# Amortized Computational Complexity

## Robert Endre Tarjan 1985

Τσούμας Γεώργιος

# Αποσβεστική Ανάλυση

Αποσβεστική Ανάλυση **(Amortized Analysis)**: είναι μια διαφορετική οπτική σκοπιά υπολογισμού της πολυπλοκότητας ενός αλγορίθμου. Με αυτόν τον τρόπο, υπολογίζουμε τον *μέσο χρόνο* που απαιτείται για την εκτέλεση κάποιας πράξης μιας δομής δεδομένων, επί όλων τον εκτελούμενων πράξεων.

- Επιλέγουμε να εστιάσουμε στο κόστος *μιας ακολουθίας πράξεων*, σε αντίθεση με το κόστος κάθε μεμονωμένης πράξης.
- **Προσοχή!** Η Αποσβεστική Ανάλυση δεν είναι κάνει χρήση της έννοιας της πιθανότητας. Αυτό σημαίνει ότι *εγγυάται* ένα αυστηρό άνω φράγμα για την χειρότερη δυνατή περίπτωση.

# Αποσβεστική Ανάλυση- Χτίζοντας την διαίσθηση μας

Ας σκεφτούμε το ακόλουθο απλό παράδειγμα. Έχουμε n το πλήθος στοιχεία, και μια στοίβα με τρεις βασικές διεργασίες:

- Την push(i), η οποία τοποθετεί το στοιχείο i στην κορυφή της στοίβας.
- Την pop, η οποία αφαιρεί το κορυφαίο στοιχείο της στοίβας και το επιστρέφει.
- Την multipop(k), η οποία αφαιρεί k στοιχεία από την κορυφή.

Πολυπλοκότητα αλγορίθμου: $O(n^2)$ καθώς σε μια γεμάτη στοίβα μπορούμε να αφαιρέσουμε το πολύ n στοιχεία και υπάρχουν n στοιχεία. Αυτή η ανάλυση, αν και τυπικά σωστή, είναι υπερβολικά γενναιόδωρη.

# Αποσβεστική Ανάλυση- Χτίζοντας την διαίσθηση μας

Πολυπλοκότητα την χρήση Αποσβεστικής Ανάλυσης: αρκεί να παρατηρήσουμε ότι κάθε στοιχείο που εισάγεται στην στοίβα, μπορεί να εξαχθεί μόνο μία φορά. Άρα για n το πλήθος στοιχεία, έχουμε το πολύ 2n διεργασίες. Έτσι βλέπουμε ότι το συνολικό κόστος είναι της τάξης του O(n).

Αν και το προηγούμενο παράδειγμα ήταν τετριμμένο, και έκανε χρήση μιας φορμαλιστικής "αδυναμίας" του συνηθισμένου ορισμού της πολυπλοκότητας, βλέπουμε ότι η αποσβεστική προσέγγιση μπορεί να μας δώσει καλύτερα αποτελέσματα σε κάποιες περιπτώσεις, στις οποίες ο κλασσικός ορισμός αποτυγχάνει.

Κατά κύριο λόγο, η Αποσβεστική Ανάλυση χρησιμοποιείται για την ανάλυση της συμπεριφοράς μιας δομής δεδομένων στην οποία διενεργείται μια ακολουθία από αλλεπάλληλες διεργασίες.

# Αποσβεστική Ανάλυση - Χρεωπιστωτική μέθοδο

Χρεωπιστωτική μέθοδος **(accounting method)**.Εδώ αποδίδουμε διαφορετικό αποσβεστικό κόστος στους διαφορετικούς τύπους πράξεων, χρεώνοντας ορισμένες πράξεις περισσότερο ή λιγότερο από ότι κοστίζουν στην πραγματικότητα. Όταν το αποσβεστικό κόστος μιας πράξης υπερβαίνει το πραγματικό κόστος, η διαφορά τους αποδίδεται σε συγκεκριμένα αντικείμενα της δομής δεδομένων σαν **πίστωση.** Στην συνέχεια, μπορούμε να κάνουμε χρήση της πίστωσης αυτής, για να μειώσουμε το αποσβεστικό κόστος μιας πράξης στο μέλλον.

Στο προηγούμενο παράδειγμα: Ορίζω το αποσβεστικό κόστος της διεργασίας pop να είναι 2, και τα υπόλοιπα να είναι 0.

# Αποσβεστική Ανάλυση - Μέθοδος του δυναμικού

Η Μέθοδο του δυναμικού **(potential method)**. Αναπαριστούμε την κατάσταση της δομής δεδομένων σαν “δυνητική ενέργεια”. Ορίζουμε **συνάρτηση δυναμικού Φ**, η οποία περιγράφει την δομή σε κάθε δυνατή κατάσταση. Ο σκοπός εδώ είναι να ορίσουμε διεργασίες έτσι ώστε να “αποθηκεύουμε ενέργεια” στην δομή, και όταν χρειαστεί να μπορέσουμε να την “απελευθερώσουμε” ώστε να αποπληρώσουμε μελλοντικές.

# Αποσβεστική Ανάλυση - Μέθοδος του δυναμικού

**Ορισμός:** έστω μια δομή δεδομένων D0 και έστω μια ακολουθία n διεργασιών πάνω σε αυτή. Για κάθε i = 1,2,...,n ορίζουμε ci το πραγματικό κόστος της i-οστης διεργασίας και Di την δομή που προκύπτει μετά την διεργασία πανω στην Di-1 δομή. Ορίζουμε την συνάρτηση Φ: Di □ R > = 0, για καθε i = (1,2,...,n).

Συμβολίζουμε επίσης, για κάθε i, σαν ΔΦ( i )= Φ(Di) – Φ(Di-1) και την ονομάζουμε δυνητική διαφορά για την κατάσταση i.

Ορίζουμε επίσης το αποσβεστικό κόστος c΄i = ci + Φ(Di) – Φ(Di-1)= ci + ΔΦ(i).

Αφού οι Φ(Di) όροι ειναι τηλεσκοπικοί:

- **Πρέπει** Φ(D0) = 0 και για κάθε i Φ(Di) να

είναι μη αρνητικό.

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} (c_i + \Phi(D_i) - \Phi(D_{i-1}))$$

$$= \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) .$$

# Αποσβεστική Ανάλυση - Μέθοδος του δυναμικού

Αν η ΔΦ(i) είναι αρνητική, τότε στην i-οστη διεργασία ‘’εκλύεται’’ ενέργεια από την δομή, την οποία χρησιμοποιούμε για να πληρώσουμε μια διεργασία.

Αν η ΔΦ(i) είναι θετική,τότε στην i-οστη διεργασία αποθηκεύουμε περαιτέρω ενέργεια μέσα στην δομή μας, για μελλοντική χρήση.

Στο προηγούμενο παράδειγμα: Ορίζουμε σαν συνάρτηση δυναμικού να της στοίβας το πλήθος τον στοιχείων της. Επομένως το κόστος της c’(push)= c(push)+(k+1)-k=2, για μια στοίβα μεγέθους k. Παρατηρούμε ότι το κόστος της c’(multipop(k))= c(multipop(k) + ΔΦ= k -k = 0. Αντίστοιχα και για pop.

Οι δύο αυτές μέθοδοι είναι λογικά ισοδύναμες, και θα χρησιμοποιούμε οποία απο τις δυο μας φαίνεται κατάλληλη.

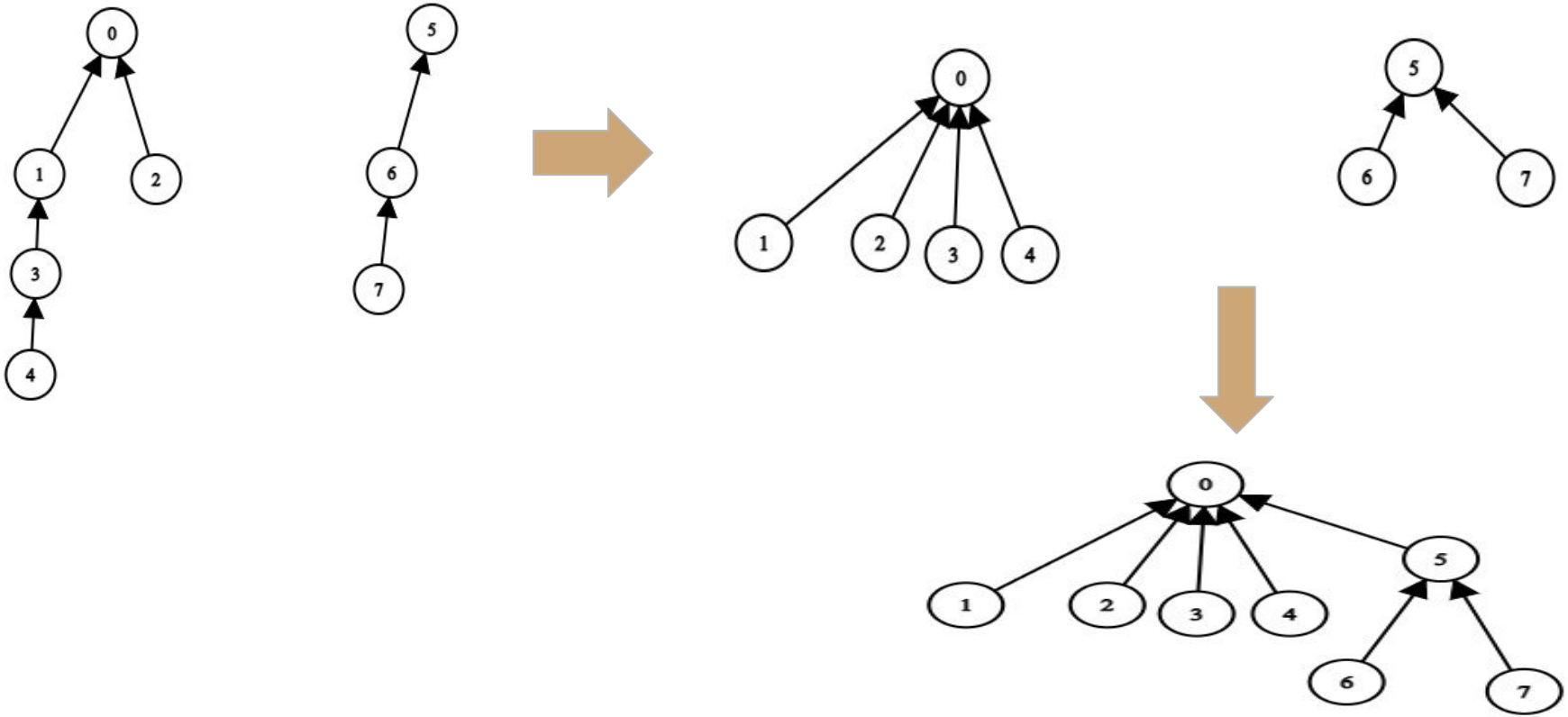
# Union Find με σύμπτυξη μονοπατιών

Ορίζουμε δύο διεργασίες:

- Την συνάρτηση **Find(a)**, η οποία θα δέχεται μια κορυφή a και θα γυρνάει την ρίζα του δέντρου που στην οποία ανήκει η a (η ρίζα κάθε δέντρου θα έχει γονέα τον εαυτό της).
- Την συνάρτηση **Union(a,b)** που θα ενώνει δύο δένδρα που περιέχουν την a και την b αντίστοιχα, μέσω της κλήσης της Find(a) και Find(b).

Η τροποποίηση που κάνουμε στην Find είναι ότι κάθε κορυφή που προσπελαύνουμε μέχρι να φτάσουμε στην ρίζα, την υποτάσουμε στην ρίζα. Στόχος μας είναι να κρατάμε τα ύψη των δένδρων μικρά.

# Union Find με σύμπτυξη μονοπατιών



# Union Find με σύμπτυξη μονοπατιών

Με την **χρεωπιστωτική μέθοδο**: στην ουσία πιστώνουμε κάθε κορυφή η οποία έγινε παιδί της ρίζας με 2 πιστωτικές μονάδες, οι οποίες δαπανήθηκαν έναντι της περίπτωσης της απλής union find. Όμως κάθε φορά που θα εκτελώ το Find(a), θα γλιτώνω 2 διεργασίες για κάθε γονέα της. Ξέρουμε όμως ότι στην πραγματικότητα θα προσπελαύνω τους προγόνους της αρκετές περισσότερες.

Με την **μέθοδο του δυναμικού**: μπορούμε να σκεφτούμε ότι η δομή αποθηκεύει ενέργεια σε κάθε μεταφορά κορυφής μέσω τις Find, ή οποία απελευθερώνεται για να εκτελέσει πιο γρήγορα τις Union και Find. Εδώ μπορούμε να σκεφτόμαστε την ενέργεια και σαν “τον δείκτη τάξης - χάους)” της δομής μας σε κάθε κατάσταση.

# Αυτοπροσαρμοζόμενες Δομές Δεδομένων

Η αποσβεστική ανάλυση μας δίνει μια άλλη ιδέα για το πως μπορούμε να δομήσουμε και να οργανώσουμε δομές δεδομένων. Σκοπός μας είναι κάθε φορά που θα προσπελαύνουμε την δομή, να την επεξεργαζόμαστε κατά ένας μικρό και φτηνό τρόπο, ώστε να μειώσουμε τον χρόνο που απαιτείται για μελλοντικές εργασίες. Ο σχεδιασμός αυτός μπορεί να πετύχει δομές με πολύ γρήγορες και απλές διεργασίες προσπέλασης και ενημέρωσης, οι οποίες προσαρμόζονται στα μοτίβα των χρηστών. Τέτειες δομές ονομάζονται αυτοπροσαρμοζόμενες **(Self-adjusting data structures).**

# Αυτοπροσαρμοζόμενες Δομές Δεδομένων - Λοξός Σωρός

- Ο λοξός σωρός **(skew heap)**, είναι ένα δυαδικό δέντρο, δηλαδή είναι δέντρο και κάθε κορυφή έχει το πολύ δύο παιδία.
- Επίσης έχει την ιδιότητα του σωρού, δηλαδή η τιμή κάθε κορυφής ειναι μικρότερη από αυτές των παιδιών της.

Θέλουμε η δομή μας να υποστηρίζει δύο διεργασίες:

- **Deletemin(h)**: θα βρίσκει το μικρότερο στοιχείο του σωρού h, θα το επιστρέφει και θα το διαγράφει.
- **Meld(h,s)**: θα ενώνει τους σωρούς h και s.

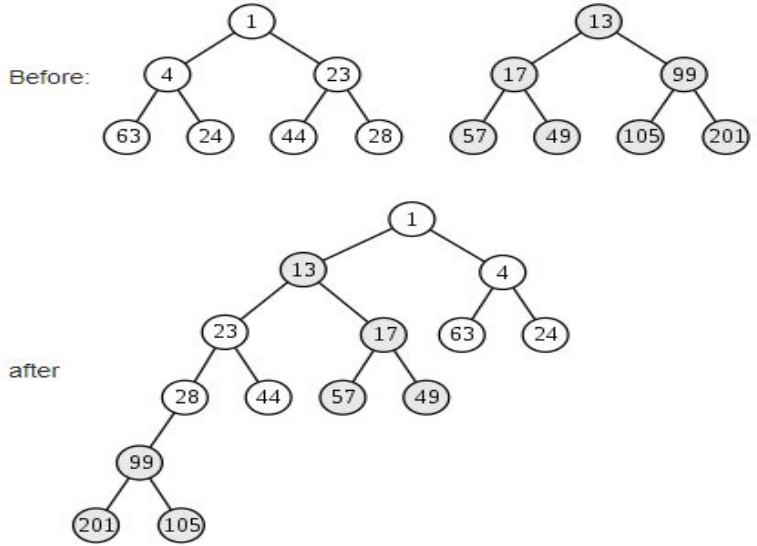
Meld(h,s):

1. Συγκρίνω τις δύο ρίζες και υποτάσω την μεγαλύτερη στην μικρότερη.
2. Κάνω το αριστερό υπόδεντρο της μικρότερης ρίζας το καινούργιο δεξί υπόδεντρο.
3. Επαναλαμβάνω το (1) για το δεξί υποδεντρό της μικρότερης ρίζας με τον σωρό της χαμένης ρίζας.

Deletemin(h):

Αφαιρώ την ρίζα και εκτελώ meld

για τα δύο υπόδεντρα.



# Λοξός Σωρός

Η εναλλαγή αριστερού-δεξιού μέρους στην διεργασία Meld αποσκοπεί στην αποφυγή δημιουργίας ενός μεγάλου μονοπατιού στο δεξιό μέλος του σωρού. Πράγματι, μπορούμε να παρατηρήσουμε ότι η διεργασία προσπελαύνει γραμμικά το πιο δεξί μονοπάτι του σωρού. Αυτό θα μας δημιουργούσε προβλήματα, καθώς και οι δύο διεργασίες έχουν κόστος όσο το ύψος του δέντρου. Εάν το meld δεν γίνονταν καλά, θα είχαν κόστος O(n), αντί για O(logn).

Αποσβεστική ανάλυση κόστους: θέτουμε σαν Βαρος(v)= πλήθος εσωτερικών απογόνων του v. Και χαρακτηρίζουμε μία κορυφή βαρία, αν έχει μεγαλύτερο βάρος από τον γονέα της. Σε κάθε βαρία δεξιά κορυφή δίνουμε μια πίστωση. Παρατηρούμε ότι οι βαριές κορυφές είναι αυτές που είναι πιο πιθανές να βρεθούν στο δεξί μέρος του σωρού έπειτα από μια συγκρίση.

# Άλλες εφαρμογές της Αποσβεστικής Ανάλυσης

- Splay trees
- Red-Black trees
- Polynomial heaps
- Fibonacci heaps

Όλες οι παραπάνω δομές δεδομένων κάνουν χρήση μιας συνάρτησης δυναμικού για την ανάλυση τους. Στην ουσία η αποσβεστική ανάλυση μας δίνει την δυνατότητα να κάνουμε χρήση της εσωτερικής δομής των δεδομένων (π.χ. διάταξη ή συχνότητα προσπέλασης), χωρίς να έχουμε παραπάνω πληροφορίες ή χωρίς να κάνουμε υποθέσεις για την δομή των δεδομένων (π.χ. την κατανομή πιθανότητας των προσπελάσεων τους).

Σας ευχαριστώ για την προσοχή σας