**BỘ GIÁO DỤC ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC THĂNG LONG**

**o0o**

**BÀI TẬP LỚN**

**CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

**SINH VIÊN THỰC HIỆN : Trần Phương Nam**

**MÃ SINH VIÊN : A25288**

**HÀ NỘI – 2017**

**MỤC LỤC**

[PHẦN 1. Giới thiệu headsort 1](#_Toc505290421)

[1.1. Giới thiệu 1](#_Toc505290422)

[1.2. Heapsort là gì? 1](#_Toc505290423)

[1.3. Cụ thể 1](#_Toc505290424)

[PHẦN 2. Thuật toán sắp xếp heapsort 2](#_Toc505290425)

[2.1. Đống 2](#_Toc505290426)

[2.2. Vun đống tại đỉnh thứ i 3](#_Toc505290427)

[2.3. Vun một mảng thành đống 3](#_Toc505290428)

[2.4. Săp xếp bằng vun đống 3](#_Toc505290429)

[PHẦN 3. Code giải mã heapsort 5](#_Toc505290430)

[3.1. Đặc tả 5](#_Toc505290431)

[3.2. Cài đặt 5](#_Toc505290432)

[3.2.1. Cài đặt chức năng 3.1.1 5](#_Toc505290433)

[3.2.2. Cài đặt chức năng 3.1.2 5](#_Toc505290434)

[3.2.3. Cài đặt chức năng 3.1.3 6](#_Toc505290435)

[3.2.4. Cài dặt chức năng 3.1.4 6](#_Toc505290436)

[3.2.5. Cài đặt chức năng 3.1.5 6](#_Toc505290437)

[3.2.6. Cài đặt chức năng 3.1.6 7](#_Toc505290438)

[PHẦN 4. Ứng dụng 8](#_Toc505290439)

# Giới thiệu headsort

## Giới thiệu

Giải thuật sắp xếp Heapsort được đề xuất vào năm 1964 bởi J. W. J. Williams. Cùng năm đó, R. W. Floyd đã đưa ra phiên bản cải tiến của thuật toán này, giúp nó trở thành thuật toán in-place với thời gian thực thi nhanh và độ phức tạp trong trường hợp xấu nhất là O (n log n).

A person wearing a suit and tie smiling at the camera

Description generated with very high confidence

Hình J. W. J. Williams

## Heapsort là gì?

**S**ắp xếp vun đống ([heapsort](http://it.die.vn/la-gi/heapsort/)**)** là một trong các phương pháp sắp xếp chọn. Ở mỗi bước của sắp xếp chọn ta chọn phần tử lớn nhất (hoặc nhỏ nhất) đặt vào cuối (hoặc đầu) danh sách, sau đó tiếp tục với phần còn lại của danh sách.

Thông thường sắp xếp chọn chạy trong thời gian *O(n*2*)*. Nhưng heapsort đã giảm độ phức tạp này bằng cách sử dụng một cấu trúc [dữ liệu](http://it.die.vn/la-gi/du-lieu/) đặc biệt được gọi là đống (*heap*).

Đống là cây nhị phân mà trọng số ở mỗi đỉnh cha lớn hơn hoặc bằng trọng số các đỉnh con của nó. Một khi danh sách dữ liệu đã được vun thành đống, gốc của nó là phần tử lớn nhất, thuật toán sẽ giải phóng nó khỏi đống để đặt vào cuối danh sách.

Sắp xếp vun đống chạy trong thời gian *O(n log n)*.

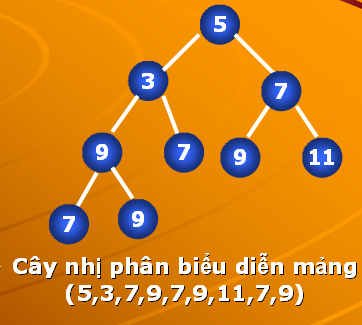
## Cụ thể

Sắp xếp vun đống (Heapsort) dựa trên một cấu trúc dữ liệu được gọi là đống nhị phân (binary heap), gọi đơn giản là đống. Trong mục này chỉ nói về đống trong bài toán sắp xếp.

# Thuật toán sắp xếp heapsort

## Đống

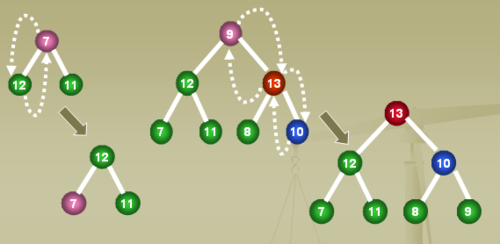
* Mỗi mảng a[1..n] có thể xem như một [cây](https://vi.wikipedia.org/wiki/C%C3%A2y_(l%C3%BD_thuy%E1%BA%BFt_%C4%91%E1%BB%93_th%E1%BB%8B)) nhị phân gần đầy (có trọng số là các giá trị của mảng), với gốc ở phần tử thứ nhất, con bên trái của đỉnh a[i] là a[2\*i] con bên phải là a[2\*i+1] (nếu mảng bắt đầu từ 1 còn nếu mảng bắt đầu từ 0 thì hai con là a[2\*i+1] và a[2\*i+2]) (nếu 2\*i<=n hoặc 2\*i+1<=n, khi đó các phần tử có chỉ số lớn hơn không có con, do đó là [lá](https://vi.wikipedia.org/w/index.php?title=L%C3%A1_(l%C3%BD_thuy%E1%BA%BFt_%C4%91%E1%BB%93_th%E1%BB%8B)&action=edit&redlink=1)).
* Một cây nhị phân, được gọi là đống cực đại nếu khóa của mọi nút không nhỏ hơn khóa các con của nó. Khi biểu diễn một mảng a[]bởi một cây nhi phân theo thứ tự tự nhiên điều đó nghĩa là a[i]>=a[2\*i] và a[i]>=a[2\*i+1] với mọi i =1..int(n/2). Ta cũng sẽ gọi mảng như vậy là đống. Như vậy trong đống a[1] (ứng với gốc của cây) là phần tử lớn nhất. Mảng bất kỳ chỉ có một phần tử luôn luôn là một đống.



Hình Ví dụ về mảng và cây nhị phân

* Một đống cực tiểu được định nghĩa theo các bất đẳng thức ngược lại: a[i]<=a[2\*i] và a[i]<=a[2\*i+1]. Phần tử đứng ở gốc cây cực tiểu là phần tử nhỏ nhất.

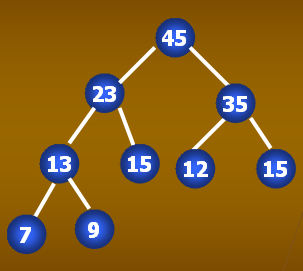
## Vun đống tại đỉnh thứ i



Hình Vun đống tại mỗi đỉnh của cây nhỏ

Nếu hai cây con gốc 2\*i và  2\*i+1 đã là đống thì để cây con gốc i trở thành đống chỉ việc so sánh giá trị  a[i] với giá trị lớn hơn trong hai giá trị  a[2\*i] và  a[2\*i+1], nếu  a[i] nhỏ hơn thì đổi chỗ chúng cho nhau. Nếu đổi chỗ cho  a[2\*i], tiếp tục so sánh với con lớn hơn trong hai con của nó cho đên khi hoặc gặp đỉnh lá. (Thủ tục DownHeap trong giả mã dưới đây)

## Vun một mảng thành đống



Hình Mảng là một đống

Để vun mảng a[1..n] thành đống ta vun từ dưới lên, bắt đầu từ phần tử a[j]với j =Int(n/2) ngược lên tới a[1]. (Thủ tục MakeHeap trong giả mã dưới đây).

## Săp xếp bằng vun đống

* Đổi chỗ (Swap): Sau khi mảng a[1..n] đã là đống, lấy phần tử a[1] trên đỉnh của đống ra khỏi đống đặt vào vị trí cuối cùng n, và chuyển phần tử thứ cuối cùng a[n] lên đỉnh đống thì phần tử a[n] đã được đứng đúng vị trí.
* Vun lại: Phần còn lại của mảng a[1..n-1] chỉ khác cấu trúc đống ở phần tử a[1]. Vun lại mảng này thành đống với n-1 phần tử.
* Lặp: Tiếp tục với mảng a[1..n-1]. Quá trình dừng lại khi đống chỉ còn lại một phần tử.

# Code giải mã heapsort

## Đặc tả

Heapsort là một trong các phương pháp sắp xếp chọn. Ở mỗi bước của sắp xếp chọn ta chọn phần tử lớn nhất đặt vào đầu danh sách và nhỏ vào cuối danh sách, sau đó tiếp tục với phần còn lại của danh sách.

Các phép toán trên danh sách:

* Hàm Hoán vị vị trí (3.1.1)
* Hàm xác định nút cha (nút có giá trị lớn nhất) (3.1.2)
* Hàm xây dựng Heap (đống) (3.1.3)
* Lấy phần tử lớn nhất (3.1.4)
* Hàm nhập vào dãy chưa sắp xếp (3.1.5)
* Hàm Xuất dãy sau khi đã sắp xếp (3.1.6)

## Cài đặt mảng

### Cài đặt chức năng 3.1.1

Hàm hoán vị vị trí

Void swap (int &a, int &b)

{

Int temp = a;

A = b;

B = temp;

}

### Cài đặt chức năng 3.1.2

Hàm tìm nút cha

Void MaxHeapify (int a [], int n, int i)

{

Int left = 2\*(i+1) – 1;

Int right = 2\*(i+1);

Int max;

If (left < n && a [left] > a [i])

Max = left;

Else

Max = i;

If (right < n && a [right] > a [max])

Max = right;

If (i != max)

{

Swap (a[i], a[max]);

MaxHeapify (a, n, max);

}

}

### Cài đặt chức năng 3.1.3

Xây dựng cây nhị phân duyệt từ gốc về đỉnh

Void BuildHeap (int a [], int n)

{

For (int i=n/2 -1; i>=0; i--)

{

MaxHeapify (a, n, i);

}

}

### Cài dặt chức năng 3.1.4

Lấy phần tử lớn nhất của cây ra

Void HeapSort (int a [], int n)

{

BuildHeap (a, n);

For (int I = n-1; i>0; i--)

{

Swap (a [0], a[i]);

MaxHeapify (a, i, 0);

}

}

### Cài đặt chức năng 3.1.5

Nhập dãy ban đầu

Void Nhap (int a [], int n)

{

For (int i=0; i<n; i++)

{

Cout<<” Nhap phan tu thu a[“<<i+1<<”] = ”;

Cin>>a[i];

}

Cout<<endl;

}

### Cài đặt chức năng 3.1.6

Xuất dãy đã sắp xếp

Void Xuat (int a [], int n)

{

For (int i=0; i<n; i++)

{

Cout<<” “<<a[i];

}

Cout<<endl;

}

# Ứng dụng

Ngoài [giải thuật sắp xếp vun đống](http://it.die.vn/la-gi/giai-thuat-sap-xep-vun-dong/), cấu trúc đống còn được ứng dụng trong nhiều giải thuật khác, khi cần lấy ra nhanh chóng các phần tử lớn nhất (hoặc nhỏ nhất) của một dãy phần tử, chẳng hạn trong hàng đợi có ưu tiên trong đó tiêu chuẩn ưu tiên là có khóa lớn nhất (hoặc nhỏ nhất). Có thể tìm thấy điều đó trong giải thuật tìm bộ mã Huffman cho một bảng tần số của các kí tự.