



# HUST

**ĐẠI HỌC BÁCH KHOA HÀ NỘI**  
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.





ĐẠI HỌC  
BÁCH KHOA HÀ NỘI  
HANOI UNIVERSITY  
OF SCIENCE AND TECHNOLOGY

# LẬP TRÌNH C CƠ BẢN

SẮP XẾP VÀ ỨNG DỤNG (PHẦN 1)

ONE LOVE. ONE FUTURE.

- Bài toán sắp xếp dãy các số nguyên (P.05.11.01)
- Bài toán sắp xếp dãy các chuỗi ký tự (P.05.11.02)
- Bài toán xếp hạng năng lực học tập (P.05.11.03)

# BÀI TẬP SẮP XẾP CÁC SỐ NGUYÊN (P.05.11.01)

- Cho dãy số nguyên  $a_1, a_2, \dots, a_n$ . Hãy sắp xếp dãy đã cho theo thứ tự không giảm
- Dữ liệu
  - Dòng 1: ghi số nguyên dương  $n$  ( $1 \leq n \leq 100000$ )
  - Dòng 2 ghi  $a_1, a_2, \dots, a_n$  trong đó ( $1 \leq a_i \leq 1000000$ )
- Kết quả
  - Ghi ra dãy đã được sắp xếp

stdin	stdout
5 4 3 6 5 1	1 3 4 5 6

# BÀI TẬP SẮP XẾP CÁC SỐ NGUYÊN – MÃ GIẢ

- Áp dụng sắp xếp vun đống

```
Heapify(a, i, n){
    L = 2*i; R = 2*i+1; maxIdx = i;
    if L <= n and a[L] > a[maxIdx] then maxIdx = L;
    if R <= n and a[R] > a[maxIdx] then maxIdx = R;
    if maxIdx != i then {
        swap(a[i], a[maxIdx]); Heapify(maxIdx, n);
    }
}

BuildHeap(){
    for i = n/2 downto 1 do Heapify(i, n);
}

HeapSort(){
    BuildHeap();
    for i = n downto 2 do {
        swap(a[1], a[i]); Heapify(1, i-1);
    }
}
```

# BÀI TẬP SẮP XẾP CÁC SỐ NGUYÊN – CODE

```
#include <stdio.h>
#define N 1000001
int a[N];
int n;
void swap(int i, int j){
    int t = a[i]; a[i] = a[j]; a[j] = t;
}
void heapify(int i, int n){
    int L = 2*i; int R = 2*i+1; int maxIdx = i;
    if(L <= n && a[maxIdx] < a[L]) maxIdx = L;
    if(R <= n && a[maxIdx] < a[R]) maxIdx = R;
    if(maxIdx != i){
        swap(i, maxIdx); heapify(maxIdx, n);
    }
}
```

```
void buildHeap(){
    for(int i = n/2; i >= 1; i--) heapify(i,n);
}
void heapSort(){
    buildHeap();
    for(int i = n; i >= 2; i--){ swap(1,i); heapify(1,i-1); }
}
int main(){
    scanf("%d",&n);
    for(int i = 1; i <= n; i++) scanf("%d",&a[i]);
    heapSort();
    for(int i = 1; i <= n; i++) printf("%d ",a[i]);
    return 0;
}
```

# BÀI TẬP SẮP XẾP CÁC XÂU KÝ TỰ (P.05.11.02)

- Cho dãy các chuỗi ký tự  $S_1, S_2, \dots, S_n$ . Hãy sắp xếp dãy đã cho theo thứ tự từ điển
- Dữ liệu
  - Dòng 1: ghi số nguyên dương  $n$  ( $1 \leq n \leq 100000$ )
  - Dòng 2 ghi  $S_1, S_2, \dots, S_n$  trong đó độ dài các chuỗi từ 1 đến 100
- Kết quả
  - Ghi ra trên mỗi dòng 1 chuỗi trong dãy đã được sắp xếp

stdin	stdout
10	I010
O0001	N09
Z002	O0001
R003	P00006
R00004	P05
P05	R00004
P00006	R003
T0007	T0007
X08	X08
N09	Z002
I010	



# BÀI TẬP SẮP XẾP CÁC XÂU KÝ TỰ - THUẬT TOÁN – MÃ GIẢ

- Sử dụng mảng các con trỏ, mỗi con trỏ trỏ đến 1 mảng các ký tự (cấp phát động)
- Khi đổi chỗ 2 phần tử thì chỉ đổi chỗ 2 con trỏ (không dùng hàm copy xâu)
- Thuật toán sắp xếp vun đống được áp dụng

```
Heapify(a, i, n){
    L = 2*i; R = 2*i+1; maxIdx = i;
    if L <= n and a[L] > a[maxIdx] then maxIdx = L;
    if R <= n and a[R] > a[maxIdx] then maxIdx = R;
    if maxIdx != i then {
        swap(a[i], a[maxIdx]); heapify(maxIdx, n);
    }
}

BuildHeap(){
    for i = n/2 downto 1 do Heapify(i, n);
}

HeapSort(){
    BuildHeap();
    for i = n downto 2 do {
        swap(a[1], a[i]); Heapify(1, i-1);
    }
}
```

# BÀI TẬP SẮP XẾP CÁC XÂU KÝ TỰ - CODE

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define N 100001
#define MAX_LEN 100
char* s[N];
int n;
void input(){
    scanf("%d",&n);
    char str[MAX_LEN];
    for(int i=1; i <= n; i++){
        scanf("%s",str);
        s[i] = (char*)malloc(strlen(str)+1);
        strcpy(s[i],str);
    }
}
```

```
void swap(int i, int j){
    char* t = s[i]; s[i] = s[j]; s[j] = t;
}
void heapify(int i, int n){
    int L = 2*i; int R = 2*i+1;
    int maxIdx = i;
    if(L <= n && strcmp(s[maxIdx],s[L]) < 0) maxIdx = L;
    if(R <= n && strcmp(s[maxIdx],s[R]) < 0) maxIdx = R;
    if(maxIdx != i){
        swap(i,maxIdx);
        heapify(maxIdx,n);
    }
}
```

# BÀI TẬP SẮP XẾP CÁC XÂU KÝ TỰ - CODE

```
void buildHeap(){
    for(int i = n/2; i >= 1; i--){
        heapify(i,n);
    }
}
void heapSort(){
    buildHeap();
    for(int i = n; i >= 2; i--){
        swap(1,i); heapify(1,i-1);
    }
}
```

```
int main(){
    input();
    heapSort();
    for(int i =1; i <= n; i++){
        printf("%s\n",s[i]);
    }
    return 0;
}
```

# BÀI TẬP XẾP HẠNG NĂNG LỰC HỌC TẬP (P.05.11.03)

- Có một danh sách các sinh viên cần được xếp hạng theo điểm số, mỗi sinh viên có 2 trường thông tin:
  - <studentID>: xâu ký tự độ dài từ 1 đến 10 là mã số sinh viên
  - <grade>: điểm số (số nguyên dương)
- Biết rằng điểm số của các sinh viên đôi một khác nhau. Hãy tính toán vị trí của mỗi sinh viên trong bảng xếp hạng (số lượng sinh viên trong danh sách có điểm số nhỏ hơn)
- Dữ liệu
  - Dòng 1: ghi số nguyên dương  $n$  ( $1 \leq n \leq 100000$ )
  - Dòng  $i+1$  ( $i = 1, 2, \dots, n$ ): ghi <studentID> và <grade> của sinh viên thứ  $i$
- Kết quả
  - Ghi ra trên mỗi dòng <studentID> và vị trí trong bảng xếp hạng (các dòng được sắp xếp theo thứ tự từ điển của mã số sinh viên)

stdin	stdout
5	S000001 3
S000003 3	S000002 2
S000002 6	S000003 0
S000005 5	S000004 4
S000004 10	S000005 1
S000001 8	

# BÀI TẬP XẾP HẠNG NĂNG LỰC HỌC TẬP – THUẬT TOÁN – MÃ GIẢ

- Sắp xếp danh sách sinh viên theo thứ tự tăng dần của điểm số bằng thuật toán sắp xếp vun đống được danh sách  $S_1, S_2, \dots, S_n$ .
- Khi đó vị trí của sinh viên  $S_i$  chính là  $i$  ( $i = 1, 2, \dots, n$ )
- Sắp xếp danh sách sinh viên theo thứ tự từ điển của mã số sinh viên và in kết quả.

```
Struct Student {  
    ID; // mã số sinh viên  
    grade; // điểm  
    pos; // vị trí  
}
```

```
Heapify(S, i, n){  
    L = 2*i; R = 2*i+1; maxIdx = i;  
    if L <= n and S[L].grade > S[maxIdx].grade then maxIdx = L;  
    if R <= n and S[R].grade > S[maxIdx].grade then maxIdx = R;  
    if maxIdx != i then {  
        swap(S[i], S[maxIdx]); Heapify(maxIdx, n);  
    }  
}  
  
BuildHeap(){  
    for i = n/2 downto 1 do Heapify(i, n);  
}  
  
HeapSort(){  
    BuildHeap();  
    for i = n downto 2 do {  
        swap(a[1], a[i]); Heapify(1, i-1);  
    }  
}
```

# BÀI TẬP XẾP HẠNG NĂNG LỰC HỌC TẬP – THUẬT TOÁN – MÃ GIẢ

- Sắp xếp danh sách sinh viên theo thứ tự tăng dần của điểm số bằng thuật toán sắp xếp vun đống được danh sách  $S_1, S_2, \dots, S_n$ .
- Khi đó vị trí của sinh viên  $S_i$  chính là  $i-1$  ( $i = 1, 2, \dots, n$ )
- Sắp xếp danh sách sinh viên theo thứ tự từ điển của mã số sinh viên và in kết quả.

```
Struct Student {  
    ID; // mã số sinh viên  
    grade; // điểm  
    pos; // vị trí  
}
```

```
Main() {  
    Đọc danh sách sinh viên S[1], S[2], . . ., S[n];  
    HeapSort();  
    for i = 1 to n do S[i].pos = i-1;  
    Sắp xếp lại S[1], . . ., S[n] theo mã số sinh viên;  
    for i = 1 to n do {  
        print(S[i].ID, ' ', S[i].pos);  
    }  
}
```

# BÀI TẬP XẾP HẠNG NĂNG LỰC HỌC TẬP – CODE

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define N 100001
typedef struct Student{
    char ID[10];
    int grade;
    int pos;
}Student;

Student S[N];
int n;
```

```
void input(){
    scanf("%d",&n);
    for(int i=1; i <= n; i++){
        scanf("%s %d",S[i].ID,&S[i].grade);
    }
}

void swap(int i, int j){
    Student t = S[i]; S[i] = S[j]; S[j] = t;
}

void heapify(int i, int n){
    int L = 2*i; int R = 2*i+1;
    int maxIdx = i;
    if(L <= n && S[maxIdx].grade < S[L].grade) maxIdx = L;
    if(R <= n && S[maxIdx].grade < S[R].grade) maxIdx = R;
    if(maxIdx != i){ swap(i,maxIdx); heapify(maxIdx,n); }
}
```

# BÀI TẬP XẾP HẠNG NĂNG LỰC HỌC TẬP – CODE

```
void buildHeap(){
    for(int i = n/2; i >= 1; i--){
        heapify(i,n);
    }
}
void heapSort(){
    buildHeap();
    for(int i = n; i >= 2; i--){
        swap(1,i); heapify(1,i-1);
    }
}
```

```
void heapify2(int i, int n){
    int L = 2*i; int R = 2*i+1;
    int maxIdx = i;
    if(L <= n && strcmp(S[maxIdx].ID,S[L].ID) < 0) maxIdx = L;
    if(R <= n && strcmp(S[maxIdx].ID,S[R].ID) < 0) maxIdx = R;
    if(maxIdx != i){ swap(i,maxIdx); heapify2(maxIdx,n); }
}
void buildHeap2(){
    for(int i = n/2; i >= 1; i--){
        heapify2(i,n);
    }
}
```



# BÀI TẬP XẾP HẠNG NĂNG LỰC HỌC TẬP – CODE

```
void heapSort2(){
    buildHeap2();
    for(int i = n; i >= 2; i--){
        swap(1,i); heapify2(1,i-1);
    }
}

void print(){
    for(int i =1; i <= n; i++){
        printf("%s %d\n",S[i].ID,S[i].pos);
    }
}
```

```
int main(){
    input();
    heapSort();
    for(int i= 1; i <= n; i++){
        S[i].pos = i-1;
    }
    heapSort2();
    print();
    return 0;
}
```

- Visualization:
- <https://www.sortvisualizer.com/mergesort/>
- <https://www.toptal.com/developers/sorting-algorithms>
- <https://visualgo.net/en/sorting>

A large graphic on the left side of the slide. It features a dark blue background with a circular pattern of red dots of varying sizes, creating a sense of depth and movement. The word "HUST" is centered within this graphic in a white, bold, sans-serif font.

# HUST

# THANK YOU !