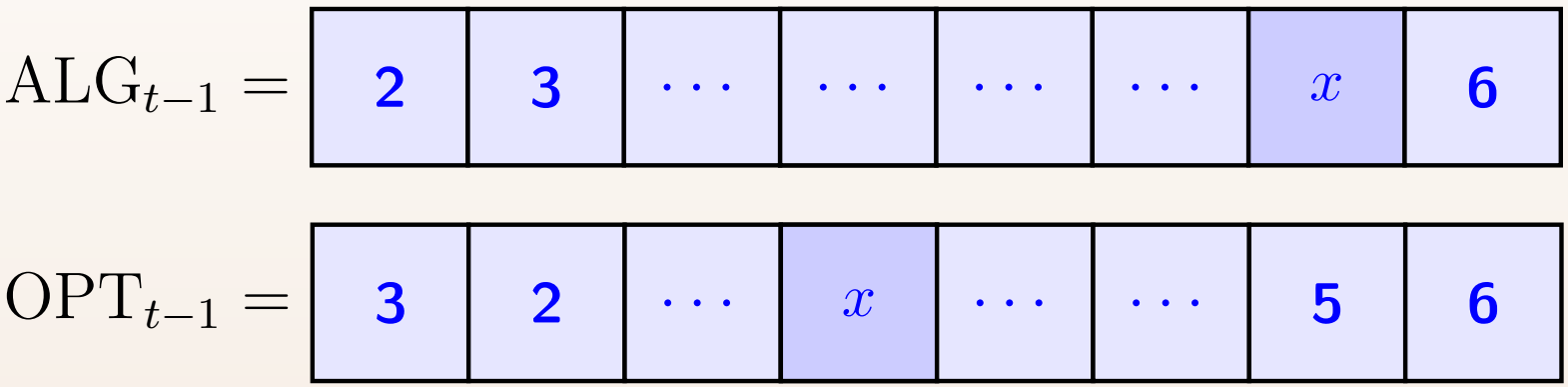
$\Phi(t) = 2$ (#αντιστροφών μεταξύ της λίστας του ALG και του OPT τη στιγμή t).



Το $(1, 2)$ είναι αντιστροφή. Όπως και το $(5, 4)$. Όπως και το $(4, 5)$.

Ανάλυση αντισταθμιστικού κόστους.

Την t-οστή στιγμή έστω x προς προσπέλαση στη θέση i του ALG_{t-1} και στη θέση j του OPT_{t-1}.



$S := \{y \mid y$ είναι πριν το x στο ALG_{t-1} και πριν το x στο $\mathrm{OPT}_{t-1}\}$
$T := \{y \mid y$ είναι πριν το x στο ALG_{t-1} και μετά το x στο $\mathrm{OPT}_{t-1}\}$

Ανάλυση αντισταθμιστικού κόστους.

Την t-οστή στιγμή έστω x προς προσπέλαση στη θέση i του ALG_{t-1} και στη θέση j του OPT_{t-1}.

$\mathrm{ALG}_{t-1} =$

2	3	$\cdots$	$\cdots$	$\cdots$	$\cdots$	x	6

$\mathrm{OPT}_{t-1} =$

3	2	$\cdots$	x	$\cdots$	$\cdots$	5	6

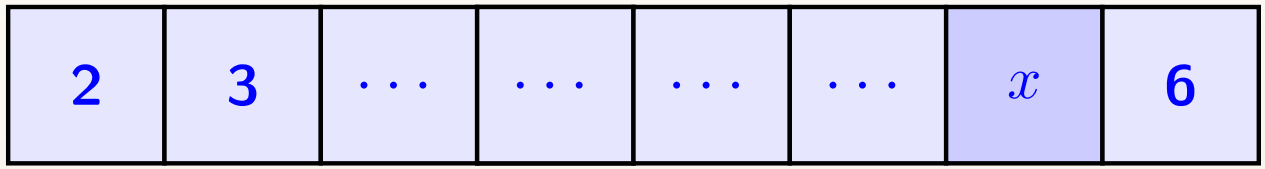
$S := \{y \mid y$ είναι πριν το x στο ALG_{t-1} και πριν το x στο $\mathrm{OPT}_{t-1}\}$
$T := \{y \mid y$ είναι πριν το x στο ALG_{t-1} και μετά το x στο $\mathrm{OPT}_{t-1}\}$

Ο ALG_t πληρώνει $c_t = 1 + (|S| + |T|) + (|S| + |T|) = 1 + 2(|S| + |T|)$.

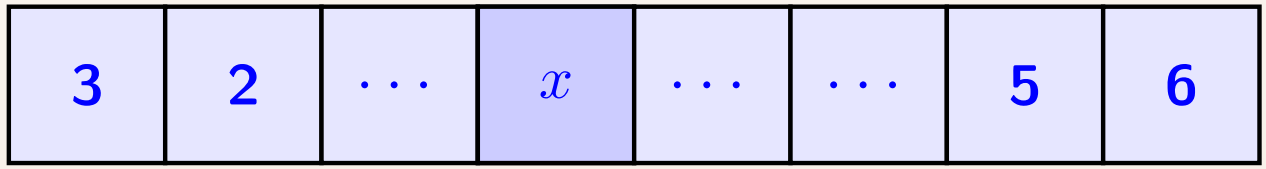
Ανάλυση αντισταθμιστικού κόστους.

Την t-οστή στιγμή έστω x προς προσπέλαση στη θέση i του ALG_{t-1} και στη θέση j του OPT_{t-1}.

$\mathrm{ALG}_{t-1} =$



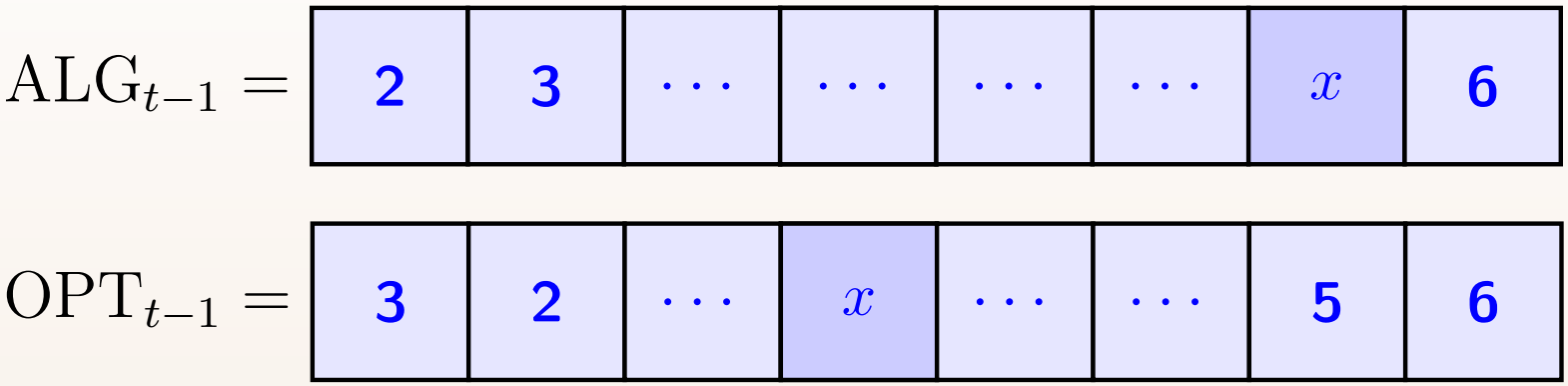
$\mathrm{OPT}_{t-1} =$



$S := \{y \mid y$ είναι πριν το x στο ALG_{t-1} και πριν το x στο $\mathrm{OPT}_{t-1}\}$
$T := \{y \mid y$ είναι πριν το x στο ALG_{t-1} και μετά το x στο $\mathrm{OPT}_{t-1}\}$

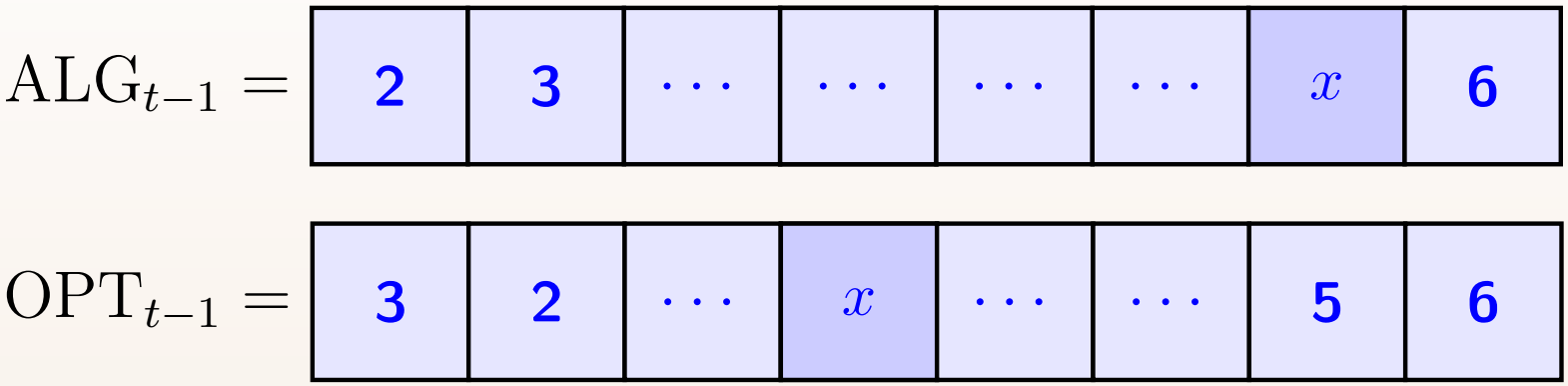
Ο ALG_t πληρώνει $c_t = 1 + (|S| + |T|) + (|S| + |T|) = 1 + 2(|S| + |T|)$.
Ο OPT_t πληρώνει $k_t \geq |S| + 1$.

Ανάλυση αντισταθμιστικού κόστους.



$S := \{y \mid y$ είναι πριν το x στο ALG_{t-1} και πριν το x στο $\mathrm{OPT}_{t-1}\}$
$T := \{y \mid y$ είναι πριν το x στο ALG_{t-1} και μετά το x στο $\mathrm{OPT}_{t-1}\}$

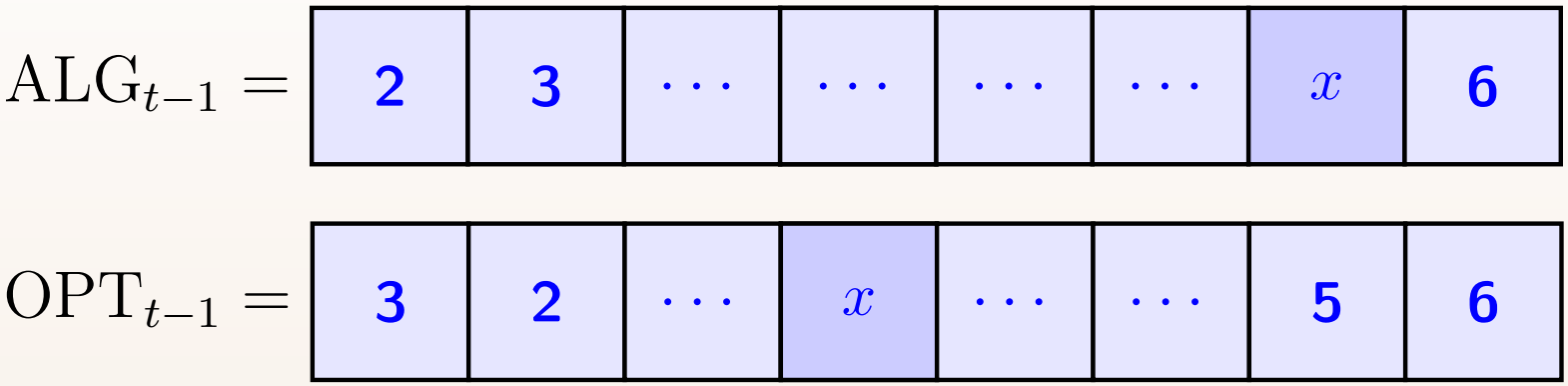
Ανάλυση αντισταθμιστικού κόστους.



$S := \{y \mid y$ είναι πριν το x στο ALG_{t-1} και πριν το x στο $\mathrm{OPT}_{t-1}\}$
$T := \{y \mid y$ είναι πριν το x στο ALG_{t-1} και μετά το x στο $\mathrm{OPT}_{t-1}\}$

$\Phi(t) = 2$ (#αντιστροφών μεταξύ του ALG_t και του OPT_t)

Ανάλυση αντισταθμιστικού κόστους.



$S := \{y \mid y$ είναι πριν το x στο ALG_{t-1} και πριν το x στο $\mathrm{OPT}_{t-1}\}$
$T := \{y \mid y$ είναι πριν το x στο ALG_{t-1} και μετά το x στο $\mathrm{OPT}_{t-1}\}$

$\Phi(t) = 2$ (#αντιστροφών μεταξύ του ALG_t και του OPT_t)

☛ $\Delta\Phi = 2 \cdot (|S| - |T|)$

Ανάλυση αντισταθμιστικού κόστους.

$\mathrm{ALG}_{t-1} =$

2	3	$\cdots$	$\cdots$	$\cdots$	$\cdots$	x	6

$\mathrm{OPT}_{t-1} =$

3	2	$\cdots$	x	$\cdots$	$\cdots$	5	6

$S := \{y \mid y$ είναι πριν το x στο ALG_{t-1} και πριν το x στο $\mathrm{OPT}_{t-1}\}$
$T := \{y \mid y$ είναι πριν το x στο ALG_{t-1} και μετά το x στο $\mathrm{OPT}_{t-1}\}$

$$\Phi(t) = 2\ (\#\text{αντιστροφών μεταξύ του } \mathrm{ALG}_t \text{ και του } \mathrm{OPT}_t)$$

$$\text{👉}\quad \Delta\Phi = 2 \cdot (|S| - |T|)$$

$\hat{c}_t = (1 + 2|S| + 2|T|) + 2(|S| - |T|) =$

Ανάλυση αντισταθμιστικού κόστους.

$\mathrm{ALG}_{t-1} =$

2	3	$\cdots$	$\cdots$	$\cdots$	$\cdots$	x	6

$\mathrm{OPT}_{t-1} =$

3	2	$\cdots$	x	$\cdots$	$\cdots$	5	6

$S := \{y \mid y$ είναι πριν το x στο ALG_{t-1} και πριν το x στο $\mathrm{OPT}_{t-1}\}$
$T := \{y \mid y$ είναι πριν το x στο ALG_{t-1} και μετά το x στο $\mathrm{OPT}_{t-1}\}$

$$\Phi(t) = 2\ (\#\text{αντιστροφών μεταξύ του } \mathrm{ALG}_t \text{ και του } \mathrm{OPT}_t)$$

$$\text{👉}\quad \Delta\Phi = 2 \cdot (|S| - |T|)$$

$$\hat{c}_t = (1 + 2|S| + 2|T|) + 2(|S| - |T|) = 1 + 4|S| \leq 4(1 + |S|)$$

Ανάλυση αντισταθμιστικού κόστους.

$\mathrm{ALG}_{t-1} =$

2	3	$\cdots$	$\cdots$	$\cdots$	$\cdots$	x	6

$\mathrm{OPT}_{t-1} =$

3	2	$\cdots$	x	$\cdots$	$\cdots$	5	6

$S := \{y \mid y$ είναι πριν το x στο ALG_{t-1} και πριν το x στο $\mathrm{OPT}_{t-1}\}$
$T := \{y \mid y$ είναι πριν το x στο ALG_{t-1} και μετά το x στο $\mathrm{OPT}_{t-1}\}$

$$\Phi(t) = 2\ (\#\text{αντιστροφών μεταξύ του } \mathrm{ALG}_t \text{ και του } \mathrm{OPT}_t)$$

$$\text{👉}\quad \Delta\Phi = 2 \cdot (|S| - |T|)$$

$$\hat{c}_t = (1 + 2|S| + 2|T|) + 2(|S| - |T|) = 1 + 4|S| \leq 4(1 + |S|) \leq 4 \cdot \mathrm{OPT}_t \checkmark$$

Ανάλυση Αντισταθμιστικού Κόστους.

Βέβαια δεν αναλύσαμε την περίπτωση όπου ο OPT μετακινεί αντικείμενα μεταξύ δύο διαδοχικών αφίξεων.

Ανάλυση Αντισταθμιστικού Κόστους.

Βέβαια δεν αναλύσαμε την περίπτωση όπου ο OPT μετακινεί αντικείμενα μεταξύ δύο διαδοχικών αφίξεων. Αλλά αυτή μια εύκολη περίπτωση.

👉 Ο ALG σε αυτό το γεγονός δεν πληρώνει τίποτα, και αν ο OPT κάνει s εναλλαγές τότε

Ανάλυση Αντισταθμιστικού Κόστους.

Βέβαια δεν αναλύσαμε την περίπτωση όπου ο OPT μετακινεί αντικείμενα μεταξύ δύο διαδοχικών αφίξεων. Αλλά αυτή μια εύκολη περίπτωση.

👉 Ο ALG σε αυτό το γεγονός δεν πληρώνει τίποτα, και αν ο OPT κάνει s εναλλαγές τότε $\Delta \leq 2s \Rightarrow$ το αντισταθμιστικό κόστος αυξάνεται το πολύ κατά $4s$. ✓

Ανάλυση Αντισταθμιστικού Κόστους.

Βέβαια δεν αναλύσαμε την περίπτωση όπου ο OPT μετακινεί αντικείμενα μεταξύ δύο διαδοχικών αφίξεων. Αλλά αυτή μια εύκολη περίπτωση.

☛ Ο ALG σε αυτό το γεγονός δεν πληρώνει τίποτα, και αν ο OPT κάνει s εναλλαγές τότε $\Delta \leq 2s \Rightarrow$ το αντισταθμιστικό κόστος αυξάνεται το πολύ κατά $4s$. ✓Αθροίζοντας παίρνουμε:

Θεώρημα

Το κόστος του αλγορίθμου ALG είναι το πολύ 4 φορές το κόστος του βέλτιστου αλγόριθμου ο οποίος ξέρει όλες τις προσπελάσεις εκ των προτέρων.

Ανάλυση Αντισταθμιστικού Κόστους.

Βέβαια δεν αναλύσαμε την περίπτωση όπου ο OPT μετακινεί αντικείμενα μεταξύ δύο διαδοχικών αφίξεων. Αλλά αυτή μια εύκολη περίπτωση.

👉 Ο ALG σε αυτό το γεγονός δεν πληρώνει τίποτα, και αν ο OPT κάνει s εναλλαγές τότε $\Delta \leq 2s \Rightarrow$ το αντισταθμιστικό κόστος αυξάνεται το πολύ κατά $4s$. ✓Αθροίζοντας παίρνουμε:

Θεώρημα

Το κόστος του αλγορίθμου ALG είναι το πολύ 4 φορές το κόστος του βέλτιστου αλγόριθμου ο οποίος ξέρει όλες τις προσπελάσεις εκ των προτέρων.

Στο επόμενο επεισόδιο: σωροί Fibonacci.