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 2)

ONE LOVE. ONE FUTURE.

NỘI DUNG

- Bài toán sắp xếp dãy các vector số nguyên (P.05.12.01)
- Bài toán sắp xếp dãy các bản ghi (P.05.12.02)
- Bài toán tìm tập con chung lớn nhất của 2 tập hợp (P.05.12.03)

BÀI TẬP SẮP XẾP CÁC VECTOR SỐ NGUYÊN (P.05.12.01)

- Cho dãy các vector (mỗi vector gồm m số nguyên) $a_1, a_2, \ldots, a_n$. Hãy sắp xếp dãy các vector đã cho theo thứ tự không giảm
- Dữ liệu
  - Dòng 1: ghi 2 số nguyên dương n và m ($1 <= n <= 100000$, $1 <= m <= 10$)
  - Dòng i+1 (i = 1, 2, …, n): ghi m phần tử của vector a_i (các phần tử nhận giá trị từ 1 đến 100)
- Kết quả
  - Dòng i (i = 1, 2, …, n): ghi m phần tử của vector a_i trên dãy đã được sắp xếp (sau mỗi phần tử có 1 ký tự SPACE)

stdin	stdout
6 3 10 9 7 5 10 2 10 9 1 4 7 3 7 5 10 7 9 3	4 7 3 5 10 2 7 5 10 7 9 3 10 9 1 10 9 7

BÀI TẬP SẮP XẾP CÁC SỐ NGUYÊN – MÃ GIẢ

- Áp dụng sắp xếp vun đống
- Sử dụng con trỏ và cấp phát động:
  - a[i] là con trỏ đến vector thứ i

```
cmp(a, i, j){// so sánh vector a[i] và a[j]
   for k = 1 to m do {
     if a[i][k] < a[j][k] then return -1;
     else if a[i][k] > a[j][k] then return 1;
   }
   return 0; //  hai vector bằng nhau
}
```

```
Heapify(a, i, n){
  L = 2*i; R = 2*i+1; maxIdx = i;
  if L <= n and cmp(a,L,maxIdx) = 1 then maxIdx = L;
  if R <= n and cmp(a,R,maxIdx) = 1 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 100001
int* a[N];
int n,m;
void input(){
  scanf("%d%d",&n,&m);
  for(int i = 1; i <= n; i++){
    a[i] = (int*)malloc(sizeof(int)*(m+1));
    for(int j = 1; j <= m; j++) scanf("%d",&a[i][j]);
  }
}
```

```
int cmp(int i, int j){
  for(int k = 1; k <= m; k++){
    if(a[i][k] < a[j][k]) return -1;
    else if(a[i][k] > a[j][k]) return 1;
  }
  return 0;
}
void swap(int i, int j){
  int* tmp = a[i]; a[i] = a[j]; a[j] = tmp;
}
```

BÀI TẬP SẮP XẾP CÁC SỐ NGUYÊN – CODE

```
void heapify(int i, int n){
  int L = 2*i; int R = 2*i+1; int maxIdx = i;
  if(L <= n && cmp(maxIdx,L) < 0) maxIdx = L;
  if(R <= n && cmp(maxIdx,R) < 0) 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); }
}
```

```
void print(){
  for(int i = 1; i <= n; i++){
    for(int j = 1; j <= m; j++) printf("%d ",a[i][j]);
    printf("\n");
  }
}
int main(){
  input();
  heapSort();
  print();
  return 0;
}
```

BÀI TẬP SẮP XẾP DÃY CÁC BẢN GHI (P.05.12.02)

- Mỗi thí sinh trong một cuộc thi có 2 thông tin chính:
  - code: là mã thí sinh (xâu ký tự độ dài từ 2 đến 10)
  - score: điểm số (số nguyên từ 0 đến 1000000, điểm số các thí sinh đôi một khác nhau)
- Hãy viết chương trình sắp xếp các thí sinh theo thứ tự giảm dần của điểm số
- Dữ liệu
  - Mỗi dòng chứa 2 thông tin và code và score của một thí sinh
  - Dữ liệu đầu vào kết thúc bởi dòng chứa #
- Kết quả
  - Ghi ra trên mỗi dòng code và score của 1 thí sinh trong danh sách đã sắp xếp (sau mỗi thông tin có 1 ký tự SPACE)

stdin	stdout
S00001 27412 S00002 22981 S00003 32561 S00004 10915 S00005 17566 #	S00003 32561 S00001 27412 S00002 22981 S00005 17566 S00004 10915

BÀI TẬP SẮP XẾP DÃY CÁC BẢN GHI - THUẬT TOÁN – MÃ GIẢ

- Định nghĩa kiểu cấu trúc để lưu dữ liệu

```
struct Candidate{
   code; // mã
   score; // điểm số
}
```

- Sử dụng mảng các con trỏ, mỗi con trỏ trỏ đến 1 bản ghi (cấp phát động)
- Khi đổi chỗ 2 phần tử thì chỉ đổi chỗ 2 con trỏ (không cần hoán đổi nội dung vùng nhớ của 2 bản ghi)
- Thuật toán sắp xếp vun đống được áp dụng

```
Heapify(a, i, n){
  L = 2*i; R = 2*i+1; minIdx = i;
  if L <= n and a[L].score < a[minIdx].score then minIdx = L;
  if R <= n and a[R].score < a[minIdx].score then minIdx = R;
  if minIdx != i then {
     swap(a[i], a[minIdx]); heapify(minIdx, 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 DÃY CÁC BẢN GHI - CODE

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define N 100001
typedef struct Candidate{
  int score;
  char code[10];
}Candidate;
int n;
Candidate* a[N];
```

```
void input(){
  char code[11];
  n = 0;
  while(1){
    scanf("%s",code);
    if(strcmp(code,"#")==0) break;
    int score; scanf("%d",&score);
    n++;
    a[n] = (Candidate*)malloc(sizeof(Candidate));
    a[n]->score = score;
    strcpy(a[n]->code,code);
  }
}
void swap(int i, int j){
  Candidate* t = a[i]; a[i] = a[j]; a[j] = t;
}
```

BÀI TẬP SẮP XẾP DÃY CÁC BẢN GHI - CODE

```
void heapify(int i, int n){
  int L = 2*i; int R = 2*i+1; int minIdx = i;
  if(L <= n && a[L]->score < a[minIdx]->score) minIdx = L;
  if(R <= n && a[R]->score < a[minIdx]->score) minIdx = R;
  if(minIdx != i){
    swap(i,minIdx); heapify(minIdx,n);
  }
}
void buildHeap(){
  for(int i = n/2; i >= 1; i--) heapify(i,n);
}
void sort(){
  buildHeap();
  for(int i = n; i >= 2; i--){ swap(1,i); heapify(1,i-1);  }
}
```

```
void print(){
  for(int i = 1; i <= n; i++){
    printf("%s %d\n",a[i]->code,a[i]->score);
  }
}
int main(){
  input();
  sort();
  print();
  return 0;
}
```

BÀI TẬP TẬP CON CHUNG LỚN NHẤT (P.05.12.03)

- Cho 2 tập số nguyên $A = \{a_1, a_2, \ldots, a_n\}$ và $B = \{b_1, b_2, \ldots, b_m\}$. Hãy tìm tập con của A và B có nhiều phần tử nhất
- Dữ liệu
  - Dòng 1: ghi 2 số nguyên dương n và m ($1 <= n, m <= 100000$)
  - Dòng 2: ghi n số nguyên dương $a_1, a_2, \ldots, a_n$ ($1 <= a_i <= 1000000$)
  - Dòng 3: ghi m số nguyên dương $b_1, b_2, \ldots, b_m$ ($1 <= b_i <= 1000000$)
- Kết quả
  - Ghi ra số lượng phần tử của tập con chung tìm được

stdin	stdout
6 6 7 3 10 1 2 8 6 2 8 10 5 7	4

BÀI TẬP TẬP CON CHUNG LỚN NHẤT – THUẬT TOÁN – MÃ GIẢ

- Sắp xếp 2 dãy $a_1, a_2, \ldots, a_n$ và $b_1, b_2, \ldots, b_m$ theo thứ tự tăng dần
- Cho 2 biến chỉ số i và j chạy từ trái qua phải trên dãy $a_1, a_2, \ldots, a_n$ và dãy $b_1, b_2, \ldots, b_m$.
  - Nếu $a_i = b_j$ thì tăng biến đếm lên 1, đồng thời tăng i và j lên 1
  - Nếu $a_i < b_j$ thì tăng i lên 1
  - Nếu $a_i > b_j$ thì tăng j lên 1

```
maxCommonSubset(a, b, n, m){
  sắp xếp a và b theo thứ tự tăng dần;
  cnt = 0;
  i = 1; j = 1;
  while i <= n and j <= m do {
    if a[i] = b[j] then cnt = cnt + 1;
    else if a[i] > b[j] then j = j + 1;
    else i = i + 1;
  }
  return cnt;
}
```

BÀI TẬP TẬP CON CHUNG LỚN NHẤT– CODE

```
#include <stdio.h>
#define N 100001
int a[N],b[N];
int n,m;
void input(){
  scanf("%d%d",&n,&m);
  for(int i = 1; i<= n; i++) scanf("%d",&a[i]);
  for(int j = 1; j <= m; j++) scanf("%d",&b[j]);
}
void swap(int* a, int i, int j){
  int t = a[i]; a[i] = a[j]; a[j] = t;
}
```

```
void heapify(int* a, int i, int n){
  int L = 2*i; int R = 2*i+1; int maxIdx = i;
  if(L <= n && a[L] > a[maxIdx]) maxIdx = L;
  if(R <= n && a[R] > a[maxIdx]) maxIdx = R;
  if(maxIdx != i){
    swap(a, i, maxIdx); heapify(a, maxIdx, n);
  }
}
void buildHeap(int* a, int n){
  for(int i = n/2; i >= 1; i--) heapify(a,i,n);
}
void sort(int* a, int n){
  buildHeap(a,n);
  for(int i = n; i >= 2; i--){ swap(a,1,i); heapify(a,1,i-1); }
}
```

BÀI TẬP TẬP CON CHUNG LỚN NHẤT– CODE

```
void print(int* a, int n){
  for(int i =1 ; i <= n; i++) printf("%d ",a[i]);
  printf("\n");
}
int main(){
  input();
  sort(a,n); sort(b,m);
  int cnt = 0;  int i = 1; int j = 1;
  while(i <= n && j <= m){
    if(a[i] == b[j]){  cnt += 1; i++; j++;  }
    else if(a[i] > b[j]) j++;
    else i++;
  }
  printf("%d",cnt);
  return 0;
}
```

Tham khảo

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

HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !