Διάλεξη 28: Ο Αλγόριθμος Ταξινόμησης HeapSort

Στην ενότητα αυτή θα μελετηθούν τα εξής επιμέρους θέματα:

- Η διαδικασία PercolateDown, Δημιουργία Σωρού
- Ο Αλγόριθμος Ταξινόμησης HeapSort
- Υλοποίηση, Παραδείγματα

Διδάσκων: Παναγιώτης Ανδρέου

Διαδικασία Καθόδου PercolateDown

- Έστω ένας πίνακας A[1..n] και μια τιμή i, θα ορίσουμε διαδικασία PercoladeDown(i), η οποία μετακινεί το στοιχείο A[i] μέσα στον σωρό προς τα κάτω όσο χρειάζεται.
- Έστω ότι A[i] = k.
- Θεωρούμε πως η i είναι άδεια θέση.
- Αν η άδεια θέση έχει παιδί που περιέχει στοιχείο μικρότερο του k και x είναι το μικρότερο τέτοιο παιδί, τότε μετακινούμε το στοιχείο του x στην κενή θέση και μετακινούμε την κενή θέση στο x.
- Επαναλαμβάνουμε την ίδια διαδικασία μέχρι τη στιγμή που η κενήθέση δεν έχει παιδιά με στοιχεία μικρότερα του k. Τότε αποθηκεύουμε το k στην θέση αυτή.
- Ο χρόνος εκτέλεσης είναι ανάλογος του ύψους του κόμβου που αντιστοιχεί στη θέση i του σωρού. Δηλαδή, στη χείριστη περίπτωση, όπου i=n, O(lg n).

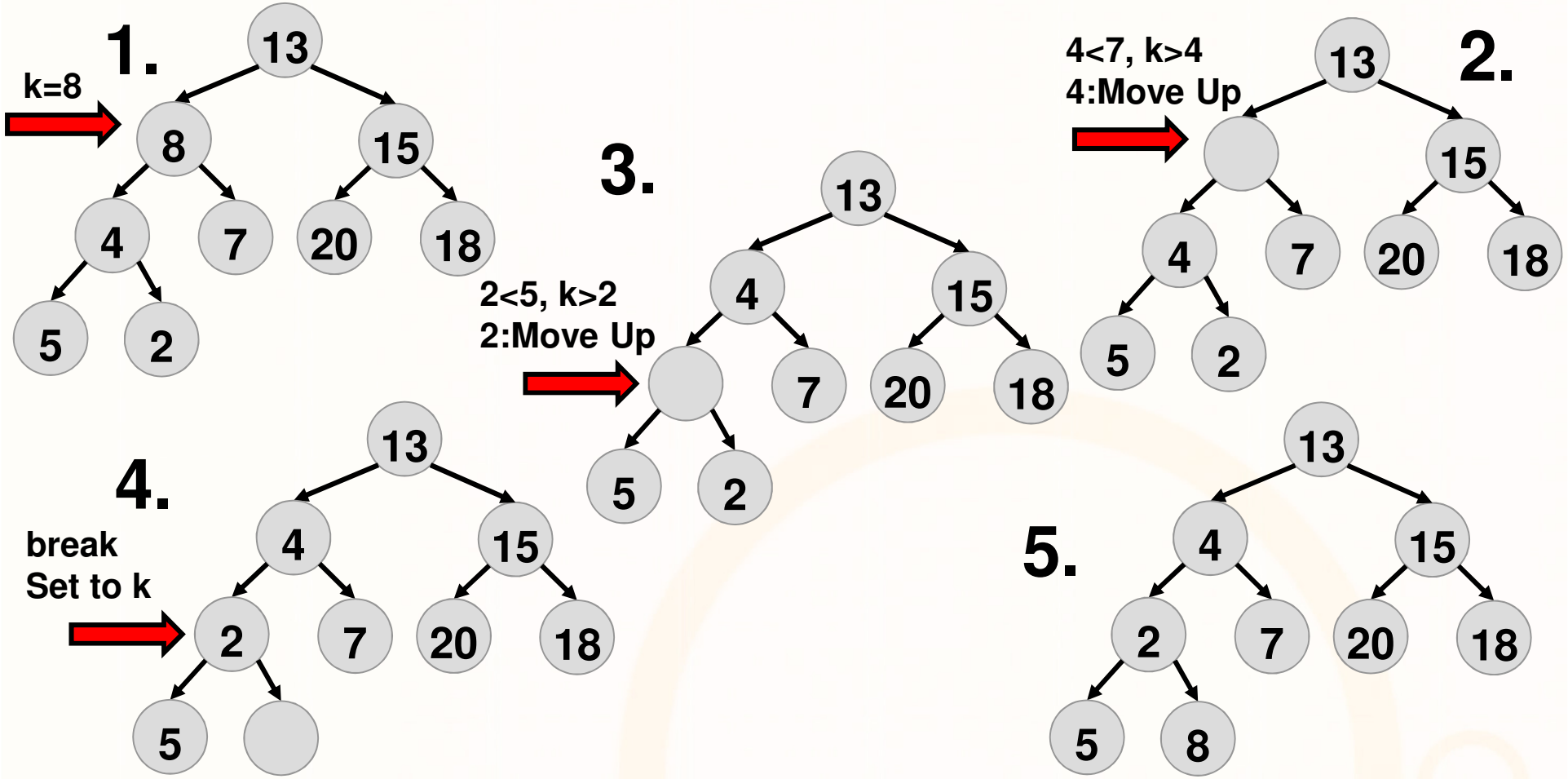
Διαδικασία Καθόδου PercolateDown (συν.)

- Μη αναδρομική διαδικασία PercolateDown

```
void PercolateDown(type A[], int n, int i) {
  type k = A[i];
  int j;
  while( 2*i <=n ) {
    j = 2*i;
    if (j<n && A[j+1]<A[j]) j++;
    if (k > A[j]) {
      A[i] = A[j];
      i=j;
    }
    else break;
   }
   A[i] = k;
}
```

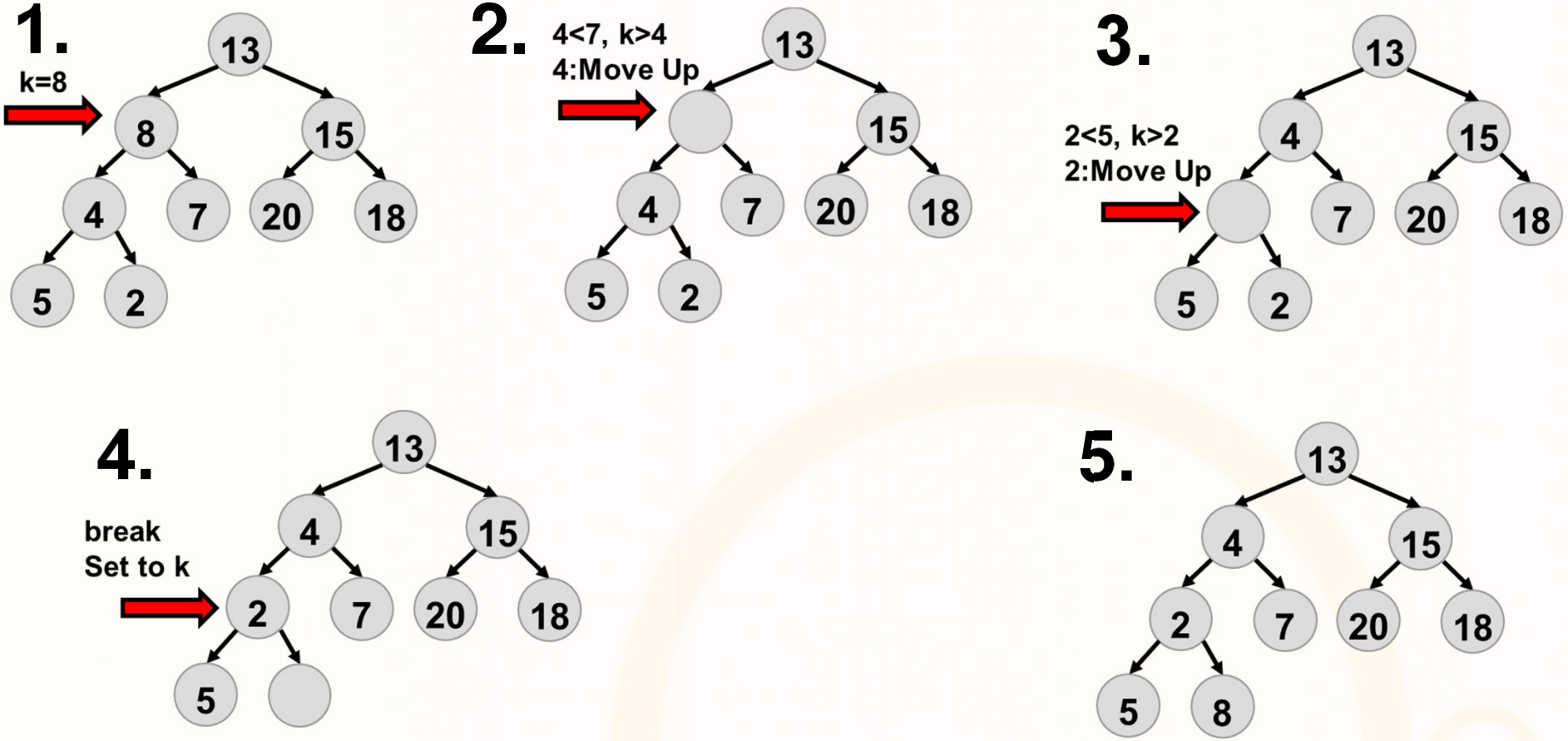
Παράδειγμα Εκτέλεσης PercolateDown

- `int A[]={-1 , 13, 8, 15, 4, 7, 20, 18, 5, 2};`
- `i=2, n=9, PercolateDown(A, 9, 2);` ➔ `k=8`
- Αν φανταστούμε τον πίνακα σαν δυαδικό δέντρο…



Παράδειγμα Εκτέλεσης PercolateDown

- `int A[]={-1 , 13, 8, 15, 4, 7, 20, 18, 5, 2};`
- `i=2, n=9, PercolateDown(A, 9, 2);` ➔ `k=8`
- Αν φανταστούμε τον πίνακα σαν δυαδικό δέντρο…



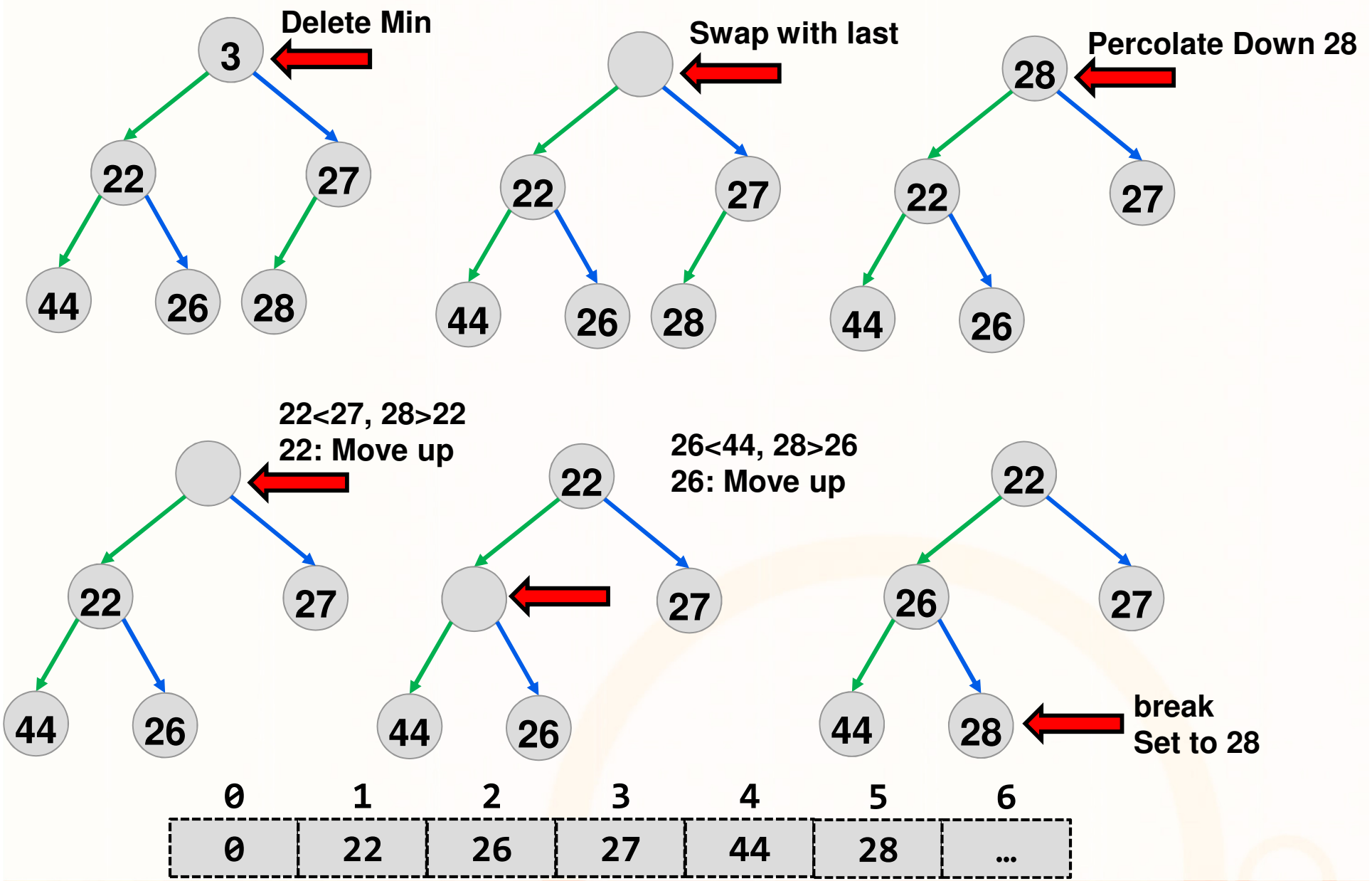
Διαδικασία DeleteMin (2)

- Αφαιρούμε το στοιχείο της ρίζας (είναι το μικρότερο κλειδί του σωρού).
- Μεταφέρουμε το τελευταίο κλειδί στη ρίζα, και εφαρμόζουμε τη διαδικασία PercoladeDown(A, n, 1):

```
type DeleteMin(HEAP* heap) {
  type min=NULL;
  type swap=NULL;
  if(!IsEmptyHeap(heap)){
    min = heap->contents[1];
    swap(contents[1],contents[size]);
    heap->size --;
    PercolateDown(heap->contents, heap->size, 1);
  }
  return min;
}
```

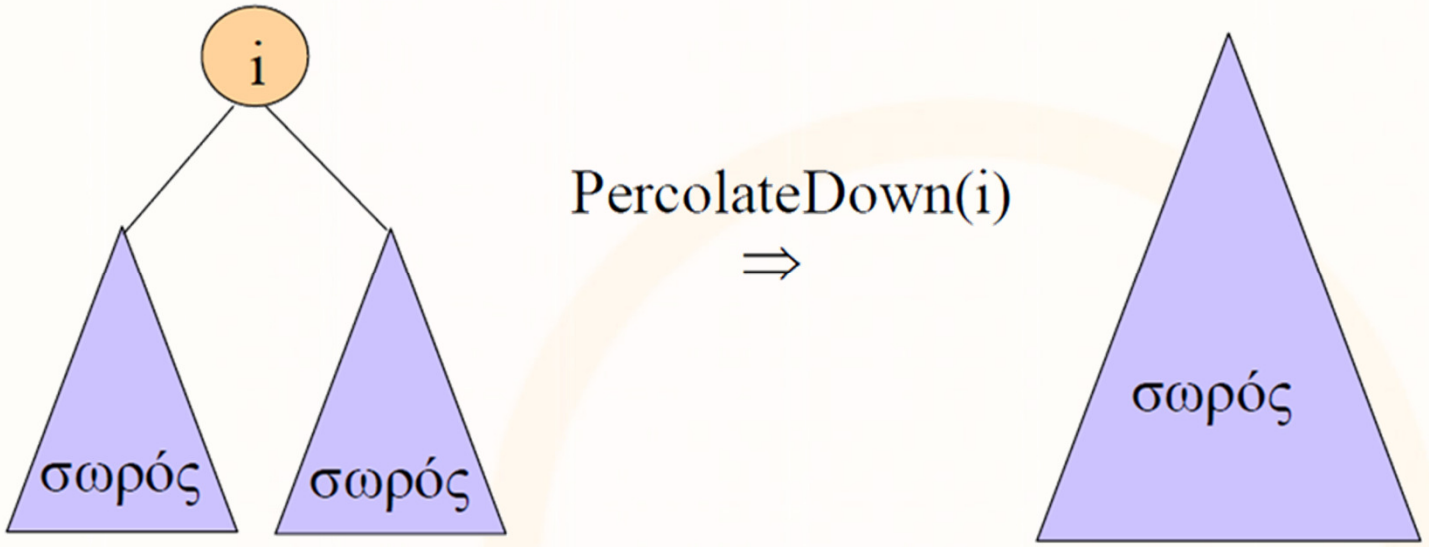
- Χρόνος Εκτέλεσης: O(h) = O(log n)

Παράδειγμα DeleteMin



Από πίνακες σε σωρούς

- Έστω πίνακας A[1..n].
- Μπορούμε να θεωρήσουμε τον πίνακα ως ένα πλήρες δυαδικό δένδρο με n κόμβους.
- Αν για μια τιμή i το αριστερό και το δεξί υπόδενδρο του i ικανοποιούν τις ιδιότητες ενός σωρού, τότε, αν καλέσουμε τη διαδικασία PercolateDown(A, n, i) θα έχουμε σαν αποτέλεσμα το υπόδενδρο που ριζώνει στη θέση i να ικανοποιεί τις ιδιότητες ενός σωρού.



Κτίσιμο σωρού από ένα πίνακα

- Μπορούμε να μετατρέψουμε ένα πίνακα A[1..n] σε ένα σωρό με διαδοχική εφαρμογή της διαδικασίας PercoladeDown() από κάτω προς τα πάνω.
- **Παρατήρηση:** οι θέσεις > n/2 αντιστοιχούν σε φύλλα.

```
void BuildHeap( int A[], int n) {
  for (int i=n/2; i>0; i--)
  PercolateDown(A,n,i);
}
```

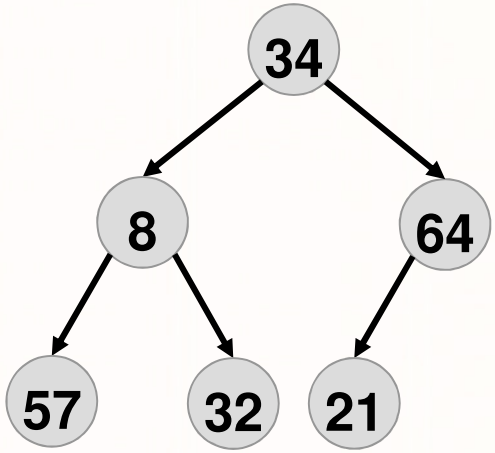
- **Ορθότητα (αποδεικνύεται με τη μέθοδο της επαγωγής):** μετά από την εφαρμογή της διαδικασίας PercoladeDown(A,n,i), τα υπόδενδρα που ριζώνουν στις θέσεις i, ..., n, ικανοποιούν τις ιδιότητες σωρού.
- **Ανάλυση του Χρόνου Εκτέλεσης:** Ο ολικός χρόνος εκτέλεσης είναι ανάλογος του αθροίσματος των υψών όλων των εσωτερικών κόμβων, το οποίο είναι O(n).

Τι κάνει ο πιο κάτω αλγόριθμος;

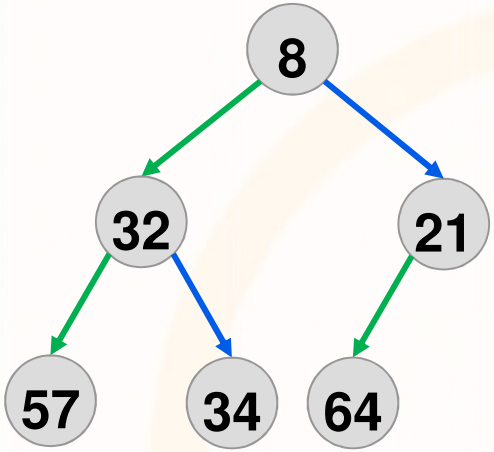
```
void mystery (type A[], int n) {
  BuildHeap(A, n);
  type swap;
  for (int i=n; i>1; i--){
    swap (A[1], A[i]);
    PercolateDown(A, i-1, 1);
  }
}
```

Παράδειγμα Εκτέλεσης Διαδικασίας mystery

- Είσοδος, A={ -, 34, 8, 64, 57, 32, 21}

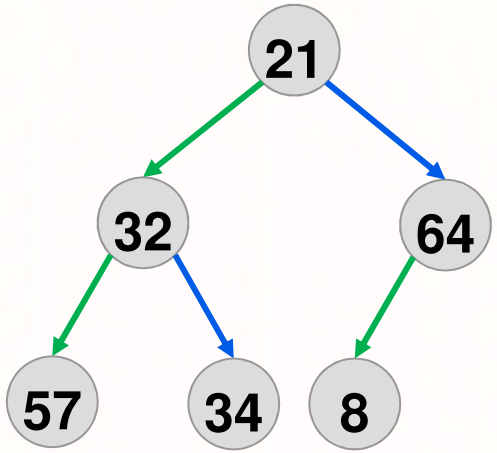


- Μετά την εκτέλεση της γραμμής BuildHeap

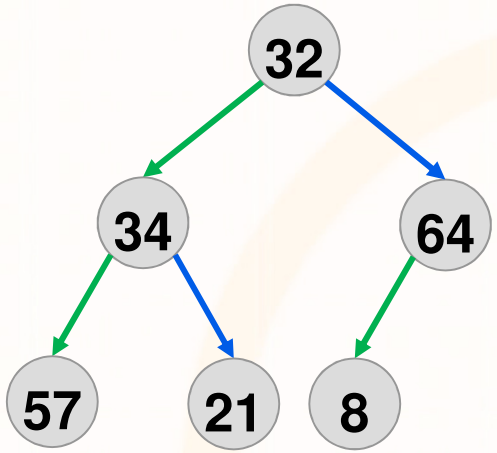


Παράδειγμα Εκτέλεσης Διαδικασίας mystery

- Μετά από την πρώτη επανάληψη του for

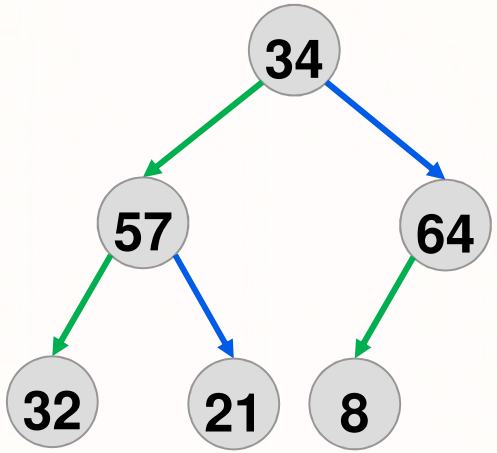


- Μετά από την δεύτερη επανάληψη του for

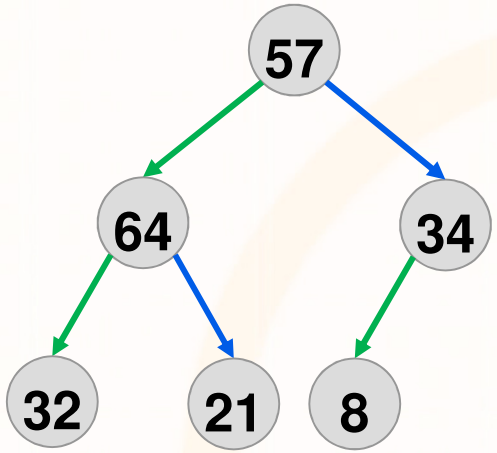


Παράδειγμα Εκτέλεσης Διαδικασίας mystery

- Μετά από την τρίτη επανάληψη του for

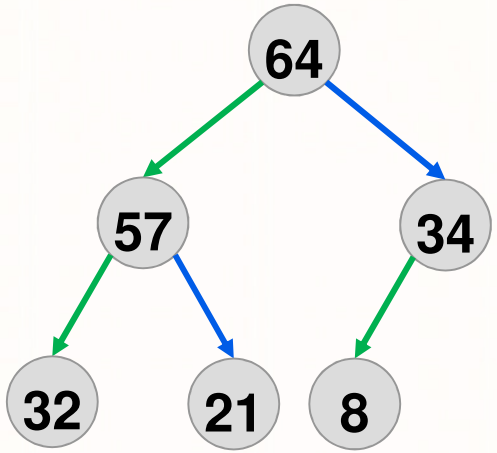


- Μετά από την τέταρτη επανάληψη του for



Παράδειγμα Εκτέλεσης Διαδικασίας mystery

- Μετά από την πέμπτη επανάληψη του for



- Σε επίπεδο πίνακα

-	64	57	34	32	21	8

Ο αλγόριθμος ταξινόμησης HeapSort

- Η διαδικασία mystery ταξινομεί ένα πίνακα σε φθίνουσα σειρά.
- Αρχικά δημιουργεί ένα σωρό σε χρόνο O(n).
- Στη συνέχεια επαναλαμβάνει το εξής: αφαιρεί το μικρότερο στοιχείο (της ρίζας του σωρού) και το μετακινεί στο τέλος (εκτελεί την PercolateDown). Κάθε εκτέλεση της PercoladeDown χρειάζεται χρόνο της τάξης O(log n).
- Ολικός Χρόνος Εκτέλεσης: O(n·log n)

- **Ο αλγόριθμος ονομάζεται Heapsort**

- Μπορούμε εύκολα να αλλάξουμε τον κώδικα ώστε να επιστρέφεται η λίστα σε αύξουσα σειρά.

Άλλες διαδικασίες σε σωρούς

- Παρόλο που εύρεση του ελάχιστου κλειδιού σε ένα σωρό μπορεί να πραγματοποιηθεί σε σταθερό χρόνο, η εύρεση τυχαίου στοιχείου στη χειρότερη περίπτωση επιβάλλει διερεύνηση ολόκληρης της δομής (δηλαδή, είναι της τάξης O(n)).
- Αν όμως γνωρίζουμε τη θέση στοιχείων με κάποιο άλλο τρόπο, διαδικασίες σε σωρούς πραγματοποιούνται εύκολα, π.χ. οι πιο κάτω εκτελούνται σε χρόνο λογαριθμικό.
- Increase_Key(Ρ,Δ), αυξάνει την προτεραιότητα του κλειδιού Ρ, κατά Δ. Χρησιμοποιείται από χειριστές λειτουργικών συστημάτων για αύξηση της προτεραιότητας σημαντικών διεργασιών. Η συμμετρική διαδικασία Decrease_Key(Ρ,Δ) συχνά εκτελείται αυτόματα σε λειτουργικά συστήματα σε περίπτωση που κάποια δουλειά χρησιμοποιεί υπερβολικά μεγάλη ποσότητα χρόνου του CPU.
- Remove(I), αφαιρεί τον κόμβο της θέσης I (χρήσιμη σε περίπτωση τερματισμού διαδικασίας).