

CHƯƠNG 2: HEAP SORT

GVGD: Ths NGUYỄN CHÍ THANH

Email: thanh.nc@ou.edu.com

NỘI DUNG

1. Định nghĩa Heap

2. Thuật toán

3. Thí dụ

4. Chương trình

5. Độ phức tạp

1. PHÁT BIỂU BÀI TOÁN



Cho một tập các số nguyên gồm n phần tử:


$$a_0, a_2, a_3, \dots, a_{n-1}$$


Yêu cầu: hãy thực hiện sắp xếp n phần tử này theo thứ tự tăng dần như sau:

$$a_0, a_2, a_3, \dots, a_{n-1}$$

Với $a_0 \leq a_2 \leq a_3 \leq \dots \leq a_{n-1}$

MÔ HÌNH HÓA BÀI TOÁN

 **Đầu vào:** một danh sách đặc (các số nguyên) gồm có n phần tử $a_0, a_2, a_3, \dots, a_{n-1}$.

 **Đầu ra:** một danh sách đặc (các số nguyên) gồm có n phần tử: $a_0, a_2, a_3, \dots, a_{n-1}$ ($a_0 \leq a_2 \leq a_3 \leq \dots \leq a_{n-1}$)

BIỂU DIỄN BÀI TOÁN TRÊN MÁY TÍNH

```
# define MAX 100
```

```
int a[MAX];
```

```
int n; // n là tổng số phần tử hiện có trong danh sách,  $0 \leq n < \text{MAX}$ 
```

ĐỊNH NGHĨA HEAP SORT



Một danh sách các phần tử là một Heap (Heap Max) khi và chỉ khi: Với mọi phần tử $a[i]$ bất kì trong danh sách ($i=0\dots n-1$), luôn có: $a[i] \geq a[2*i+1]$, và $a[i] \geq a[2*i+2]$



Mọi danh sách các phần tử là một Heap (Heap Min) khi và chỉ khi: Với mọi phần tử $a[i]$ bất kì trong danh sách ($i=0\dots n-1$), luôn có: $a[i] \leq a[2*i+1]$, và $a[i] \leq a[2*i+2]$.

HỆ QUẢ



Trong một Heap (Heap Max) phần tử đầu Heap là phần tử lớn nhất.

Ý TƯỞNG HEAPSORT

- **Bước 1:** Tạo Heap (Heap Max) ban đầu từ danh sách các phần tử cho trước. Thực hiện các bước tuần tự sau:
- **Bước 2:** Hoán vị phần tử đầu Heap ($a[0]$) với phần tử cuối *Heap* đang xét
- **Bước 3:** trong dãy đang xét, giới hạn phần tử cuối dãy (vừa thay thế giá trị phần tử đầu *Heap*). Kết quả là dãy đang xét giảm đi một phần tử bên phải.
- **Bước 4:** Lặp lại bước 2 trong khi dãy đang xét còn nhiều hơn 1 phần tử.

Ý TƯỞNG TẠO HEAP BAN ĐẦU

Chia dãy ban đầu $a[0], a[1], \dots, a[n-1]$, thành hai phần:

Nửa dãy bên trái: $a[0], a[1], \dots, a[(n/2)-1]$.

Nửa dãy bên phải: $a[n/2], \dots, a[n-1]$: các phần tử nửa dãy bên phải này thỏa tính chất các phần tử trong **Heap**, vì với mọi vị trí i trong nửa dãy này không tồn tại vị trí $2*i+1$.

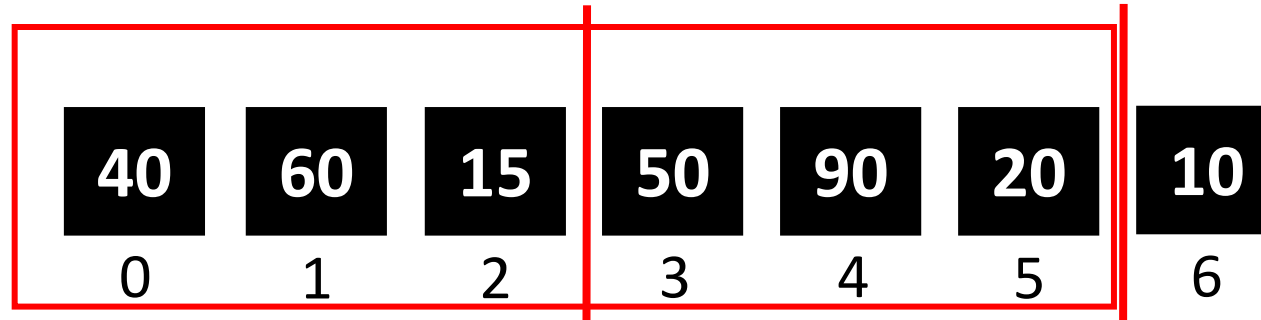
Ý TƯỞNG TẠO HEAP BAN ĐẦU (TT)

Để tạo Heap ban đầu, đầu tiên ta thực hiện tại vị trí $i = (n/2) - 1$

So sánh $a[i]$ với $a[2*i+1]$ và $a[2*i+2]$, nếu không thỏa tính chất Heap (Max) hoán vị $a[i]$ với phần tử $\max(a[2*i+1], a[2*i+2])$

Sau đó, giảm i một giá trị, và lặp lại việc so sánh $a[i]$ với $a[2*i+1]$ và $a[2*i+2]$, thực hiện hoán vị như trên nhằm đảm bảo phần tử tại vị trí i thỏa tính chất Heap

MINH HỌA



Việc tạo Heap (Max) hoàn tất. Ta được Heap như trên.

(thay thế giá trị 50) trước đó thỏa tính chất của heap so với $a[5]$ và $a[4]$

Bước 1: Tạo Heap ban đầu.

Bước 2: Hoán vị phần tử $a[0]$ và phần tử cuối Heap đang xét:

Bước 3: Trong dãy đang xét. Ta giới hạn phần tử cuối dãy (không xét nữa)

Bước 4: Khi $a[0], \dots, a[5]$ là một Heap, ta hoán vị $a[0]$ và $a[5]$. Tiếp tục xét danh sách các phần tử từ $a[0]$ đến $a[4]$. (tương tự cho đến khi danh sách đang xét chỉ còn 1 phần tử)

CÀI ĐẶT THUẬT GIẢI HEAP SORT

```
void HeapSort(int a[], int n)
{
    int i = n/2;
    while (i >= 0) // tạo heap ban đầu
    {
        shift(a, i, n-1);      i - -;
    }
    int right=n-1; // right là vị trí cuối Heap đang xét
    while (right>0)
    {
        swap(a[0], a[right]); // hoán vị phần tử a[0] cho phần tử cuối Heap đang xét
        right - -; // giới hạn lại phần tử cuối đang xét
        if (right > 0) // Kiểm tra dãy đang xét còn nhiều hơn 1 phần tử
            shift(a, 0, right); // tạo Heap lại tại vị trí 0
    }
}
```

```

void shift(int a[], int i, int n)
{
    int j = 2*i+1;
    if (j>=n) // nếu vị trí j không tồn tại trong danh sách đang xét thì thoát khỏi chương trình
        return;
    if (j+1 <n) // nếu tồn tại vị trí j+1 trong danh sách đang xét thì thoát khỏi chương trình
        if ( a[j]<a[j+1] ) // nếu vị trí j không tồn tại phần tử a[j] <a[j+1]
            j++;

    if (a[i] >= a[j] )
        return;
    else {
        int x = a[i];
        a[i] = a[j];
        a[j] = x;
        shift(a, j, n);
    }
}

```

ĐỘ PHỨC TẠP HEAP SORT

TRƯỜNG HỢP	ĐỘ PHỨC TẠP
Tốt nhất	$O(n \log n)$
Xấu nhất	$O(n \log n)$





TÀI LIỆU THAM KHẢO

- 1. Thomas H.Cormen, Charles E.Leiserson, Ronald L. Rivest, Clifford Stein**, (Chapter 2, 3) *Introduction to Algorithms*, Third Edition, 2009.
- 2. Adam Drozdek**, (Chapter 9) *Data Structures and Algorithms in C++*, Fourth Edition, CENGAGE Learning, 2013.
- 3. Lê Xuân Trường**, (Chapter 2) *Cấu trúc dữ liệu*, NXB Trường Đại học Mở TP-HCM, 2016.

BÀI TẬP CHƯƠNG 2

Bài 1: Thực hiện mô tả từng bước quá trình sắp xếp thứ tự dãy số nguyên bằng thuật toán Heap Sort

Bài 2: Quản lý danh sách đặc 100 phần tử kiểu số nguyên (int)

-  2.1. Khai báo cấu trúc danh sách.
-  2.2. Viết thủ tục nhập danh sách.
-  2.3. Viết thủ tục xuất danh sách
-  2.4. Viết thủ tục sắp xếp danh sách theo thứ tự tăng dần bằng thuật toán HeapSort

Bài tập thêm

Bài 3: Thực hiện đánh giá độ phức tạp của thuật toán HeapSort

Bài 4: Quản lý danh sách liên kết đơn (các phần tử kiểu số nguyên)

-  Viết thủ tục sắp xếp danh sách theo thứ tự tăng dần bằng thuật toán HeapSort