PRESENTASI

HEAP SHORT

KELOMPOK 4

- Ujang Herlan
- Dean Satria
- Gepira Nur Patimah
- Lukas febrian
- Rizki Ramadhan
- Rahmat Hidayat
- Alfian
- Taufik Hidayat

HEAP SHORT

Heap sort adalah algoritma pengurutan yang menggunakan struktur data heap dalam implementasinya. Dalam heap sort, array diubah menjadi heap dengan membangun struktur data heap. Kemudian, elemen terbesar secara berulang dihapus dari heap dan ditempatkan di akhir array yang belum terurut. Proses ini diulang hingga seluruh array terurut. Heap sort memiliki kompleksitas waktu O(n log n) di mana n adalah jumlah elemen dalam array.

AlLGORITMA DARI HEAP SHORT

MaxHeapify(arr,i)

```
MaxHeapify(arr,i)
L = left(i)
R = right(i)
if L ? heap_size[arr] and arr[L] > arr[i]
largest = L
else
largest = i
if R ? heap_size[arr] and arr[R] > arr[largest]
largest = R
if largest != i
swap arr[i] with arr[largest]
MaxHeapify(arr,largest)
End
```

BuildMaxHeap(arr)

```
BuildMaxHeap(arr)
    heap_size(arr) = length(arr)
    for i = length(arr)/2 to 1
MaxHeapify(arr,i)
End
```

Algorithm

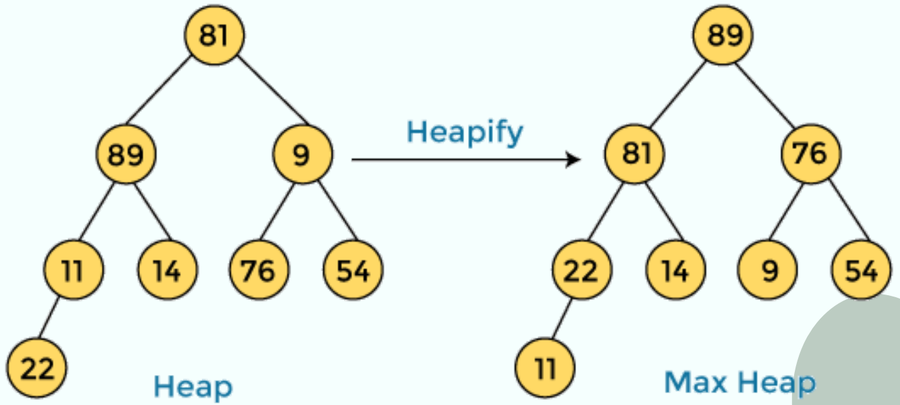
```
HeapSort(arr)
BuildMaxHeap(arr)
for i = length(arr) to 2
    swap arr[1] with arr[i]
        heap_size[arr] = heap_size[arr] ? 1
        MaxHeapify(arr,1)
End
```

Langkah-Langkah

Misalkan terdapat suatu array bilangan yang terdiri dari delapan buah anggota dengan nilai data 81, 89, 9, 11, 14, 76, 54, 22. Kita akan mengurutkan data diatas dengan menggunakan heapsort.

81	89	9	11	14	76	54	22

Pertama, kita harus membuat heap dari array yang diberikan dan mengubahnya menjadi max heap.

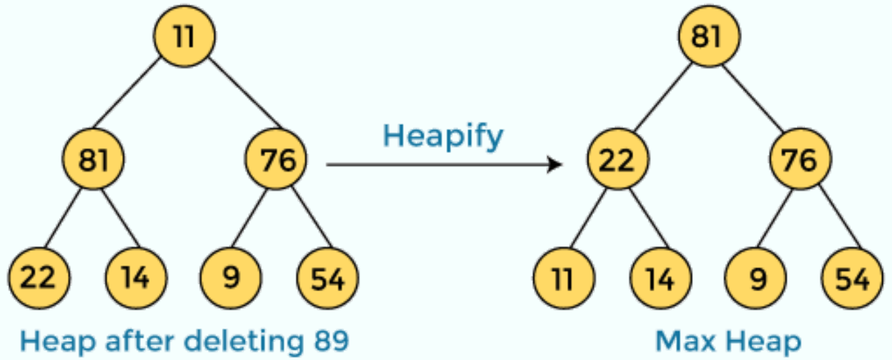


Setelah mengubah heap yang diberikan menjadi max heap, elemen array adalah -

Langkah-Langkah

89	81	76	22	14	9	54	11

Selanjutnya, kita harus menghapus elemen root (89) dari max heap. Untuk menghapus simpul ini, kita harus menukarnya dengan simpul terakhir, yaitu (11). Setelah menghapus elemen root, sekali lagi kita harus menimbunnya untuk mengubahnya menjadi tumpukan maksimal.

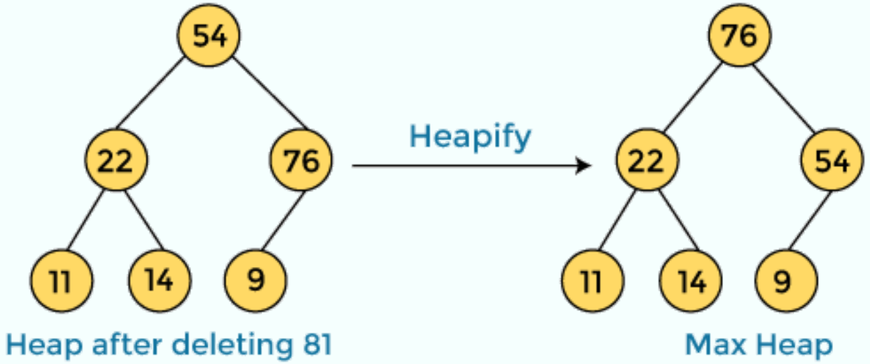


Setelah menukar elemen array 89 dengan 11, dan mengubah heap menjadi max-heap, elemen array adalah -

Langkah-Langkah

81	22	76	11	14	9	54	89

Pada langkah selanjutnya, sekali lagi, kita harus menghapus elemen root (81) dari max heap. Untuk menghapus simpul ini, kita harus menukarnya dengan simpul terakhir, yaitu (54). Setelah menghapus elemen root, sekali lagi kita harus menimbunnya untuk mengubahnya menjadi tumpukan maksimal.

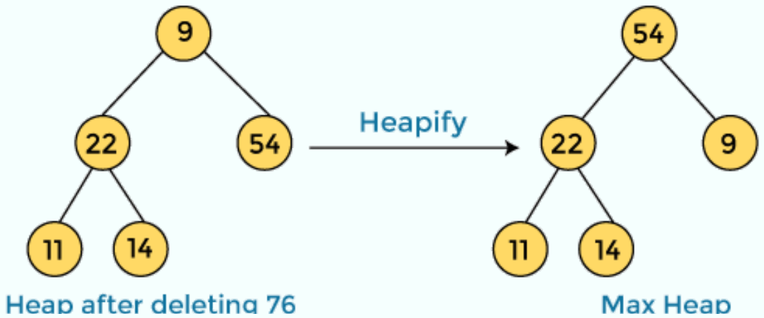


Setelah menukar elemen array 81 dengan 54 dan mengubah heap menjadi max-heap, elemen array adalah -

Langkah-Langkah

76	22	54	11	14	9	81	89

Pada langkah selanjutnya, kita harus menghapus elemen root (76) dari max heap lagi. Untuk menghapus simpul ini, kita harus menukarnya dengan simpul terakhir, yaitu (9). Setelah menghapus elemen root, sekali lagi kita harus menimbunnya untuk mengubahnya menjadi tumpukan maksimal.

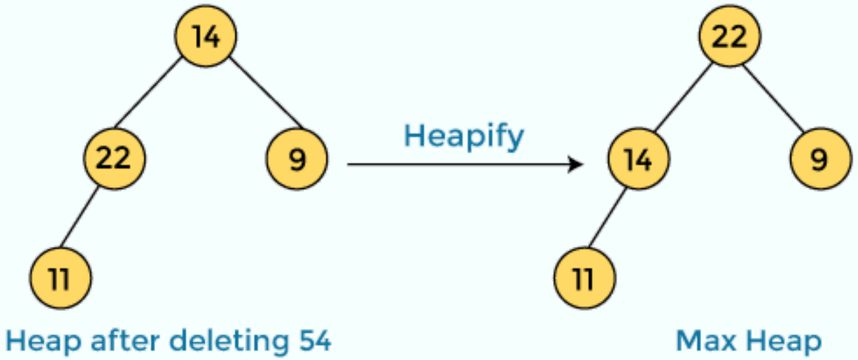


Setelah menukar elemen array 76 dengan 9 dan mengubah heap menjadi max-heap, elemen array adalah -

Langkah-Langkah

54	22	9	11	14	76	81	89

Pada langkah selanjutnya, sekali lagi kita harus menghapus elemen root (54) dari max heap. Untuk menghapus simpul ini, kita harus menukarnya dengan simpul terakhir, yaitu (14). Setelah menghapus elemen root, sekali lagi kita harus menimbunnya untuk mengubahnya menjadi tumpukan maksimal.

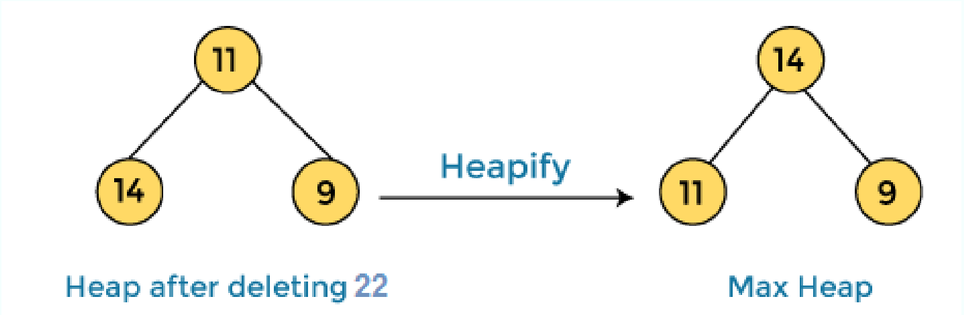


Setelah menukar elemen array 54 dengan 14 dan mengubah heap menjadi max-heap, elemen array adalah -

Langkah-Langkah

22	14	9	11	54	76	81	89

Pada langkah selanjutnya, sekali lagi kita harus menghapus elemen root (22) dari max heap. Untuk menghapus simpul ini, kita harus menukarnya dengan simpul terakhir, yaitu (11). Setelah menghapus elemen root, sekali lagi kita harus menimbunnya untuk mengubahnya menjadi tumpukan maksimal.



Setelah menukar elemen array 22 dengan 11 dan mengubah heap menjadi max-heap, elemen array adalah -

Langkah-Langkah

14	11	9	22	54	76	81	89

Pada langkah selanjutnya, sekali lagi kita harus menghapus elemen root (14) dari max heap. Untuk menghapus simpul ini, kita harus menukarnya dengan simpul terakhir, yaitu (9). Setelah menghapus elemen root, sekali lagi kita harus menimbunnya untuk mengubahnya menjadi tumpukan maksimal.

Setelah menukar elemen array 14 dengan 9 dan mengubah heap menjadi max-heap, elemen array adalah -

Langkah-Langkah

11	9	14	22	54	76	81	89

Pada langkah selanjutnya, sekali lagi kita harus menghapus elemen root (11) dari max heap. Untuk menghapus simpul ini, kita harus menukarnya dengan simpul terakhir, yaitu (9). Setelah menghapus elemen root, sekali lagi kita harus menimbunnya untuk mengubahnya menjadi tumpukan maksimal.

Setelah menukar elemen array 11 dengan 9, elemen array adalah -

9	11	14	22	54	76	81	89

Sekarang, heap hanya memiliki satu elemen tersisa. Setelah menghapusnya, heap akan kosong.

Langkah-Langkah

Setelah selesai menyortir, elemen array adalah -

9	11	14	22	54	76	81	89

Sekarang, array sepenuhnya diurutkan.

```java
public class HeapSort {
    public static void heapSort(int[] arr) {
        int n = arr.length;

        // Membangun heap maksimum
        for (int i = n / 2 - 1; i >= 0; i--)
            heapify(arr, n, i);

        // Menyortir array dengan mempertahankan properti heap
        for (int i = n - 1; i > 0; i--) {
            // Menukar elemen teratas yang merupakan elemen terbesar dengan elemen terakhir
            int temp = arr[0];
            arr[0] = arr[i];
            arr[i] = temp;

            // Memanggil heapify pada subarray yang belum terurut
            heapify(arr, i, 0);
        }
    }

    public static void heapify(int[] arr, int n, int i) {
        int largest = i; // Inisialisasi node terbesar sebagai root
        int leftChild = 2 * i + 1; // Indeks anak kiri dalam array
        int rightChild = 2 * i + 2; // Indeks anak kanan dalam array

        // Jika anak kiri lebih besar dari root
        if (leftChild < n && arr[leftChild] > arr[largest])
            largest = leftChild;

        // Jika anak kanan lebih besar dari root
        if (rightChild < n && arr[rightChild] > arr[largest])
            largest = rightChild;

        // Jika node terbesar bukan lagi root
        if (largest != i) {
            // Menukar elemen root dengan elemen terbesar
            int swap = arr[i];
            arr[i] = arr[largest];
            arr[largest] = swap;

            // Melakukan rekursi pada subarray yang terpengaruh
            heapify(arr, n, largest);
        }
    }

    public static void main(String[] args) {
        int[] arr = {12, 11, 13, 5, 6, 7};
        heapSort(arr);

        System.out.println("Hasil pengurutan:");
        for (int i : arr)
            System.out.print(i + " ");
    }
}
```

https://drive.google.com/file/d/1IhwEdipeBbdslsr1o1a1D11Ns9Rxo00U/view?usp=drivesdk

TERIMA KASIH