NAMA KELOMPOK 5

1. Nila Enjeli
2. Salsabila Okta Kirana
3. Hadifta Atallah Putri
4. Muhammad Ikhsan

Definisi Heap Sort

Heap Sort

adalah sebuah algoritma pengurutan yang paling lambat dari algoritma yangmemiliki kompleksitas O(n log n). Tetapi tidak seperti algoritma Merge Sort dan Quick Sort,algoritma Heap Sort tidak memerlukan rekursif yang besar atau menggunakan banyak tabel(array). Oleh karena itu, Heap Sort adalah pilihan yang baik untuk sebuah kumpulan data yang besar. Algoritma ini dimulai dengan membangun sebuah array heap dengan membangu ntumpukan dari kumpulan data, lalu memindahkan data terbesar ke bagian belakang dari sebuahtabel hasil. Setelah itu, array heap dibangun kembali, kemudian mengambil elemen terbesaruntuk diletakkan di sebelah item yang telah dipindahkan tadi. Hal ini diulang sampai array heaphabis. Jadi secara umum, algoritma ini memerlukan dua buah tabel; satu tabel untukmenyimpan heap, dan satu tabel lainnya untuk menyimpan hasil. Walaupun lebih lambat dariMerge Sort atau Quick Sort, algoritma ini cocok untuk digunakan pada data yang berukuran besar. Atau

dilakukan dengan cara membangun struktur data heap dan kemudian menerapkanlangkah menghapus node heap. Pada awalnya, kondisi heap kosong. Kemudian, node heap di-insert satu per satu. Cara alternatif adalah, menampung data yang akan diurutkan dalam array,kemudian node pada bagian root di hapus. Jika heap adalah minimal heap, maka data pada rootadalah data terkecil, terdapat 2 contoh heap yaitu MAXHEAP (nilai orangtua≥ nilai anaknya)MIN HEAP (nilai orangtua≤ nilai anaknya).

Contoh Algoritma Heap Sort

Pseudocode:

Judul:

program_heap_sort

Deklarasi:

Integer= A[100], i, length, le, ri, heapsize,largest

Deskripsi:

```
Heapsort(A) {
    BuildHeap(A)
    for i <- length(A) downto 2 {
        exchange A[1] <->A[i]
        heapsize <-heapsize – 1
        Heapify(A,1)
}
BuildHeap(A) {
heapsize <- length(A)for i <- floor( length/2 ) downto 1
Heapify(A, i)
}
Heapify(A, i) {
le <- left(i)
ri <- right(i)
if (le<=heapsize) and (A[le]>A[i])
largest <- le
else
largest <- i
if (ri<=heapsize) and (A[ri]>A[largest])
largest <- ri
if (largest != i) {
exchange A[1] <->A[largest]
Heapify(A,largest)
}
```

Heap

Pengertian Heap

Pohon heap adalah struktur data yang berbentuk pohon yang memenuhi sifat-sifat heap yaitu jika B adalah anak dari A, maka nilai yang tersimpan di simpul A lebih besar atau sama dengan nilai yang tersimpan di simpul B. Hal ini mengakibatkan elemen dengan nilai terbesar selalu berada pada posisi akar, dan heap ini disebut max heap. (Bila perbandingannya diterbalikkan yaitu elemen terkecilnya selalu berada di simpul akar, heap ini disebut adalah min heap). Karena itulah, heap biasa dipakai untuk mengimplementasikan priority queue. Operasi-operasi yang digunakan untuk heap adalah :

• Delete-max atau delete-min: menghapus simpul akar dari sebuah max- atau minheap.

• Increase-key atau decrease-key : mengubah nilai yang tersimpan di suatu simpul.

• Insert: menambahkan sebuah nilai ke dalam heap.

• Merge: menggabungkan dua heap untuk membentuk sebuah heap baru yang berisi semua elemen pembentuk heap tersebut.

Program Heap Sort & Penjelasannya

```
<#include <>

void restoreHup(int*,int); // pemanggilan fungsi void restoreHup

void restoreHdown(int*,int,int); //pemanggilan fungsi void restoreHdown

void main()

{ // pembuka void main

int a[20],n,i,j,k; // mendeklarasikan bahwa a[20] ,n,i,j,k adalah integer

printf(" Masukkan jumlah element : ");// untuk menampilkan kelayar perintah memasukkan jumlah element

scanf("%d",&n); // untuk mengidentifikasikan nilai yang dimasukkan melalui keyboard
```

printf(" Masukkan element : "); //untuk menampilkan kelayar perintah untuk memasukkan element

for(i=1;i<=n;i++) //funsi for dimana jika ketentuan untuk i terpenuhi maka progran di bawahnya akan dijalankan

{ // pembuka fungsi for

scanf("%d",&a[i]); // untuk mengidentifikasi array a

restoreHup(a,i); // a , i dalam fungsi restoreHup

} // penutup fungsi for

j=n; // nilai j sama dengan n

for(i=1;i<=j;i++) //funsi for dimana jika ketentuan untuk i terpenuhi maka progran di bawahnya akan dijalankan

{ // pembuka fungsi for

int temp; // temp sebagai integer

temp=a[1]; // temp sama dengan nilai array a yang pertama

a[1]=a[n]; // nilai array a yg ke 1 sama dengan array a yang ke n

a[n]=temp; // nilai array a yang ke n sama dengan nilay temp

n--; // nilai n selalu berkurang 1

restoreHdown(a,1,n); // a , 1, n dalam fungsi restoreHdown

} // penutup fungsi for

n=j; // n sama dengan nilai j

printf(" Here is it... "); // untuk menampilkan perintah ke dalam layar

for(i=1;i<=n;i++) //funsi for dimana jika ketentuan untuk i terpenuhi maka progran di bawahnya akan dijalankan

printf("%4d",a[i]); // untuk menampilkan nilai array ke i ke layer

```
} // penutup void main

void restoreHup(int *a,int i) // fungsi void restore Hup

{ // pembuka fungsi foid restoreHup

int v=a[i]; // v sama dengan nilai array a yang ke i

while((i>1)&&(a[i/2]

{ // pembuka fungsi while

a[i]=a[i/2]; // nilai array a yang ke i sama dengan nilai array a yang ke i/2

i=i/2; // nilai i sama dengan nilai i/2

} //penutup fungsi while

a[i]=v; // nilai array a yang ke i sama dengan nilai v

} // penutup fungsi while

void restoreHdown(int *a,int i,int n) // fungsi void restoreHdown

{ // pembuka fungsi void restoreHdown

int v=a[i]; // v sama dengan nilai array a yang ke i sebagai integer

int j=i*2;// nilai j sama dengan i kali 2 ialah integer

while(j<=n) // fungsi while akan dijalankan bila ketentuannya terpenuhi

{ // pembuka fungsi while

if((j

j++; // nilai j akan selalu tambah 1

if(a[j]

break;

a[j/2]=a[j]; // nilai array a yang ke j/2 sama dengan nilai array a yang ke j

j=j*2; // nilai j sama dengan nilai j*2
```

}// penutup fungsi while

a[j/2]=v;// nilai array a yang ke j/2 sama dengan v

}// penutup fungsi void restorehdown

//Suatu program untuk mengimplementasikan Heap Sort>