THUẬT TOÁN ỨNG DỤNG

CHIA ĐỂ TRỊ

Phạm Quang Dũng
Bộ môn KHMT
dungpq@soict.hust.edu.vn

Nội dung

- Tổng quan chia để trị
- Ví dụ minh họa
- Độ phức tạp chia để trị
- Giảm để trị

Tổng quan chia để trị

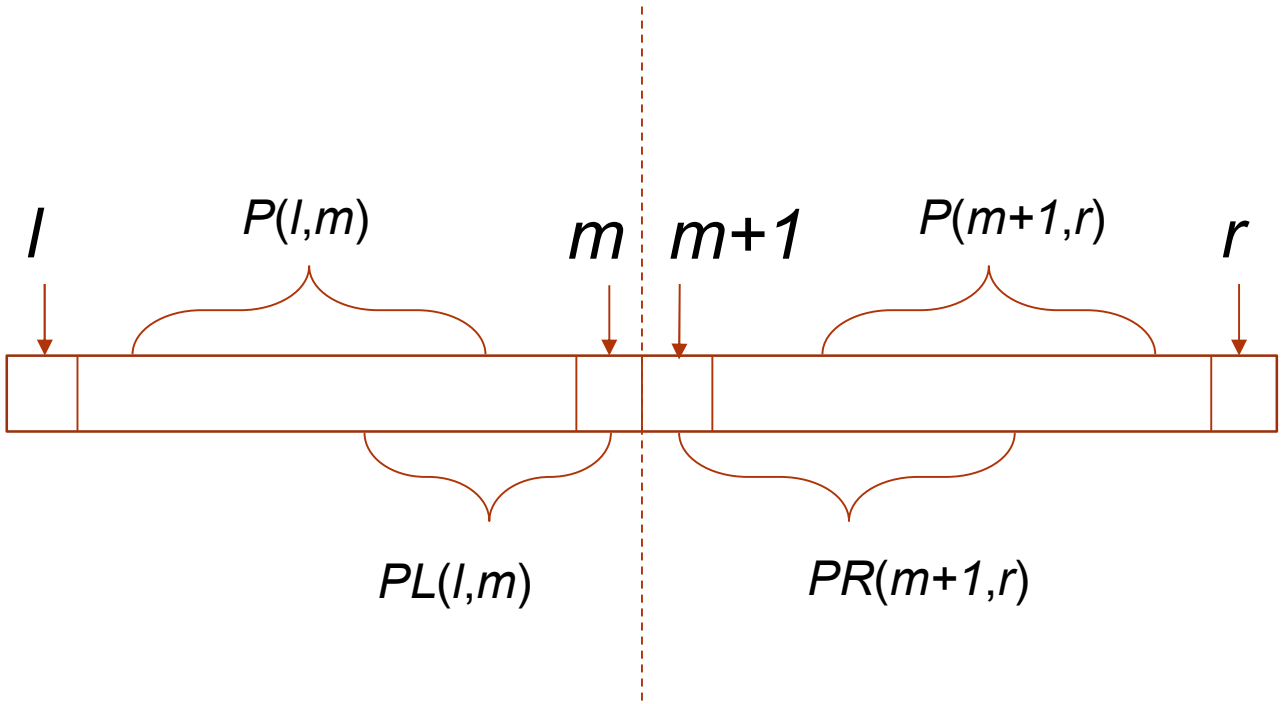
- Chia bài toán cần giải ban đầu thành các bài toán con độc lập nhau
- Giải (trị) các bài toán con
- Tổng hợp lời giải của các bài toán con để dẫn ra lời giải của bài toán xuất phát

Ví dụ minh họa

- Bài toán dãy con dài nhất: cho dãy số nguyên $a = a_1, a_2, \ldots, a_n$. Tìm dãy con gồm một số liên tiếp các phần tử có tổng lớn nhất
- Phân chia: ký hiệu $P(i, j)$ là lời giải của bài toán tìm dãy con liên tiếp của dãy $a_i, a_{i+1}, \ldots, a_j$ có tổng cực đại
- Tổng hợp lời giải
  - Ký hiệu $PL(i, j)$ là lời giải của bài toán tìm dãy con liên tiếp của dãy $a_i, a_{i+1}, \ldots, a_j$ sao cho phần tử cuối cùng là a_j có tổng cực đại
  - Ký hiệu $PR(i, j)$ là lời giải của bài toán tìm dãy con liên tiếp của dãy $a_i, a_{i+1}, \ldots, a_j$ sao cho phần tử đầu tiên là a_i có tổng cực đại

Ví dụ minh họa

- Xét đoạn [*l*,*l*+1,...,*r*]. Ký hiệu *m* = (*l*+*r*)/2
- *P*(*l*,*r*) = MAX{*P*(*l*, *m*), *P*(*m*+1,*r*), *PL*(*l*,*m*) + *PR*(*m*+1,*r*)}



Ví dụ minh họa

```
#include <bits/stdc++.h>
using namespace std;
#define INF 1e9
#define MAX 1000000

int a[MAX];
int n;
void input(){
    cin >> n;
    for(int i = 0; i < n; i++) cin >> a[i];
}
```

Ví dụ minh họa

```
int PL(int l, int r){
    int rs = -INF;
    int s = 0;
    for(int i = r; i >= l; i--){
        s += a[i];
        rs = max(rs,s);
    }
    return rs;
}

int PR(int l, int r){
    int rs = -INF;
    int s = 0;
    for(int i = l; i <= r; i++){
        s += a[i];
        rs = max(rs,s);
    }
    return rs;
}
```

Ví dụ minh họa

```
int P(int l, int r){
    if(l == r) return a[r];
    int m = (l+r)/2;
    return max(max(P(l,m),P(m+1,r)), PL(l,m)+PR(m+1,r));
}
void solve(){
    cout << P(0,n-1);
}
int main(){
    input();
    solve();
}
```

Độ phức tạp tính toán

- Chia bài toán xuất phát thành *a* bài toán con, mỗi bài toán con kích thước *n*/*b*
- *T*(*n*): thời gian của bài toán kích thước *n*
- Thời gian phân chia (dòng 4): *D*(*n*)
- Thời gian tổng hợp lời giải (dòng 6): *C*(*n*)
- Công thức truy hồi:

$$T(n) = \begin{cases} \Theta(1), & n \leq n0 \\ aT\left(\frac{n}{b}\right) + C(n) + D(n), & n > n0 \end{cases}$$

```
procedure D-and-C(n) {
1.   if(n ≤ n0)
2.     xử lý trực tiếp
3.   else{
4.     chia bài toán xuất phát
thành a bài toán con kích thước
n/b
5.     gọi đệ quy a bài toán con
6.     tổng hợp lời giải
7.   }
}
```

Độ phức tạp tính toán

- Độ phức tạp của thuật toán chia để trị (định lí thợ)
- Công thức truy hồi:

  $T(n) = aT(n/b) + cn^k$, với các hằng số $a \geq 1$, $b > 1$, $c > 0$
- Nếu $a > b^k$ thì $T(n) = \Theta(n^{log_b a})$
- Nếu $a = b^k$ thì $T(n) = \Theta(n^k logn)$ với $logn = log_2 n$
- Nếu $a < b^k$ thì $T(n) = \Theta(n^k)$

Độ phức tạp tính toán

- Độ phức tạp của thuật toán chia để trị (định lí thợ)
- Công thức truy hồi:

  $T(n) = aT(n/b) + cn^k$, với các hằng số $a \geq 1$, $b > 1$, $c > 0$
- Nếu $a > b^k$ thì $T(n) = \Theta(n^{log_b a})$
- Nếu $a = b^k$ thì $T(n) = \Theta(n^k logn)$ với $logn = log_2 n$
- Nếu $a < b^k$ thì $T(n) = \Theta(n^k)$

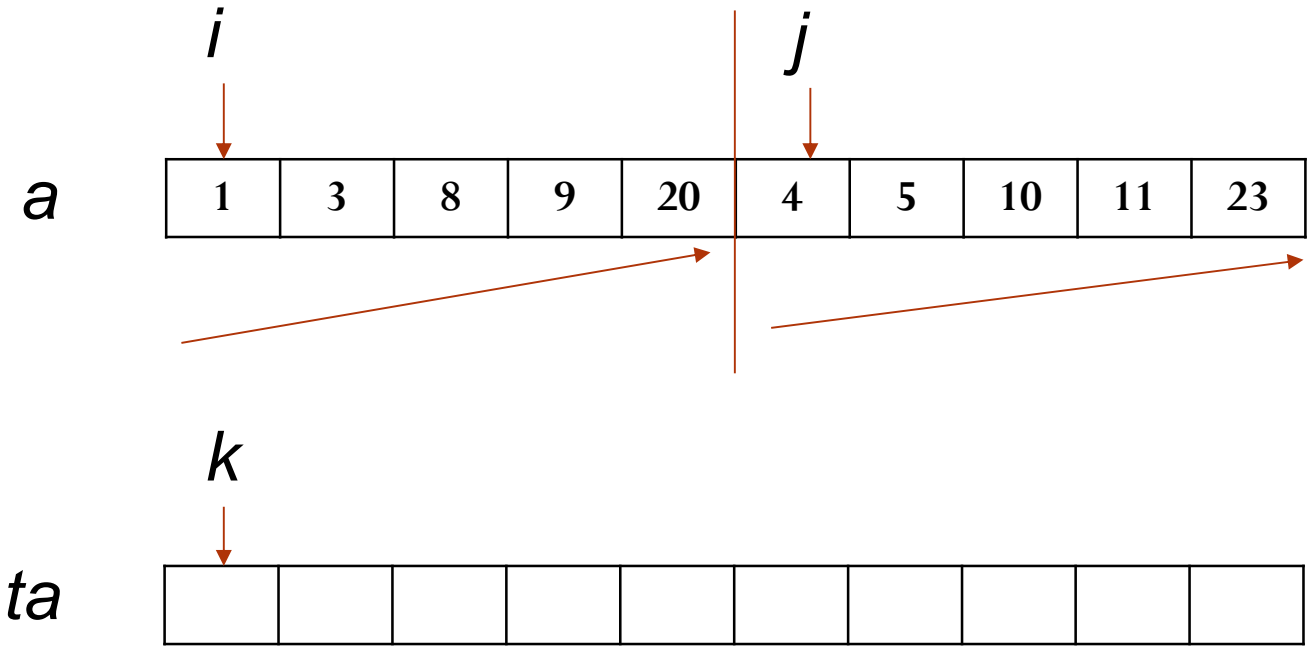
→ Thuật toán chia để trị giải bài toán tổng con cực đại có độ phức tạp là O(*nlogn*)

Sắp xếp trộn

- Phân chia: Chia dãy $a_1, \ldots, a_n$ thành 2 dãy con có độ dài bằng nhau
- Trị đệ quy: Sắp xếp 2 dãy con bằng thuật toán sắp xếp trộn
- Tổng hợp: Trộn 2 dãy con đã được sắp với nhau để thu được dãy ban đầu được sắp thứ tự

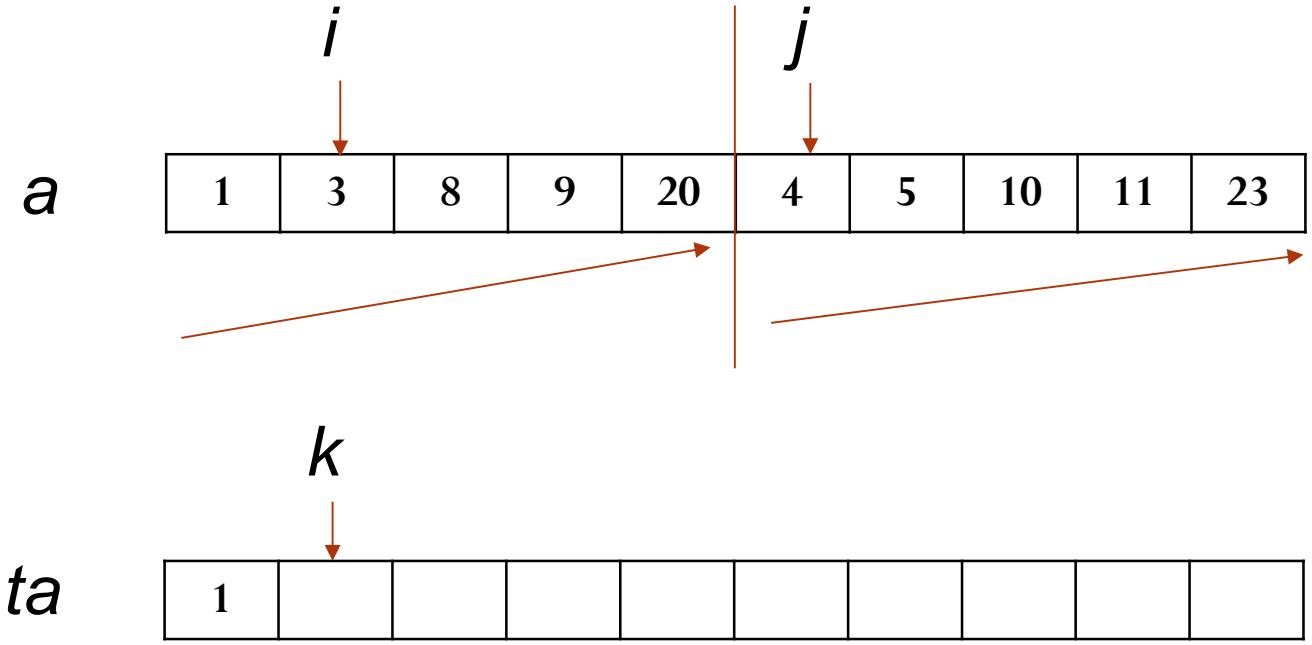
Sắp xếp trộn

- Trộn hai dãy đã được sắp xếp thành dãy mới được sắp xếp



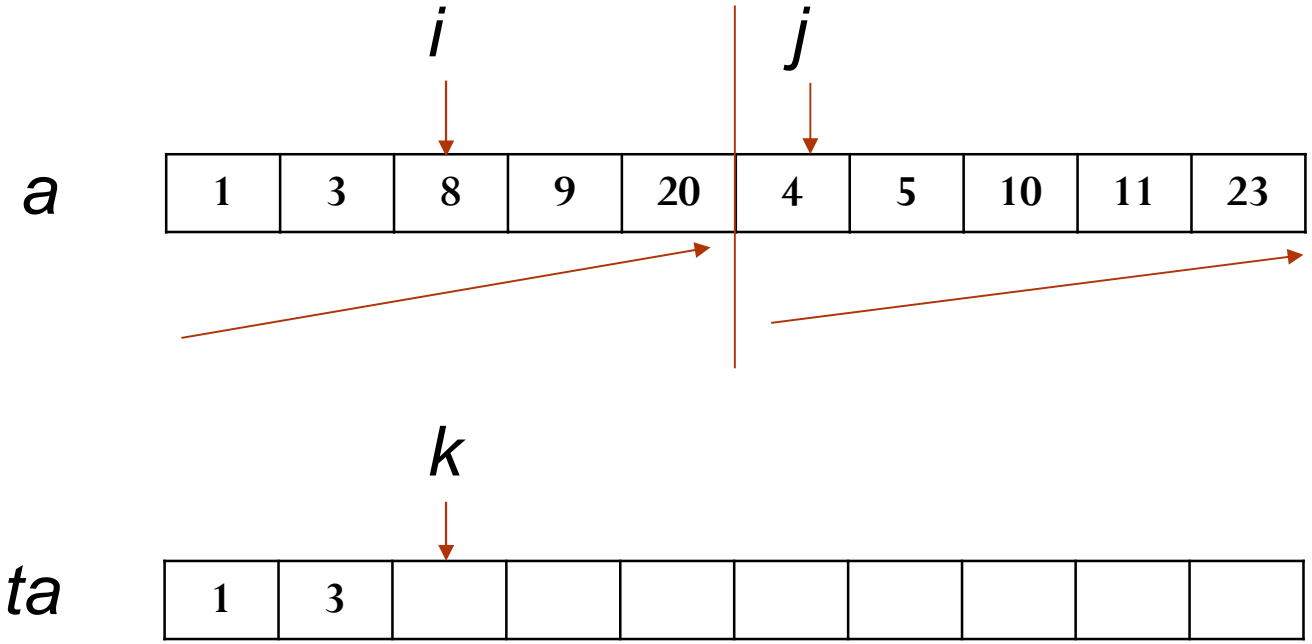
Sắp xếp trộn

- Trộn hai dãy đã được sắp xếp thành dãy mới được sắp xếp



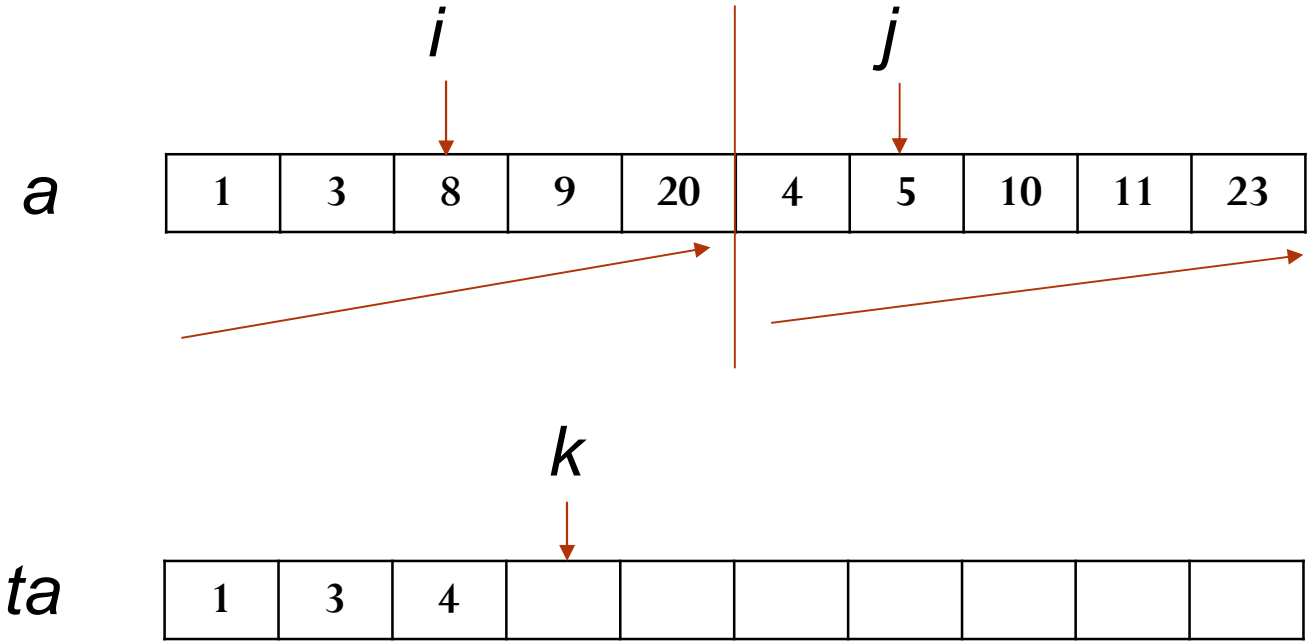
Sắp xếp trộn

- Trộn hai dãy đã được sắp xếp thành dãy mới được sắp xếp



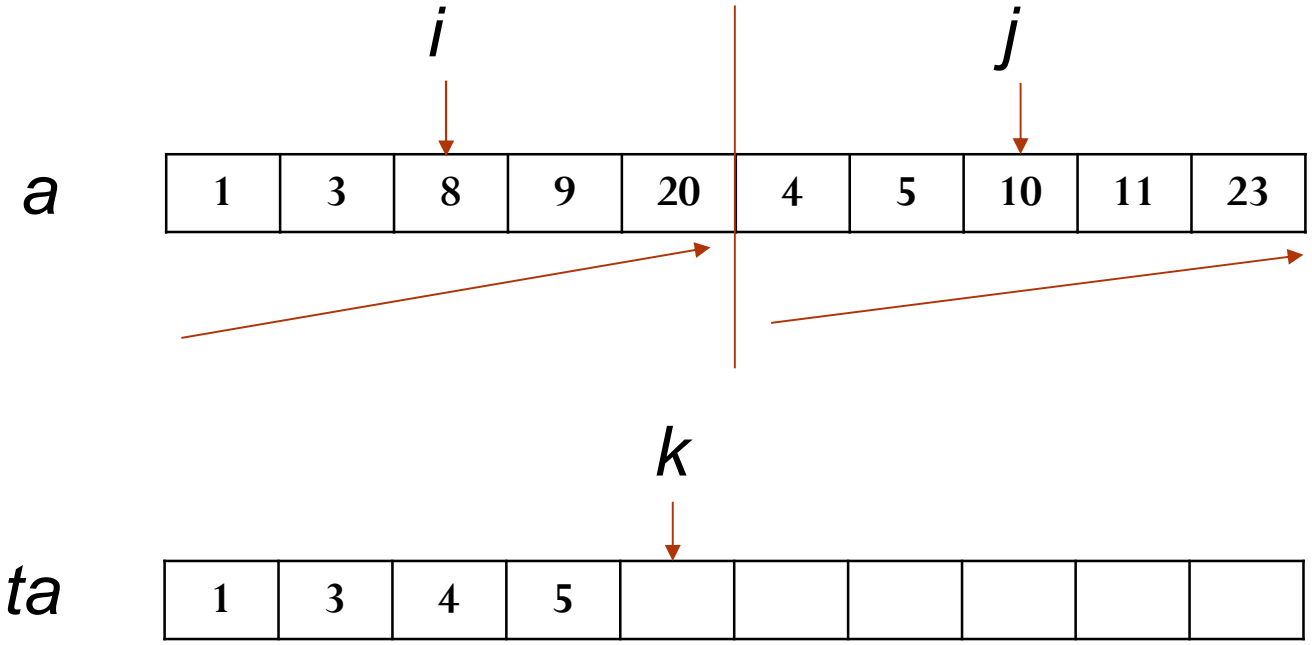
Sắp xếp trộn

- Trộn hai dãy đã được sắp xếp thành dãy mới được sắp xếp



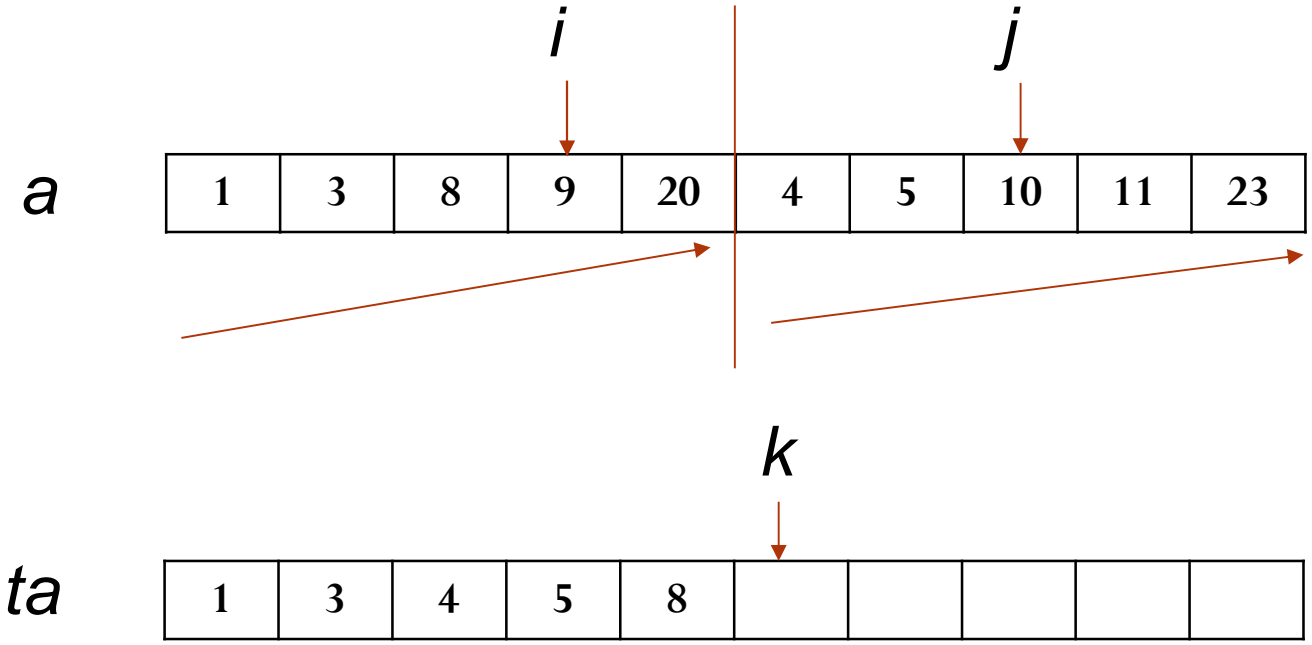
Sắp xếp trộn

- Trộn hai dãy đã được sắp xếp thành dãy mới được sắp xếp



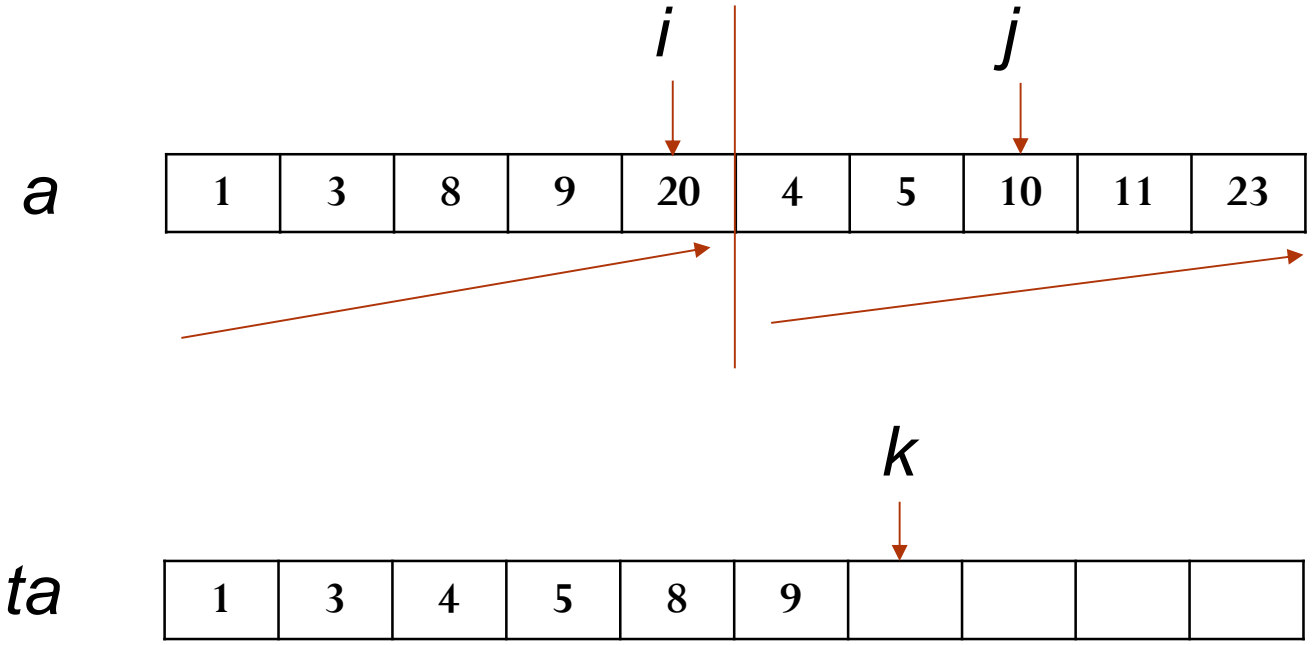
Sắp xếp trộn

- Trộn hai dãy đã được sắp xếp thành dãy mới được sắp xếp



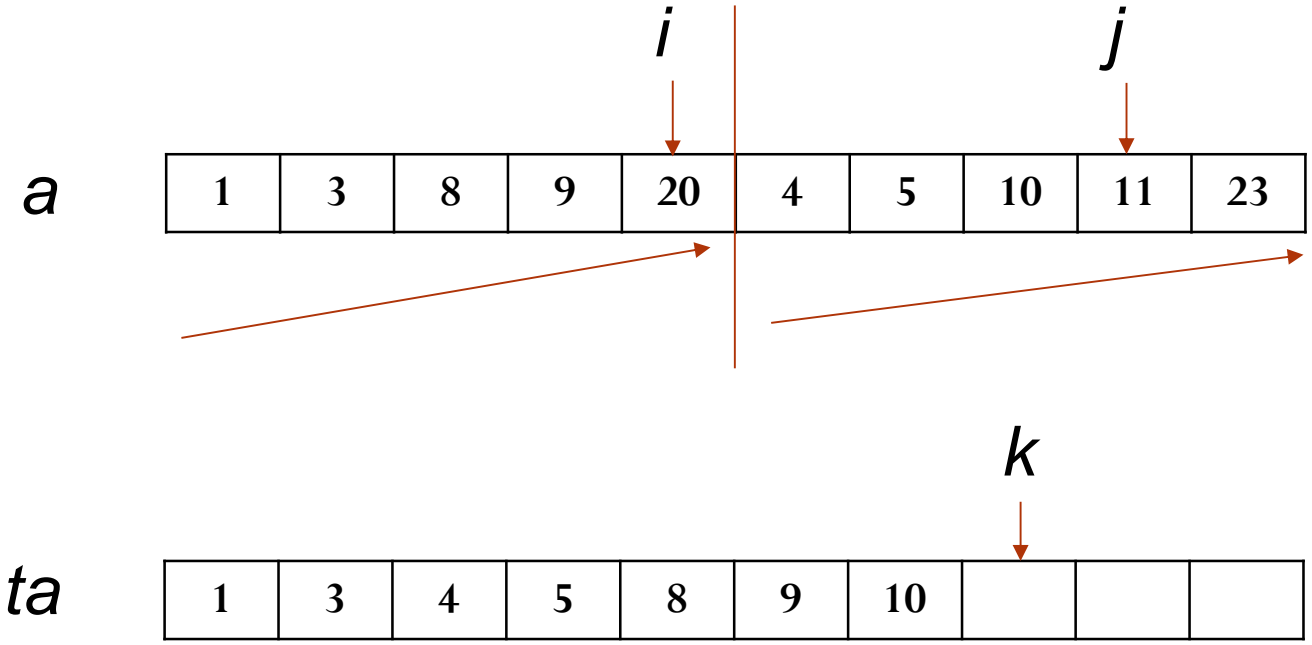
Sắp xếp trộn

- Trộn hai dãy đã được sắp xếp thành dãy mới được sắp xếp



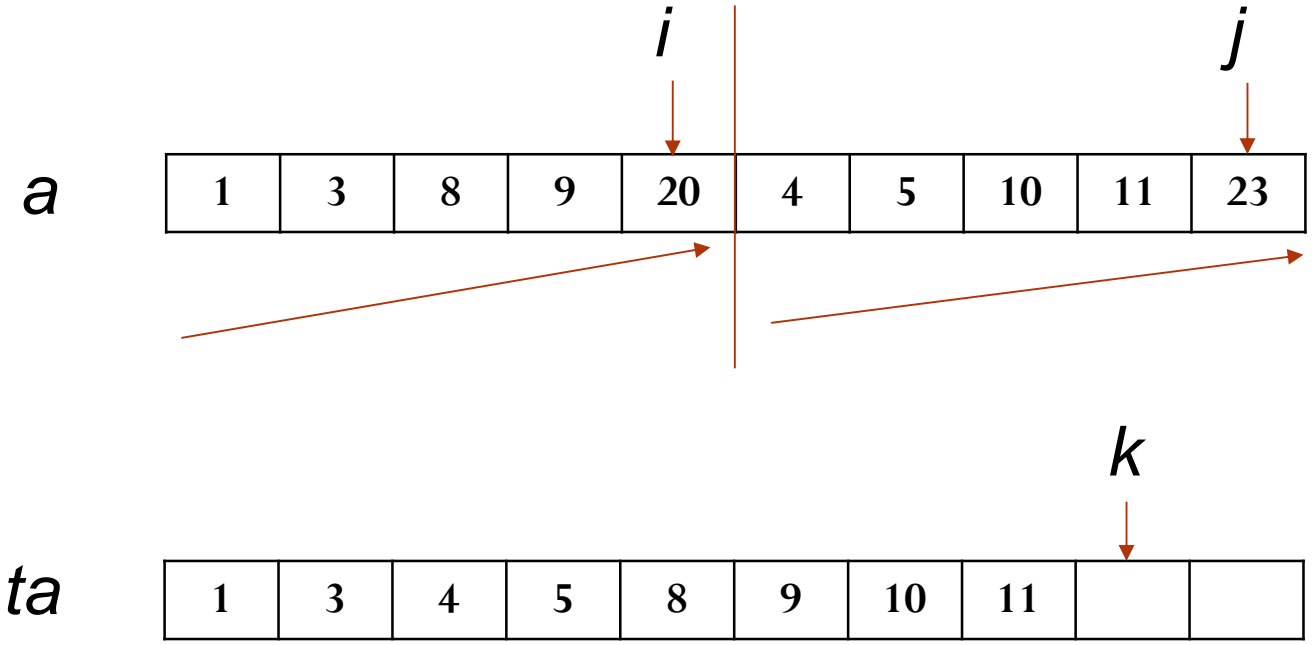
Sắp xếp trộn

- Trộn hai dãy đã được sắp xếp thành dãy mới được sắp xếp



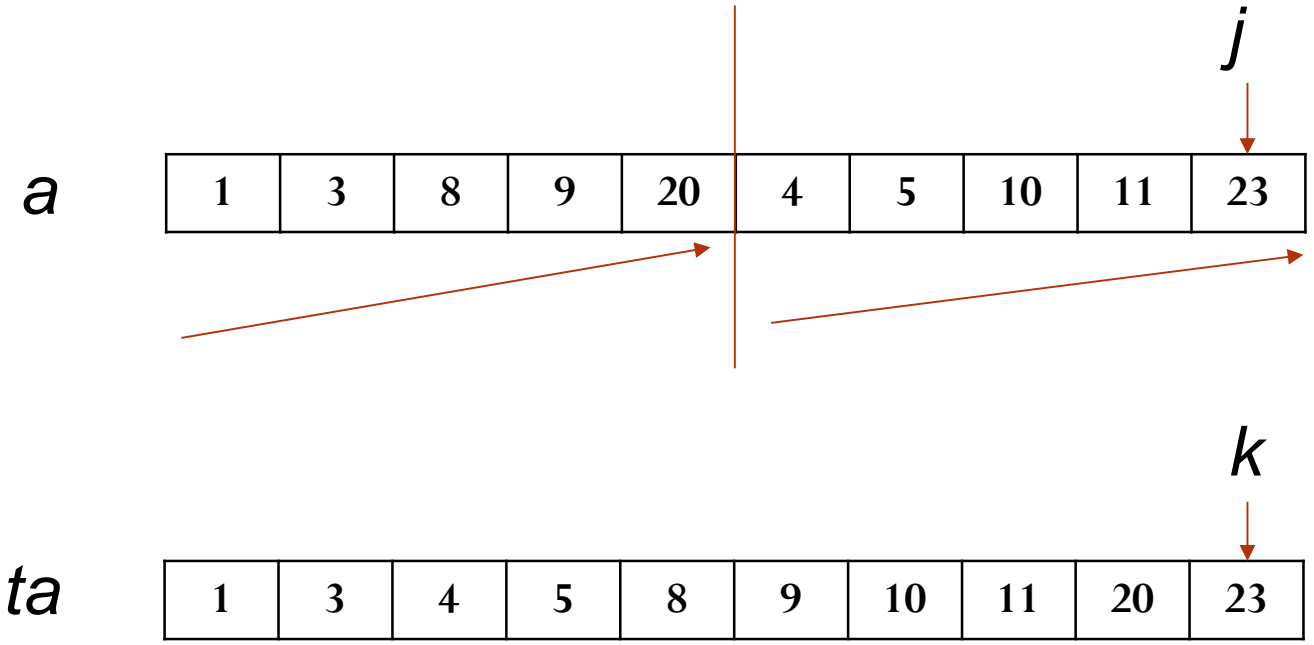
Sắp xếp trộn

- Trộn hai dãy đã được sắp xếp thành dãy mới được sắp xếp



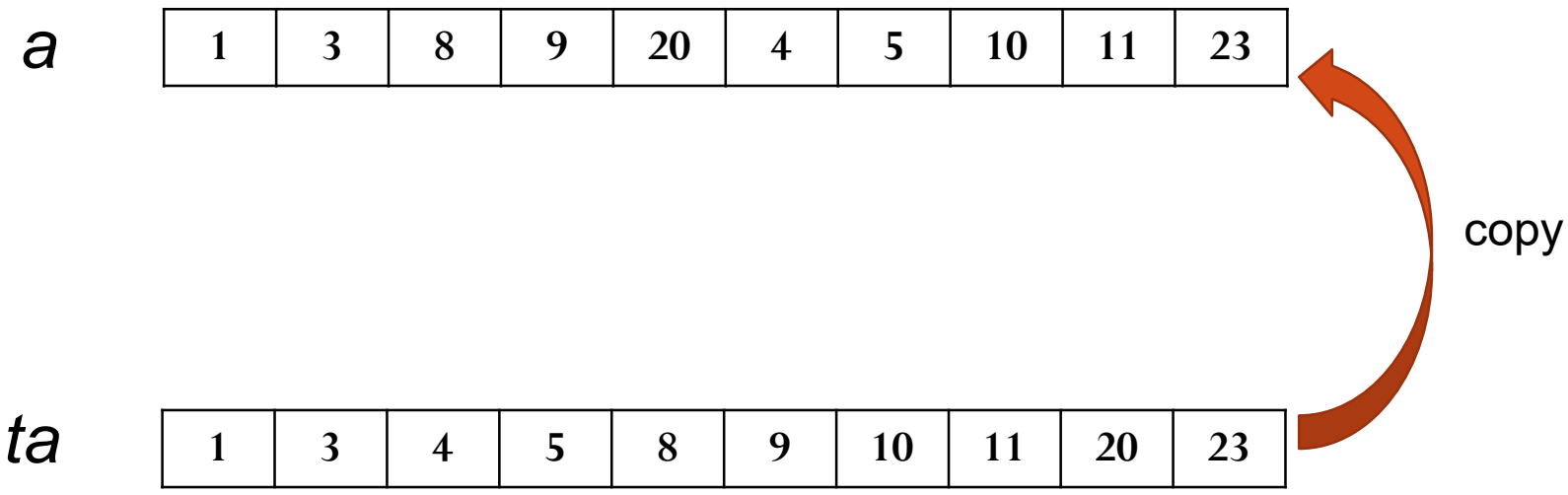
Sắp xếp trộn

- Trộn hai dãy đã được sắp xếp thành dãy mới được sắp xếp



Sắp xếp trộn

- Trộn hai dãy đã được sắp xếp thành dãy mới được sắp xếp



Sắp xếp trộn

- Trộn hai dãy đã được sắp xếp thành dãy mới được sắp xếp

a

1	3	4	5	8	9	10	11	20	23

ta

1	3	4	5	8	9	10	11	20	23

Sắp xếp trộn

```
#include <bits/stdc++.h>
using namespace std;
#define MAX 1000000
int a[MAX];
int n;
int ta[MAX];

void input(){
    cin >> n;
    for(int i = 0; i < n; i++) cin>> a[i];
}
void print(){
    for(int i = 0; i < n; i++) cout << a[i] << " ";
}
```

Sắp xếp trộn

```
void merge(int b, int m, int e){
    int i = b;
    int j = m+1;
    for(int k = b; k <= e; k++){
        if(i > m){  ta[k] = a[j]; j++; }
        else if(j > e){ta[k] = a[i]; i++;}
        else{
            if(a[i] > a[j]){ta[k] = a[j]; j++;}
            else{ta[k] = a[i]; i++;}
        }
    }
    for(int k = b; k <= e; k++) a[k] = ta[k];
}
```

Sắp xếp trộn

```
void mergeSort(int b, int e){
    if(b == e) return;
    int m = (b+e)/2;
    mergeSort(b,m);
    mergeSort(m+1,e);
    merge(b,m,e);
}
int main(){
    input();
    mergeSort(0,n-1);
    print();
    return 0;
}
```

Giảm để trị

- Chia bài toán (chia theo dữ liệu) xuất phát thành các bài toán con
- Giải 1 bài toán con và dẫn ra lời giải của bài toán xuất phát (các bài toán con khác không cần giải → tránh dư thừa)
  - Tìm kiếm nhị phân
  - Tính lũy thừa

Tìm kiếm nhị phân

Mã giả

```
bSearch(a,left, right, x){
  if left = right then{
    if a[left] = x return left; else return NOT_FOUND;
  }
  mid = (left + right)/2;
  if a[mid] = x then return mid;
  if a[mid] < x return bSearch(a,mid+1, right, x);
  else return bSearch(a,left, mid-1, x);
}
```

Độ phức tạp O(logn), với n là độ dài dãy từ chỉ số `left` đến chỉ số `right`

Tính x^n

Mã giả

```
exp(x,n){
  if n = 1 then return x;
  n2 = n/2;
  a = exp(x,n2);
  if n mod 2 = 0 then
    return a*a;
  else
    return a*a*x;
}
```

Độ phức tạp O(logn)

Bài tập ví dụ

- EXPMOD
  - Cho số nguyên dương x và N, hãy tính $x^N \bmod 10^9+7$
- TRIPLE
  - Cho dãy N số nguyên dương $a_1, a_2, \ldots, a_N$ và số nguyên dương K. Hãy đếm xem có bao nhiêu bộ chỉ số (i,j,k) sao cho $1 \le i < j < k \le N$ và $a_i + a_j + a_k = K$