Heapsort Using Multiple Heaps

Δ. Λεβεντέας – Χ. Ζαρολιάγκης

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

29 Αυγούστου 2008

Εισαγωγή

- Το heapsort προτάθηκε το 1964 από τον Williams.

Εισαγωγή

- Το heapsort προτάθηκε το 1964 από τον Williams.
- Η σχετική έρευνα επικεντρώθηκε:
  - Κατασκευή του heap (Build-Heap) - (Wegener, (1993))
  - Επαναφορά της ιδιότητας του σωρού Heapify- (Schaffer and Sedgewick, (1993))
  - Στην δομή του σωρού - (Edelkamp and Wegener, (2000))

Εισαγωγή

- Το heapsort προτάθηκε το 1964 από τον Williams.
- Η σχετική έρευνα επικεντρώθηκε:
  - Κατασκευή του heap (Build-Heap) - (Wegener, (1993))
  - Επαναφορά της ιδιότητας του σωρού Heapify- (Schaffer and Sedgewick, (1993))
  - Στην δομή του σωρού - (Edelkamp and Wegener, (2000))
- Χρήση πολλών σωρών αντί για ενός.

Εισαγωγή

- Επικεντρωνόμαστε στην διαδικασία εξαγωγής του επόμενου στοιχείου

Εισαγωγή

- Επικεντρωνόμαστε στην διαδικασία εξαγωγής του επόμενου στοιχείου
- Παρουσιάζουμε δυο τροποποιήσεις του αλγορίθμου του Williams

Εισαγωγή

- Επικεντρωνόμαστε στην διαδικασία εξαγωγής του επόμενου στοιχείου
- Παρουσιάζουμε δυο τροποποιήσεις του αλγορίθμου του Williams
- Σκοπός μας είναι αλλάζοντας μικρό μέρος του κώδικα
  - Διατήρηση της ιδιότητας ταξινόμησης in-place
  - Ταχύτερη εξαγωγή τελικού αποτελέσματος

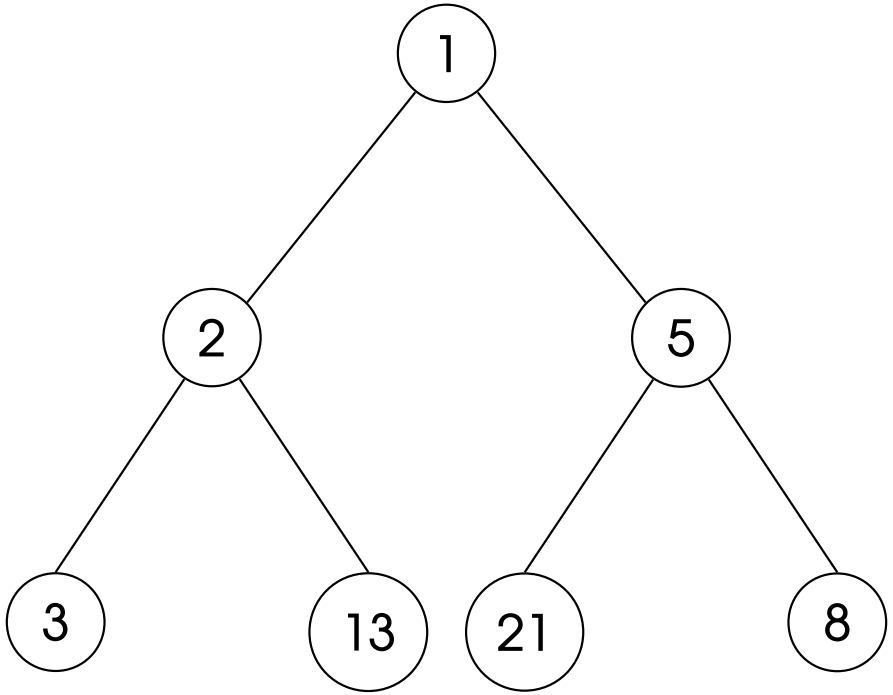
Ορισμός

Ορισμός

Ένας σωρός (min-heap) είναι ένας πίνακας στοιχείων a_j, $1 \leq j \leq n$ *που ικανοποιεί την μονοτονική ιδιότητα μονοπατιού (path-monotonic property):* $a_j \geq a_{\lfloor \frac{j}{2} \rfloor}$ *για* $j = 2, 3, \ldots, n$ *όπου* $\lfloor x \rfloor$ *είναι το ακέραιο υπόλοιπο του πραγματικού αριθμού* x

Παράδειγμα

Σε μορφή δέντρου:



Σε μορφή πίνακα:

1	2	5	3	13	21	8

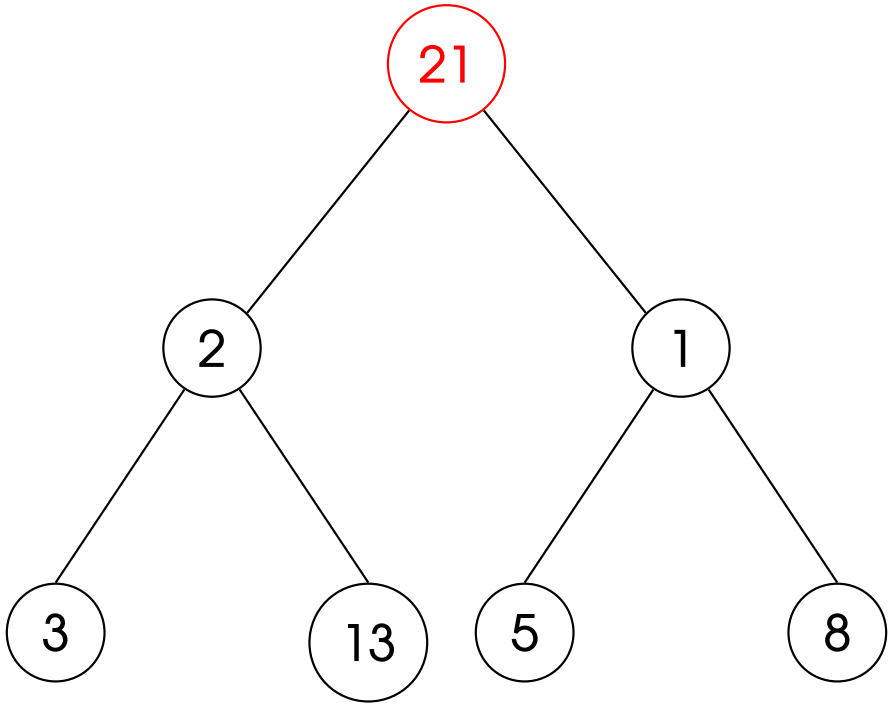
Ορισμός

Ορισμός

Θεωρούμε ένα πίνακα A και τον δείκτη i. Τα υποδέντρα που έχουν ως ρίζα τους τα παιδιά του A[i] είναι ήδη σωροί, αλλά το ίδιο το δέντρο με ρίζα το A[i] μπορεί να παραβιάζει την ιδιότητα του σωρού. Η διαδικασία Heapify κάνει τις απαραίτητες αλλαγές ώστε το δέντρο με ρίζα το A[i] να γίνει σωρός.

Παράδειγμα

Σε μορφή δέντρου:

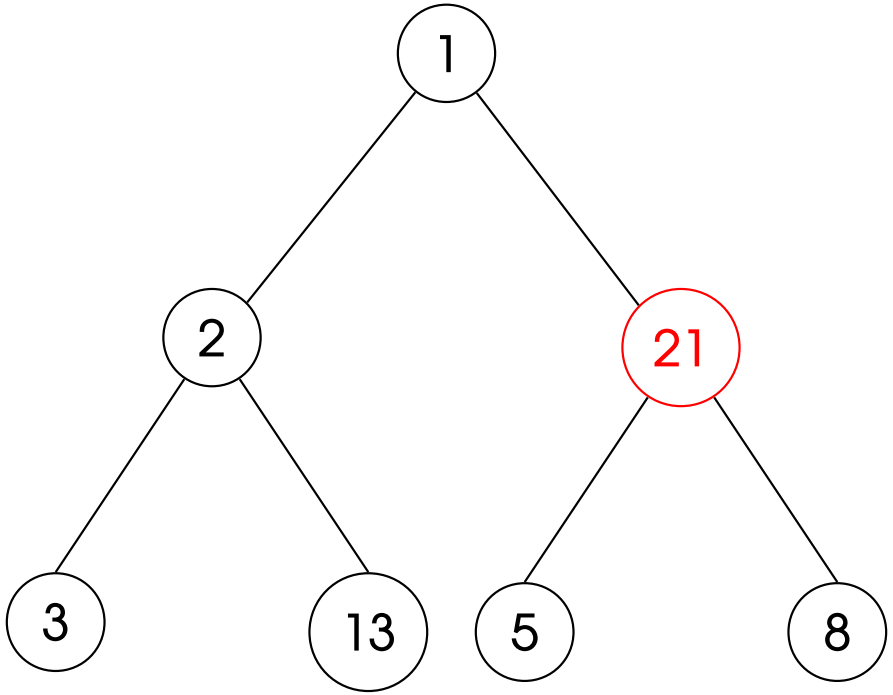


Σε μορφή πίνακα:

21	2	1	3	13	5	8

Παράδειγμα

Σε μορφή δέντρου:

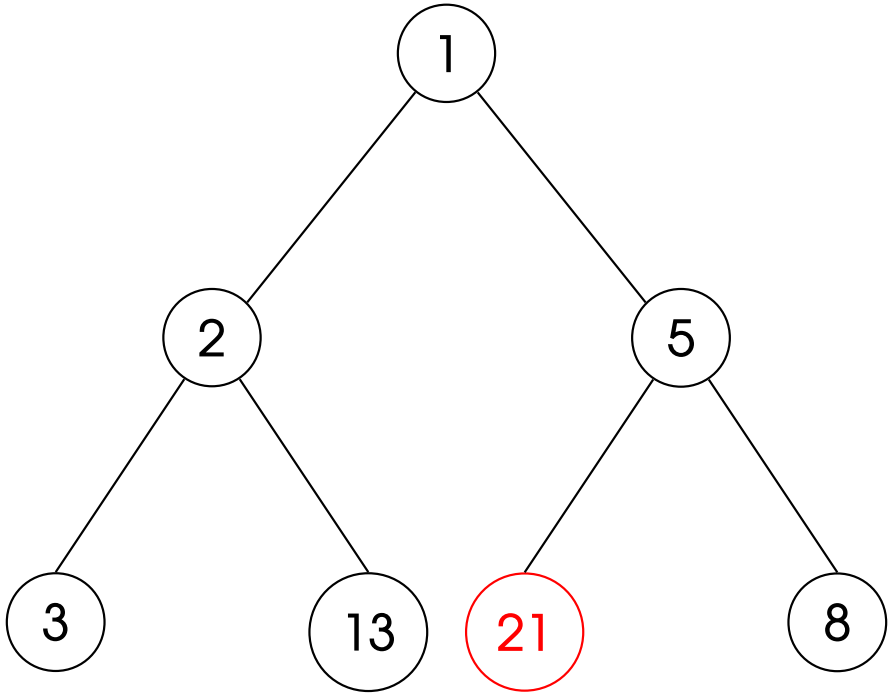


Σε μορφή πίνακα:

1	2	21	3	13	5	8

Παράδειγμα

Σε μορφή δέντρου:



Σε μορφή πίνακα:

1	2	5	3	13	21	8

Ορισμός

Ορισμός

Μπορούμε να χρησιμοποιήσουμε την συνάρτηση Heapify με μια προσέγγιση από κάτω προς τα πάνω (bottom-up) ώστε να μετατρέψουμε ένα πίνακα $A[1..n]$ σε ένα σωρό. Αφού όλα τα στοιχεία στον υποπίνακα $A[\lfloor \frac{n}{2} \rfloor + 1..n]$ είναι φύλλα, η build_heap επισκέπτεται τους υπόλοιπους κόμβους και καλεί την Heapify σε κάθε έναν.

Heapsort - Πως δουλεύει

1. Δημιουργία ενός σωρού heap - (Build-Heap)

Heapsort - Πως δουλεύει

1. Δημιουργία ενός σωρού heap - (Build-Heap)
2. Εύρεση μικρότερης τιμής (root)

Heapsort - Πως δουλεύει

1. Δημιουργία ενός σωρού heap - (Build-Heap)
2. Εύρεση μικρότερης τιμής (root)
3. Εναλλαγή με το τελευταίο αταξινόμητο στοιχείο (φύλλο του σωρού)

Heapsort - Πως δουλεύει

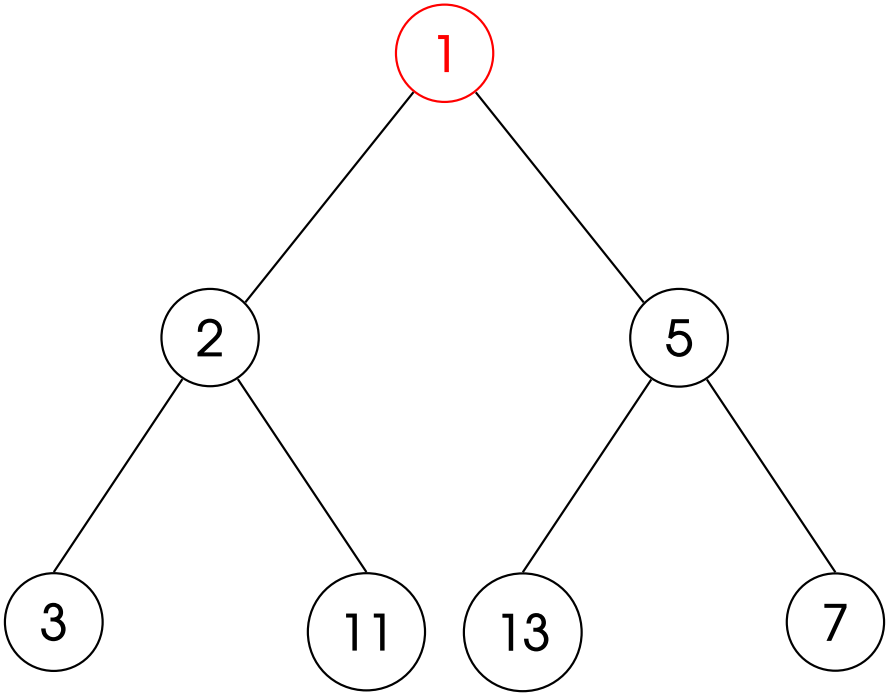
1. Δημιουργία ενός σωρού heap - (Build-Heap)
2. Εύρεση μικρότερης τιμής (root)
3. Εναλλαγή με το τελευταίο αταξινόμητο στοιχείο (φύλλο του σωρού)
4. Επαναφορά της ιδιότητας του σωρού για τα υπόλοιπα αταξινόμητα στοιχεία μέσα στο array (heapify)

Heapsort - Πως δουλεύει

1. Δημιουργία ενός σωρού heap - (Build-Heap)
2. Εύρεση μικρότερης τιμής (root)
3. Εναλλαγή με το τελευταίο αταξινόμητο στοιχείο (φύλλο του σωρού)
4. Επαναφορά της ιδιότητας του σωρού για τα υπόλοιπα αταξινόμητα στοιχεία μέσα στο array (heapify)
5. Επανάληψη της διαδικασίας από το 2ο βήμα μέχρι να ταξινομηθούν όλα τα στοιχεία του πίνακα.

Παράδειγμα

Σε μορφή δέντρου:

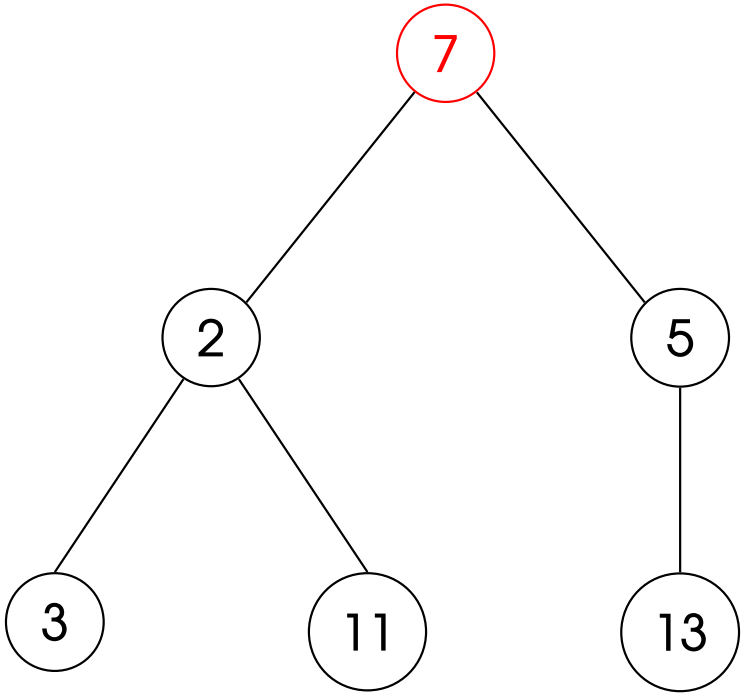


Σε μορφή πίνακα:

1	2	5	3	11	13	7

Παράδειγμα

Σε μορφή δέντρου:

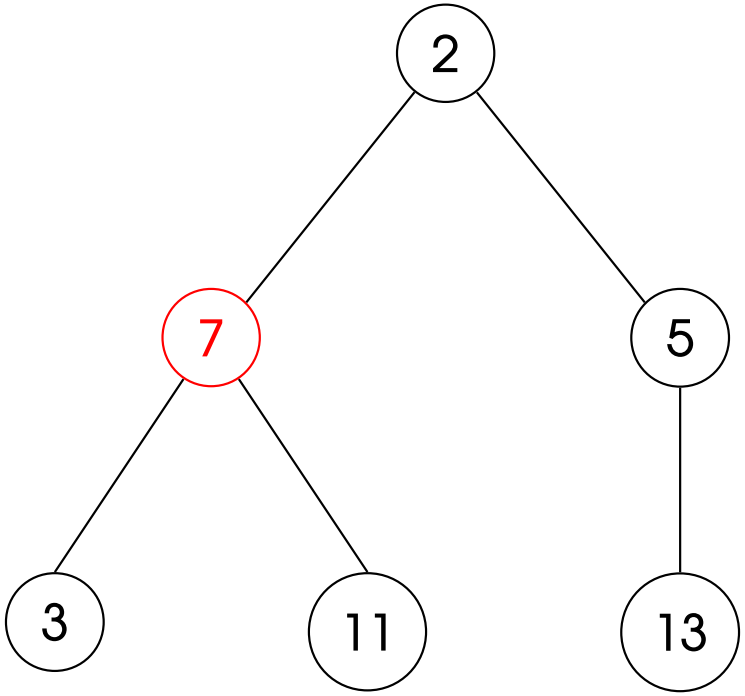


Σε μορφή πίνακα:

7	2	5	3	11	13	1

Παράδειγμα

Σε μορφή δέντρου:

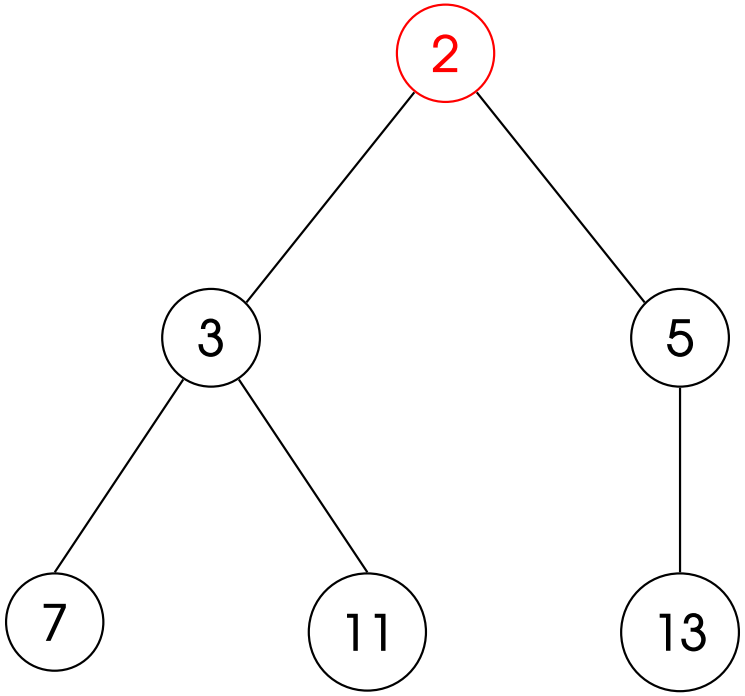


Σε μορφή πίνακα:

2	7	5	3	11	13	1

Παράδειγμα

Σε μορφή δέντρου:

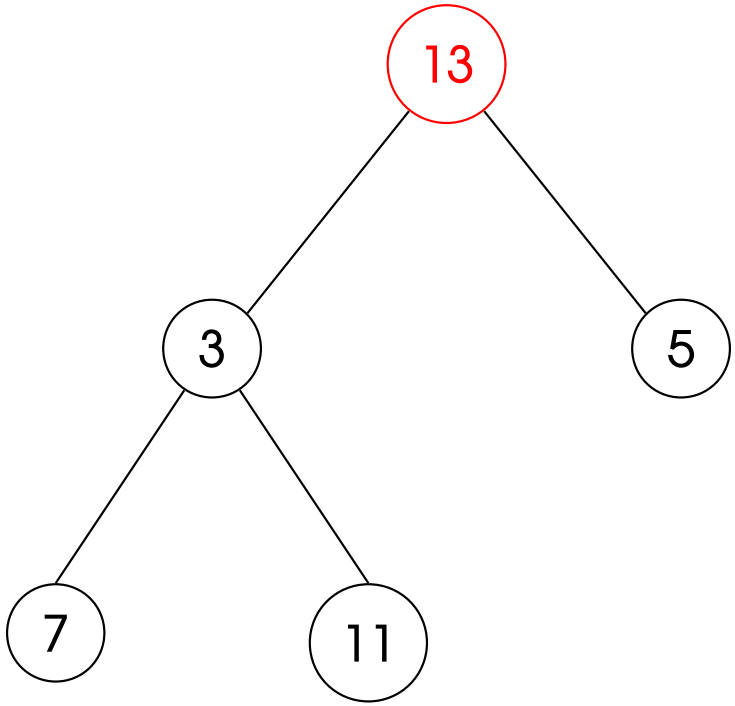


Σε μορφή πίνακα:

2	3	5	7	11	13	1

Παράδειγμα

Σε μορφή δέντρου:

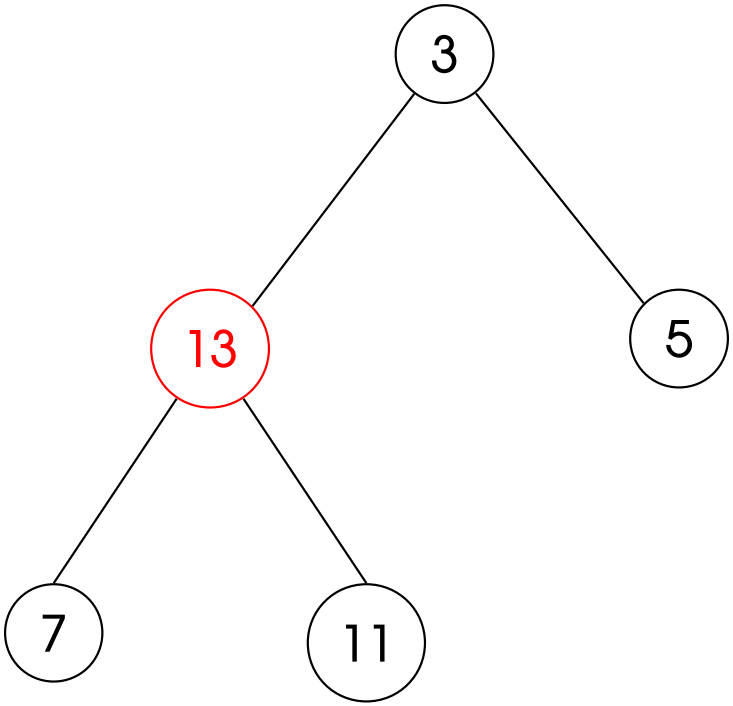


Σε μορφή πίνακα:

13	3	5	7	11	2	1

Παράδειγμα

Σε μορφή δέντρου:

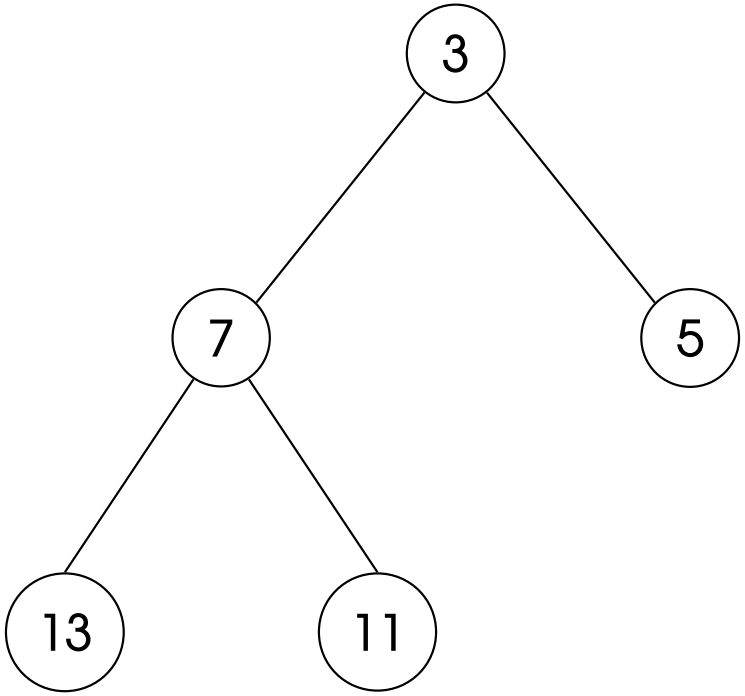


Σε μορφή πίνακα:

3	13	5	7	11	2	1

Παράδειγμα

Σε μορφή δέντρου:

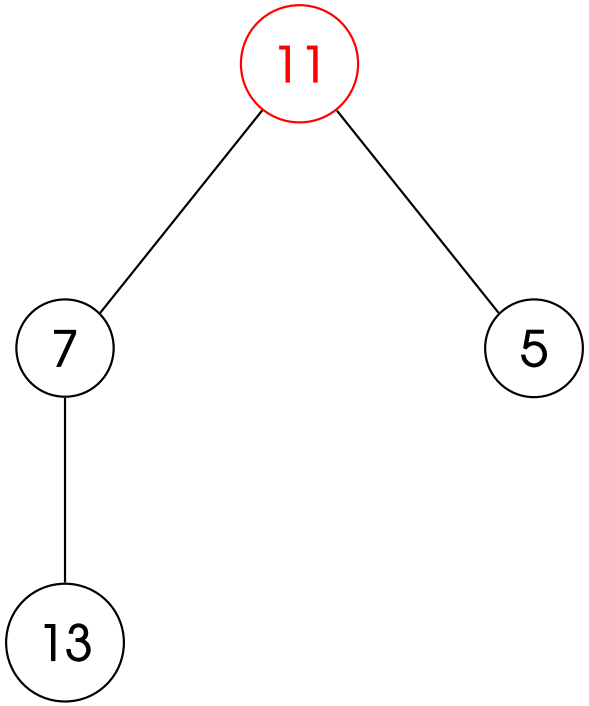


Σε μορφή πίνακα:

3	7	5	13	11	2	1

Παράδειγμα

Σε μορφή δέντρου:



Σε μορφή πίνακα:

11	7	5	13	3	2	1

Παράδειγμα

Προχωρώντας παρομοίως έχουμε σταδιακά:

1. | 5 | 7 | 11 | 13 | 3 | 2 | 1 |

Παράδειγμα

Προχωρώντας παρομοίως έχουμε σταδιακά:

1. | 5 | 7 | 11 | 13 | 3 | 2 | 1 |
2. | 7 | 13 | 11 | 5 | 3 | 2 | 1 |

Παράδειγμα

Προχωρώντας παρομοίως έχουμε σταδιακά:

1. | 5 | 7 | 11 | 13 | 3 | 2 | 1 |
2. | 7 | 13 | 11 | 5 | 3 | 2 | 1 |
3. | 11 | 13 | 7 | 5 | 3 | 2 | 1 |

Παράδειγμα

Προχωρώντας παρομοίως έχουμε σταδιακά:

1. | 5 | 7 | 11 | 13 | 3 | 2 | 1 |
2. | 7 | 13 | 11 | 5 | 3 | 2 | 1 |
3. | 11 | 13 | 7 | 5 | 3 | 2 | 1 |
4. Και τέλος έχουμε ταξινομημένους τους επτά πρώτους αριθμούς

13	11	7	5	3	2	1

Heapsort - Ιδιότητες

- Ταξινόμηση in-place

Heapsort - Ιδιότητες

- Ταξινόμηση in-place
- Ταξινόμηση σε $O(n \log n)$

Heapsort - Ιδιότητες

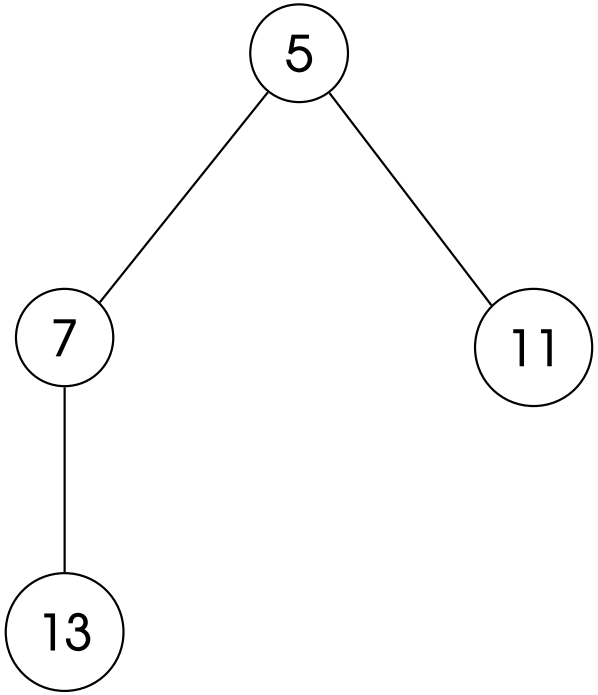
- Ταξινόμηση in-place
- Ταξινόμηση σε $O(n \log n)$
- Ασυμπτωτικά βέλτιστο για οποιοδήποτε αλγόριθμο ταξινόμησης βασισμένο σε συγκρίσεις $\Omega(n \log n)$

Heap - Παρατηρήσεις

- Υπάρχουν διάφοροι τρόποι που μπορούμε να εξετάσουμε έναν σωρό

Heap - Παρατηρήσεις

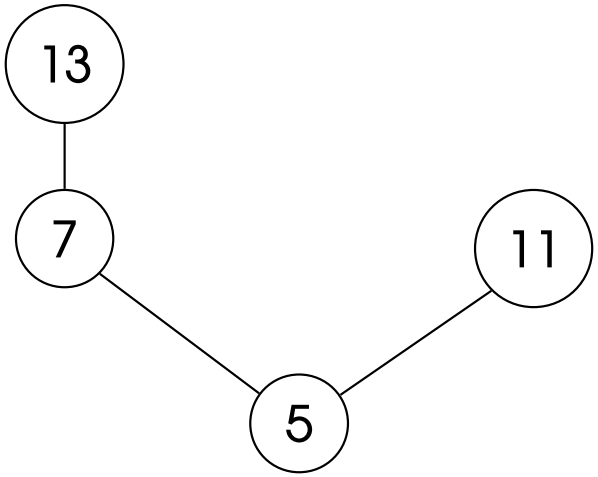
- Υπάρχουν διάφοροι τρόποι που μπορούμε να εξετάσουμε έναν σωρό



-

Heap - Παρατηρήσεις

- Υπάρχουν διάφοροι τρόποι που μπορούμε να εξετάσουμε έναν σωρό



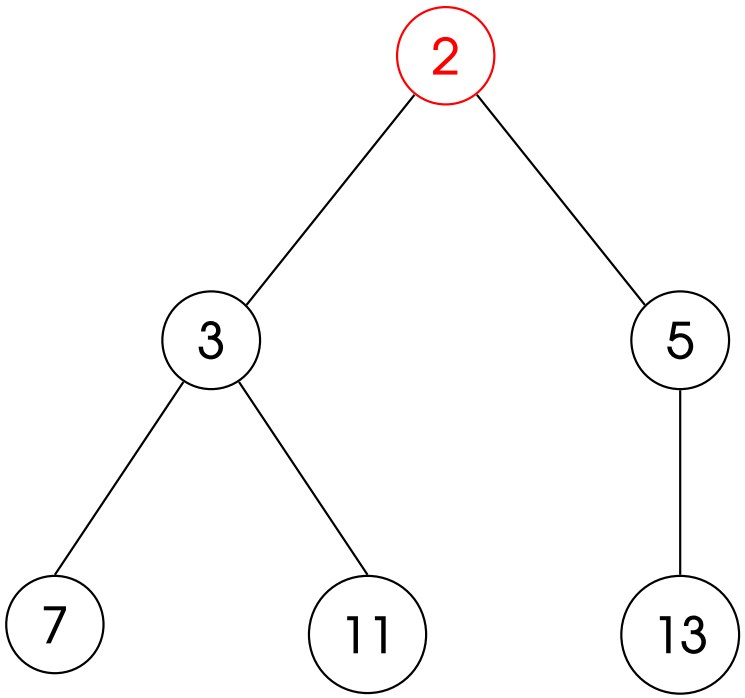
-

Heap - Παρατηρήσεις

- Αναδρομική προσέγγιση - Μετά από την αφαίρεση της ρίζας ενός heap τα δέντρα που σχηματίζονται αποτελούν ανεξάρτητα heaps

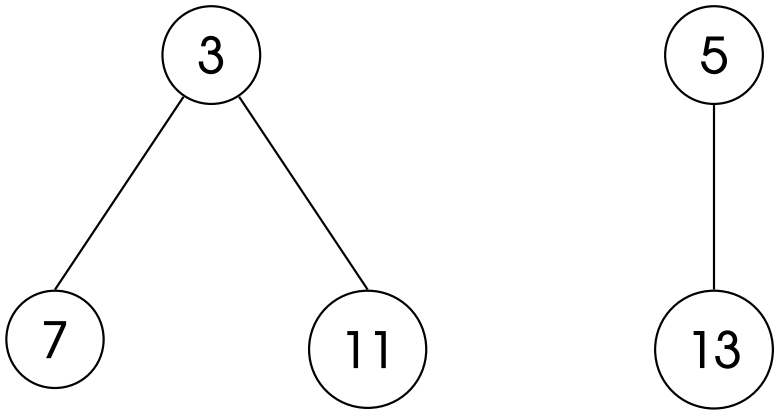
Heap - Παρατηρήσεις

- Αναδρομική προσέγγιση - Μετά από την αφαίρεση της ρίζας ενός heap τα δέντρα που σχηματίζονται αποτελούν ανεξάρτητα heaps
- Σε μορφή δέντρου:



Heap - Παρατηρήσεις

- Αναδρομική προσέγγιση - Μετά από την αφαίρεση της ρίζας ενός heap τα δέντρα που σχηματίζονται αποτελούν ανεξάρτητα heaps
- Σε μορφή δέντρου:

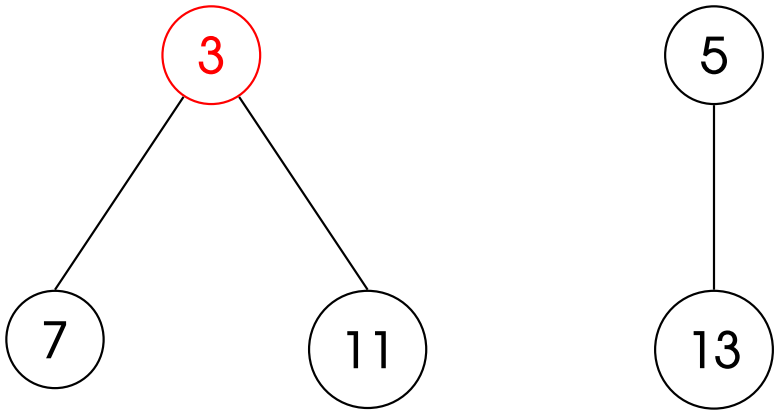


Αρχική ιδέα - Χρήση δυο σωρών

- Μετά την αφαίρεση της ρίζας το επόμενο στοιχείο θα είναι η ρίζα ενός από τα δυο heaps

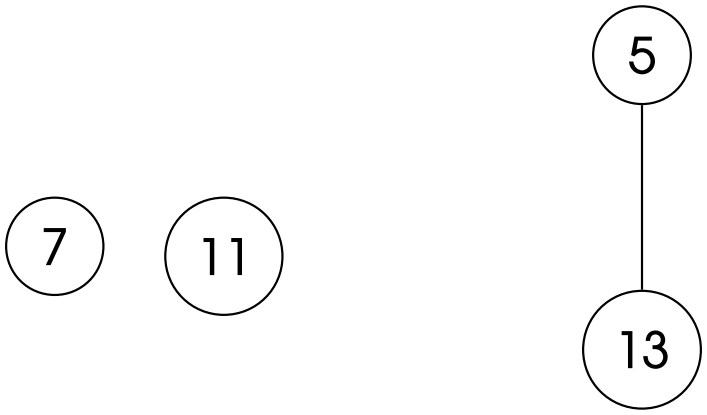
Αρχική ιδέα - Χρήση δυο σωρών

- Μετά την αφαίρεση της ρίζας το επόμενο στοιχείο θα είναι η ρίζα ενός από τα δυο heaps
- Απαιτείται μόλις μια σύγκριση



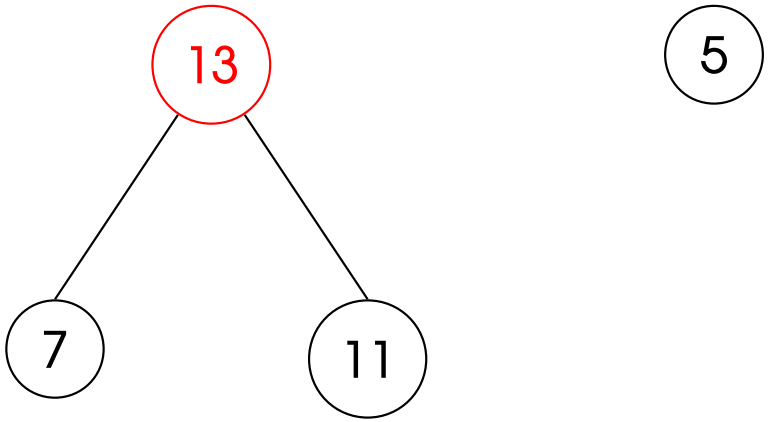
Μερική Χρήση δυο σωρών

- Μετά την αφαίρεση της ρίζας το επόμενο στοιχείο θα είναι η ρίζα ενός από τα δυο heaps
- Απαιτείται μόλις μια σύγκριση



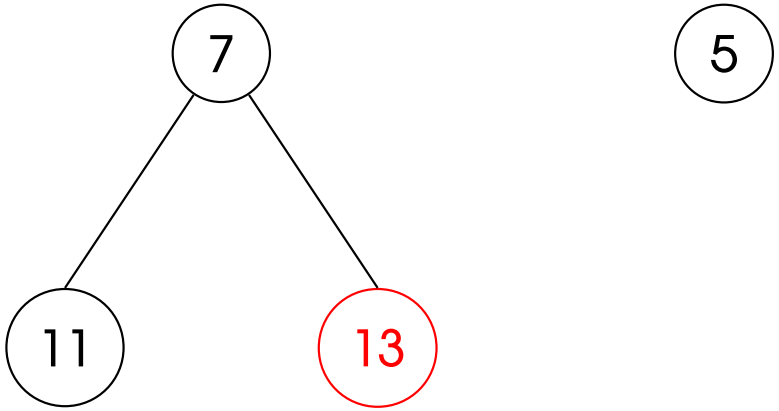
Παράδειγμα

- Σταδιακή επανασύσταση του σωρού



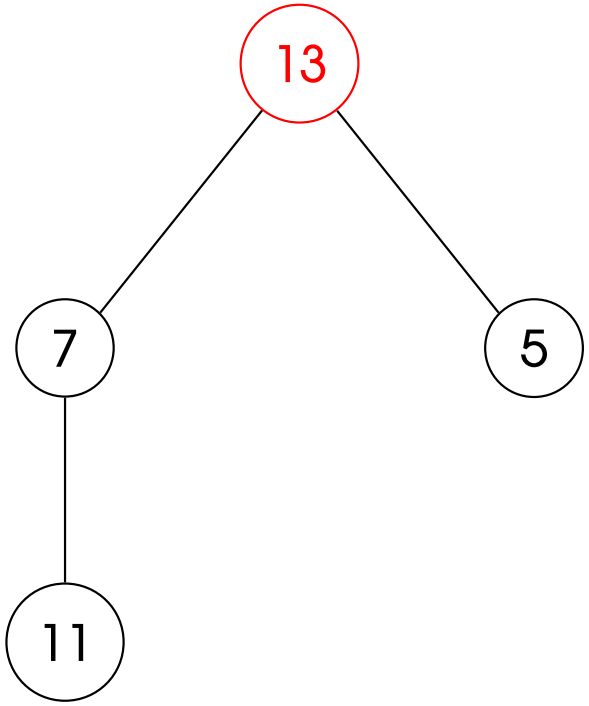
Παράδειγμα

- Σταδιακή επανασύσταση του σωρού



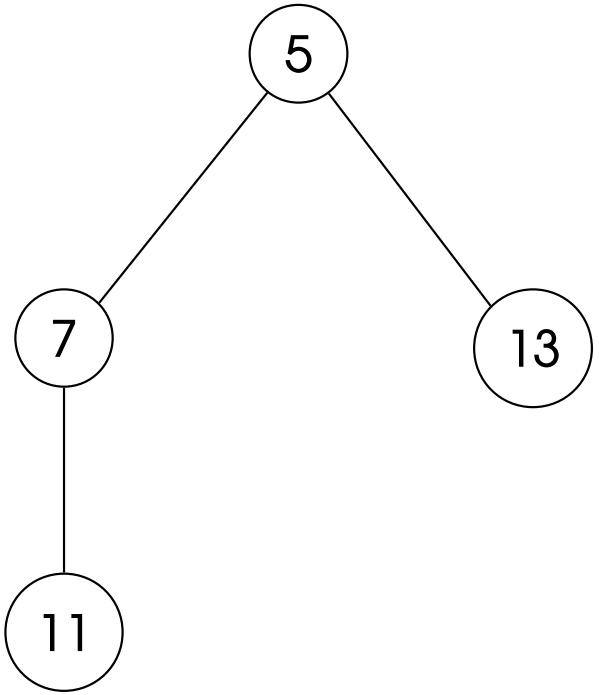
Παράδειγμα

- Μεταφορά τελευταίου στοιχείου στην ρίζα



Παράδειγμα

- Τελευταίο βήμα επαναφοράς της σωρού



Παρατηρήσεις

- Όλες οι ιδιότητες του heapsort διατηρούνται

Παρατηρήσεις

- Όλες οι ιδιότητες του heapsort διατηρούνται
- Με μια σύγκριση, τοποθετούμε το στοιχείο στην ρίζα ενός heap με μικρότερο ύψος

Παρατηρήσεις

- Όλες οι ιδιότητες του heapsort διατηρούνται
- Με μια σύγκριση, τοποθετούμε το στοιχείο στην ρίζα ενός heap με μικρότερο ύψος
- Στα μισά βήματα (iterations) όμως, δεν έχει αλλάξει τίποτα

Χρήση N σωρών - Πλήρης αξιοποίηση

- Χρήση όλων των σωρών σε όλα τα βήματα του αλγορίθμου

Χρήση N σωρών - Πλήρης αξιοποίηση

- Χρήση όλων των σωρών σε όλα τα βήματα του αλγορίθμου
- Σκοπός: Μεγιστοποίηση κέρδους σε απόδοση

Χρήση N σωρών - Πλήρης αξιοποίηση

- Χρήση όλων των σωρών σε όλα τα βήματα του αλγορίθμου
- Σκοπός: Μεγιστοποίηση κέρδους σε απόδοση
- Παράπλευρο κέρδος: Εμφάνιση δυνατοτήτων παραλληλοποίησης

Προβλήματα

- Διατήρηση της ιδιότητας της ταξινόμησης in-place

Προβλήματα

- Διατήρηση της ιδιότητας της ταξινόμησης in-place
- Οργάνωση σωρών σε μορφή πίνακα

Προβλήματα

- Διατήρηση της ιδιότητας της ταξινόμησης in-place
- Οργάνωση σωρών σε μορφή πίνακα
- Αποτελεσματική εύρεση των φύλλων των σωρών

Πρώτη σκέψη

- Ο ένας σωρός μετά τον άλλο ανεξάρτητα

Πρώτη σκέψη

- Ο ένας σωρός μετά τον άλλο ανεξάρτητα
- Αποτυγχάνει

Πρώτη σκέψη

- Ο ένας σωρός μετά τον άλλο ανεξάρτητα
- Αποτυγχάνει
- Μένουν "κενά" ανάμεσα στις σωρούς

Προτεινόμενη Λύση

- Πολύπλεξη των σωρών

Προτεινόμενη Λύση

- Πολύπλεξη των σωρών
- Προσδιορισμός παιδιών
  - Αριστερό παιδί: $2i + N$
  - Δεξί παιδί: $2i + N + 1$
  - Γονιός: $\lfloor \frac{i - N}{2} \rfloor$

Αλγόριθμος

1. Δημιουργία N σωρών heap - (Build-Heap)

Αλγόριθμος

1. Δημιουργία N σωρών heap - (Build-Heap)
2. Εύρεση μικρότερης ρίζας (root)

Αλγόριθμος

1. Δημιουργία N σωρών heap - (Build-Heap)
2. Εύρεση μικρότερης ρίζας (root)
3. Εναλλαγή με το τελευταίο αταξινόμητο στοιχείο

Αλγόριθμος

1. Δημιουργία N σωρών heap - (Build-Heap)
2. Εύρεση μικρότερης ρίζας (root)
3. Εναλλαγή με το τελευταίο αταξινόμητο στοιχείο
4. Κλήση heapify μόνο στο ένα δέντρο

Αλγόριθμος

1. Δημιουργία N σωρών heap - (Build-Heap)
2. Εύρεση μικρότερης ρίζας (root)
3. Εναλλαγή με το τελευταίο αταξινόμητο στοιχείο
4. Κλήση heapify μόνο στο ένα δέντρο
5. Επανάληψη της διαδικασίας από το 2ο βήμα μέχρι να ταξινομηθούν όλα τα στοιχεία

Ανάλυση Κόστους

- $N - 1$ συγκρίσεις για την εύρεση της μικρότερης ρίζας

Ανάλυση Κόστους

- $N - 1$ συγκρίσεις για την εύρεση της μικρότερης ρίζας
- $\log(n + N) - \log N$ το ύψος κάθε heap

Ανάλυση Κόστους

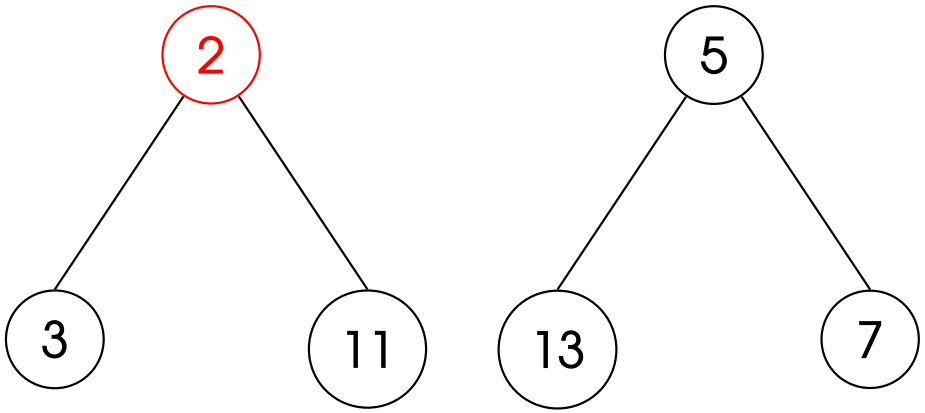
- $N - 1$ συγκρίσεις για την εύρεση της μικρότερης ρίζας
- $\log(n + N) - \log N$ το ύψος κάθε heap
- Κλήση heapify μόνο στο ένα δέντρο

Ανάλυση Κόστους

- $N - 1$ συγκρίσεις για την εύρεση της μικρότερης ρίζας
- $\log(n + N) - \log N$ το ύψος κάθε heap
- Κλήση heapify μόνο στο ένα δέντρο
- Κέρδος περίπου 10% σε ομοιόμορφη κατανομή τυχαίων τιμών με σχετικά μεγάλο πλήθος στοιχείων

Παράδειγμα

Σε μορφή δέντρου:

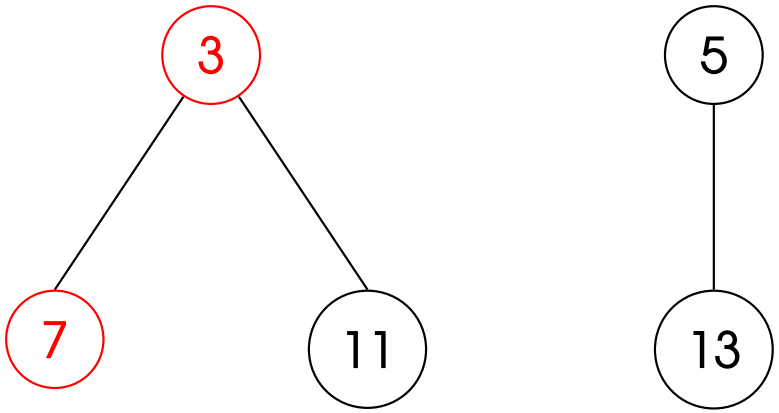


Σε μορφή πίνακα:

2	5	3	11	13	7

Παράδειγμα

Σε μορφή δέντρου:

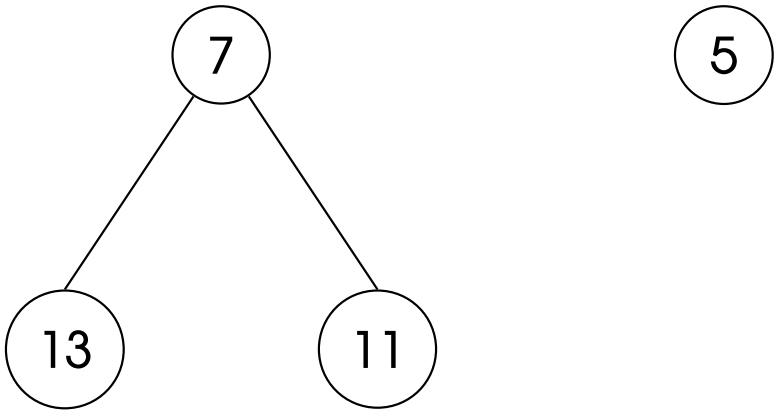


Σε μορφή πίνακα:

3	5	7	11	13	2

Παράδειγμα

Σε μορφή δέντρου:

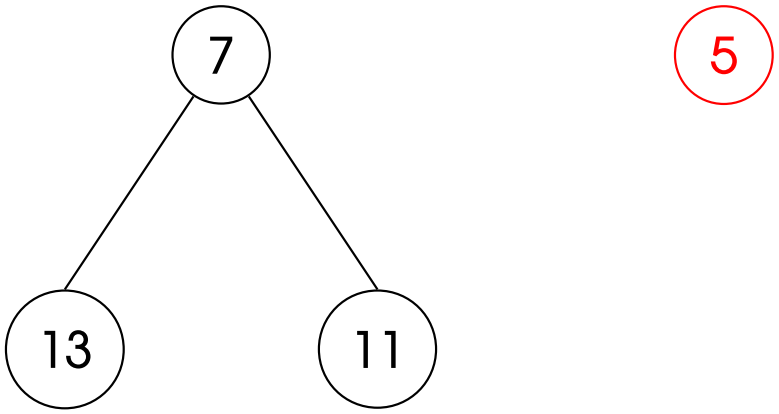


Σε μορφή πίνακα:

7	5	13	11	3	2

Παράδειγμα

Σε μορφή δέντρου:

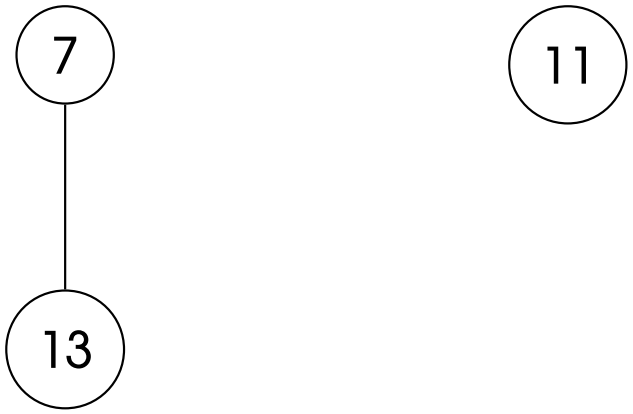


Σε μορφή πίνακα:

7	5	13	11	3	2

Παράδειγμα

Σε μορφή δέντρου:



Σε μορφή πίνακα:

7	11	13	5	3	2

Κόστος

- Συνεχίζοντας, θα πάρουμε το τελικό αποτέλεσμα με 9 συγκρίσεις και 6 εναλλαγές στοιχείων

Κόστος

- Συνεχίζοντας, θα πάρουμε το τελικό αποτέλεσμα με 9 συγκρίσεις και 6 εναλλαγές στοιχείων
-

σύγκριση	εναλλαγή
$2 < 5$	$2 \leftrightarrow 7$
$3 < 11, 7 > 3$	$3 \leftrightarrow 7$
$3 < 5$	$4 \leftrightarrow 13$
$7 < 11, 13 > 7$	$7 \leftrightarrow 13$
$5 < 7$	$5 \leftrightarrow 11$
$7 < 11$	$7 \leftrightarrow 13$
$11 < 13$	$--$

Κόστος Κλασικού Αλγορίθμου

- Χρησιμοποιώντας τον κλασικό αλγόριθμο, χρειαζόμαστε 8 συγκρίσεις και 8 εναλλαγές

Κόστος Κλασικού Αλγορίθμου

- Χρησιμοποιώντας τον κλασικό αλγόριθμο, χρειαζόμαστε 8 συγκρίσεις και 8 εναλλαγές

σύγκριση	εναλλαγή
$--$	$2 \leftrightarrow 11$
$3 < 5, 11 > 3$	$3 \leftrightarrow 11$
$--$	$7 \leftrightarrow 3$
$5 < 11, 5 < 7$	$7 \leftrightarrow 5$
$7 < 13$	$--$
$--$	$5 \leftrightarrow 13$
$7 < 11, 7 < 13$	$7 \leftrightarrow 13$
$--$	$7 \leftrightarrow 11$
$11 < 13$	$--$
$--$	$11 \leftrightarrow 13$

Επίλογος

- Εισάγαμε μια νέα ιδέα

Επίλογος

- Εισάγαμε μια νέα ιδέα
- Επιτύχαμε επιτάχυνση στην εξαγωγή του τελικού αποτελέσματος

Επίλογος

- Εισάγαμε μια νέα ιδέα
- Επιτύχαμε επιτάχυνση στην εξαγωγή του τελικού αποτελέσματος
- Εισάγονται δυνατότητες παραλληλοποίησης

Επίλογος

- Εισάγαμε μια νέα ιδέα
- Επιτύχαμε επιτάχυνση στην εξαγωγή του τελικού αποτελέσματος
- Εισάγονται δυνατότητες παραλληλοποίησης
- Μπορεί να χρησιμοποιηθεί και σε παραλλαγές του heapsort

Ευχαριστώ