2.6. Thuật toán chia để trị (Devide and Conquer)

2.6.1. Giới thiệu thuật toán

Thuật toán chia để trị (Devide and Conquer): dùng để giải lớp các bài toán có thể thực hiện được ba bước:

1. Devide (Chia). Chia bài toán lớn thành những bài toán con có cùng kiểu với bài toán lớn.
2. Conquer (Trị). Giải các bài toán con. Thông thường các bài toán con chỉ khác nhau về dữ liệu vào nên ta có thể thực hiện bằng một thủ tục đệ qui.
3. Combine (Tổng hợp). Tổng hợp lại kết quả của các bài toán con để nhận được kết quả của bài toán lớn.

Một số thuật toán chi để trị điển hình:

- Thuật toán tìm kiếm nhị phân (Binary-Search).
- Thuật toán Quick-Sort.
- Thuật toán Merge-Sort.
- Thuật toán Strassen nhân đa thức, ma trận.
- Thuật toán Kuley-Tukey nhân tính toán nhanh dịch chuyển Fourier.
- Thuật toán Karatsuba tính toán phép nhân…

2.6.2. Thuật toán nâng x lên lũy thừa n

Bước chia. Ta có .

$$X^n = \begin{cases} 1 \quad if \quad n = 0 \\ X^{n/2}.X^{n/2} \quad if \ n\%2 = 0 \\ X.X^{n/2}.X^{n/2} \quad if \ n\%2 \neq 0 \end{cases}$$

Bước trị. Tính toán $x^{n/2}$ trong trường hợp x chẵn, $x.x^{n/2}$ trong trường hợp *x* lẻ.

Bước tổng hợp. Kết quả chính là $x^{n/2}$. $x^{n/2}$ trong trường hợp x chẵn và $x.x^{n/2}$. $x^{n/2}$ trong trường hợp x lẻ.

Thuật toán:

```
int power(int x, unsigned int n){
    if( n == 0)
            return 1;
    else if (n%2 == 0)
            return power(x, n/2)*power(x, n/2);
    else
            return x*power(x, n/2)*power(x, n/2);
}
```

Ví dụ:

```
S       = power(2, 5)
        = 2* power(2, 2)* power(2, 2)
        = 2* power(2, 1)* power(2, 1)* power(2, 1)* power(2, 1)
        = 2* 2* power(0, 1)*2* power(2, 0)* 2*power(2, 0)* 2*power(2, 0)
        = 2*2*2*2*2
        =32
```

2.6.2. Thuật toán nâng x lên lũy thừa n

Thuật toán cải tiến:

```
int power(int x, unsigned int n) {
        int temp;
        if( n == 0)
                return 1;
        temp = power(x, n/2);
        if (n%2 == 0)
                return temp*temp;
        else
                return x*temp*temp;
}
```

Ví dụ:

2.6.3. Thuật toán tìm kiếm nhị phân.

Bài toán. Cho một dãy A = (a_1, a_2,..,a_n) đã được sắp xếp theo thứ tự tăng dần. Nhiệm vụ của ta là tìm vị trí của x xuất hiện trong A[].

Thuật toán Binary-Search(int Arr[], int l, int r, int x):

Input :

- Arr[] mảng số đã được sắp xếp.
- Giá trị l : vị trí bên trái nhất bắt đầu tìm kiếm.
- Giá trị r : vị trí bên phải nhất bắt đầu tìm kiếm.
- Giá trị x: số cần tìm trong mảng Arr[]

Ouput:

- Return(mid) là vị trí của x trong khoảng l, r.
- Return(-1) nếu x không có mặt trong khoảng l, r.

Formats:

Binary-Search(Arr, l, r, x)**.**

Độ phức tạp thuật toán: O(log(n)): n = l-r.

2.6.3. Thuật toán tìm kiếm nhị phân.

```
Int Binary-Search( int Arr[], int l, int r, int x ) {
        if (r >= l)   {
                int mid = l + (r - l)/2; //Bước chia:Chia bài toán làm hai phần
                if (arr[mid] == x)  //Trị trường hợp thứ nhất
                        return mid;
                if (arr[mid] > x) //Trị trường hợp thứ hai
                        return binarySearch(arr, l, mid-1, x);
                return binarySearch(arr, mid+1, r, x); //Trị trường hợp còn lại
        }
        return -1; //Kết luận tìm không thấy
}
```

Ta có thể khử đệ qui theo thủ tục dưới đây:

```
int binarySearch(int arr[], int l, int r, int x) {
        while (l <= r)   {
                int m = l + (r-l)/2; //Bước chia
                if (arr[m] == x) return m;  // Trị trường hợp 1.
                if (arr[m] < x) l = m + 1; // Trị trường hợp 2.
                else r = m - 1; // Trị trường hợp 3.
        }
        return -1; // Kết luận không tìm thấy.
}
```

Thuật toán nhân nhanh Karatsuba cho phép ta nhân nhanh hai số ở hệ cơ số bất kỳ với độ phức tạp là $O(n^{1.59})$. Thuật toán được thực hiện theo giải thuật chia để trị như sau:

Với mọi số tự nhiên *x, y* bất kỳ gồm n chữ số ở hệ cơ số B đều tồn tại số tự nhiên m<=n sao cho:

$$x = x_1 B^m + x_0 \qquad y = y_1 B^m + y_0$$

Với x_0, y_0 nhỏ hơn B^m. Khi đó:

$$xy = (x_1 B^m + x_0)(y_1 B^m + y_0)$$
$$xy = z_2 B^{2m} + z_1 B^m + z_0$$

Trong đó:

$$z_2 = x_1 y_1$$
$$z_1 = x_1 y_0 + x_0 y_1$$
$$z_0 = x_0 y_0$$

Như vậy ta chỉ cần thực hiện 4 phép nhân và một số phép cộng và một số phép trừ khi phân tích các toán hạng.

Ví dụ. Ta cần nhân hai số x = 12345, y = 6789 ở hệ cơ số B = 10. Chọn m=3 , khi đó:

$$12345 = \mathbf{12} \cdot \mathit{1000} + \mathbf{345}$$

$$6789 = \mathbf{6} \cdot \mathit{1000} + \mathbf{789}$$

Từ đó ta tính được:

$$z_2 = \mathbf{12} \times \mathbf{6} = 72$$

$$z_0 = \mathbf{345} \times \mathbf{789} = 272205$$

$$z_1 = (\mathbf{12} + \mathbf{345}) \times (\mathbf{6} + \mathbf{789}) - z_2 - z_0 = 357 \times 795 - 72 - 272205$$

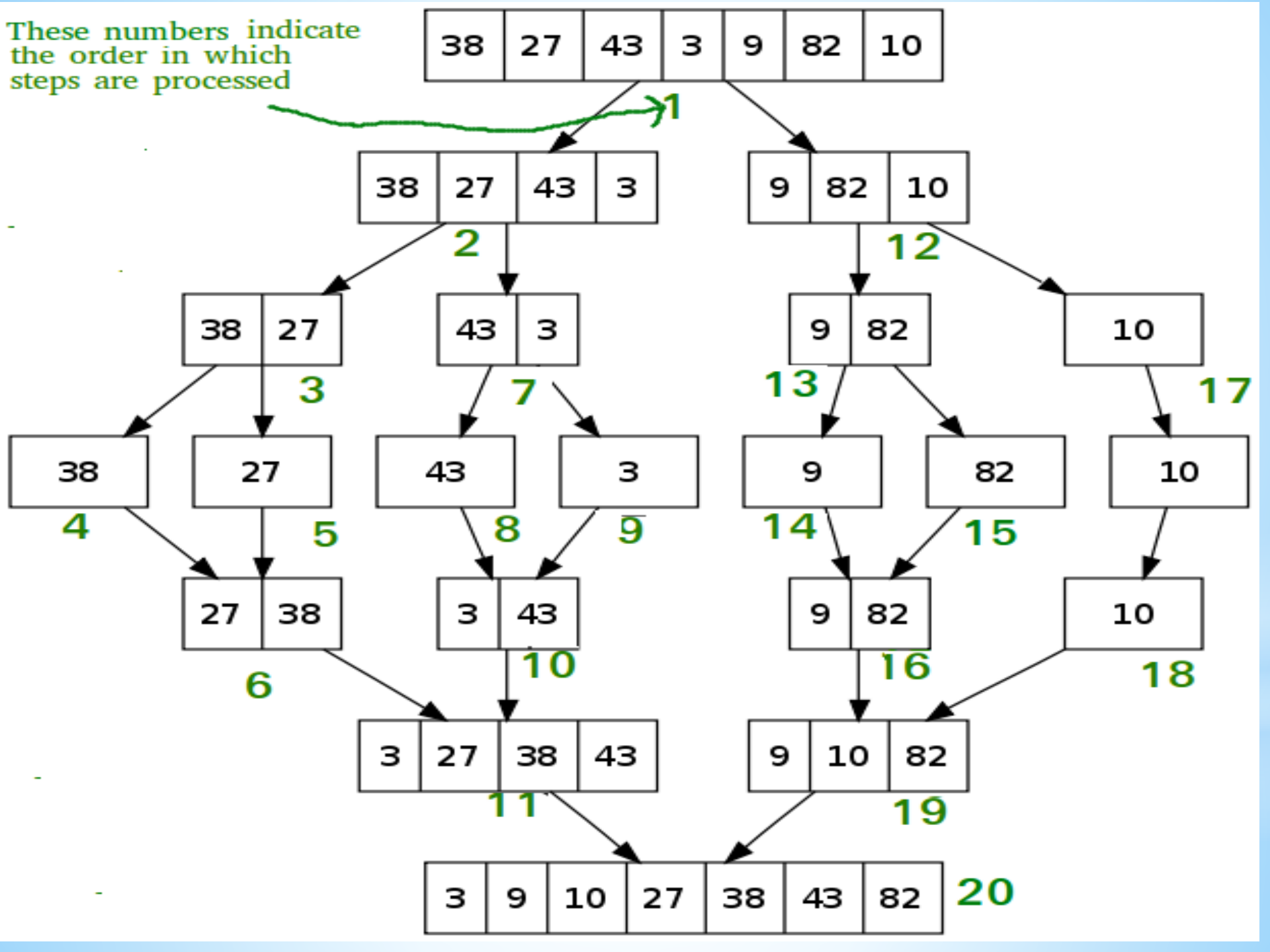
$$= 283815 - 72 - 272205 = 11538.$$

Kết quả $= 72 \cdot \mathit{1000}^2 + 11538 \cdot \mathit{1000} + 272205 = \mathbf{83810205}$.

Chú ý. Thời gian thuật toán sẽ thực hiện nhanh hơn khi ta lấy cơ số B cao (ví dụ B = 1000.

Thuật toán Merge Sort:

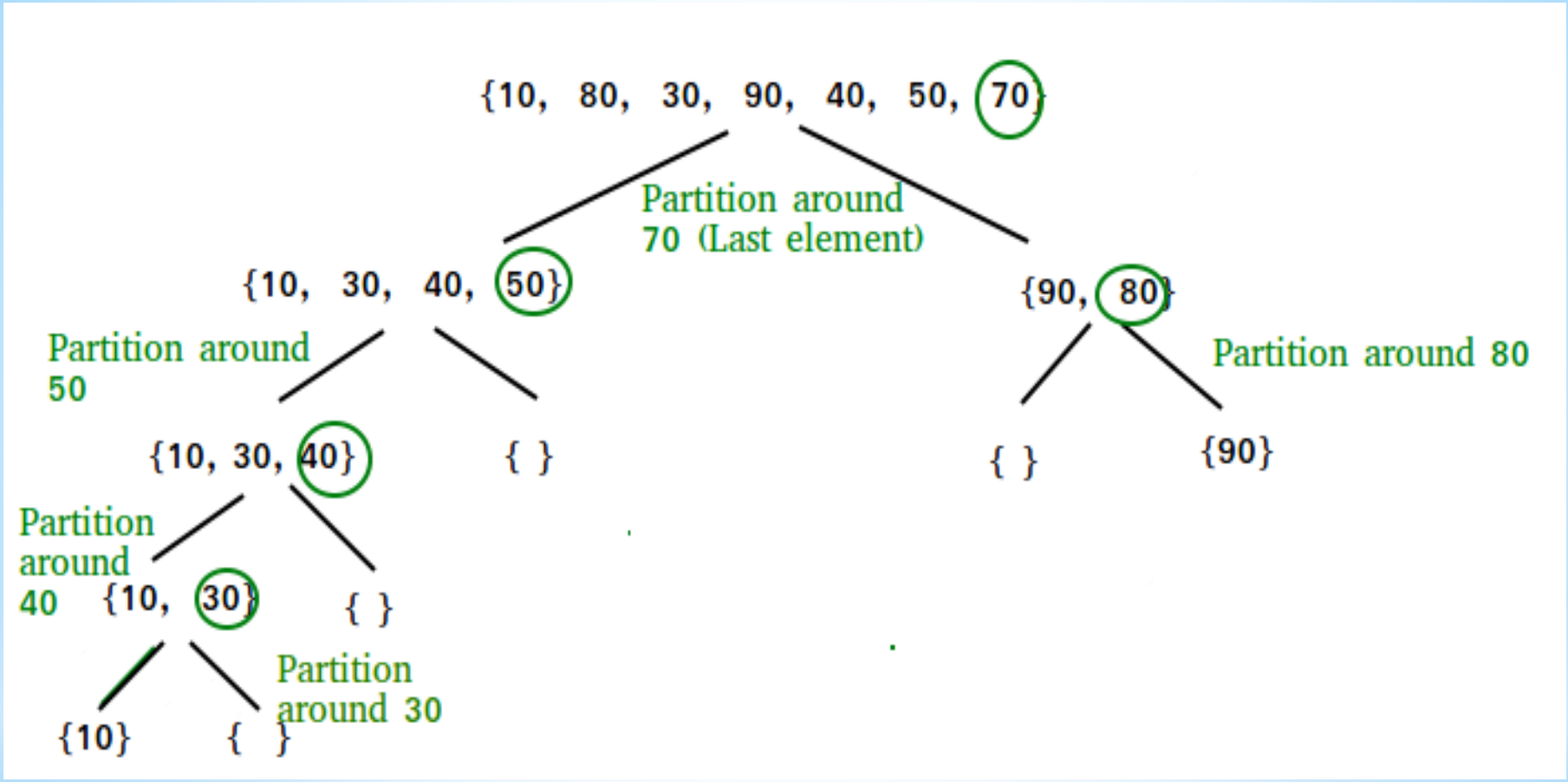
```
void  mergeSort(int A[],  int l,  int r) {
        if (l < r)  {
                int m = l+(r-l)/2; //lấy m là phần tử ở giữa
                mergeSort(A, l, m); //trị nửa thứ nhất
                mergeSort(A, m+1, r); //trị nửa thứ hai
                merge(A, l, m, r); //hòa nhập hai nửa
        }
}
```


These numbers indicate the order in which steps are processed
38 27 43 3 9 82 10
1
38 27 43 3
9 82 10
2
12
38 27
43 3
9 82
10
3
7
13
17
38
27
43
3
9
82
10
4
5
8
9
14
15
27 38
3 43
9 82
10
6
10
16
18
3 27 38 43
9 10 82
11
19
3 9 10 27 38 43 82
20

Thuật toán Quick Sort: sử dụng thủ tục phân đôi dãy theo khóa

```
int partition (int A[], int low, int high)  {
        int pivot = A[high]; // chọn khóa ở vị trí cuối: high
        int i = (low - 1); // lấy i ở vị trí low-1
        for (int j = low; j <= high - 1; j++) {  //duyệt trong đoạn [low, high-1]
                if (A[j] < pivot){ //nếu A[j] bé hơn khóa
                        i++; swap(A[i], A[j]); //tăng i và đổi nội dung
                }
        }
        swap(A[i + 1], A[high]);//đổi nội dung A[i+1] va A[high]
        return (i + 1);//đây cũng là vị trí của khóa
}
void quickSort(int A[], int low, int high) {
        if (low < high) {  //nếu cận dưới bé hơn cận trên
                int pi = partition(A, low, high); //chia mảng thành hai nửa
                quickSort(A, low, pi - 1);  //quicksort nửa thứ nhất
                quickSort(A, pi + 1, high); //quickSort nửa thứ 2
        }
}
```

Thuật toán Quick Sort:



2.6.5. Tìm tổng dãy con liên tục lớn nhất

Bài toán: Cho dãy số nguyên bao gồm cả số âm lẫn số dương. Nhiệm vụ của ta là tìm dãy con liên tục có tổng lớn nhất.

Ví dụ. Với dãy số A = {-2, -5, **6, -2, -3, 1, 5**, -6} thì tổng lớn nhất của dãy con liên tục ta nhận được là 7.

Thuật toán 1. Max-SubArray(Arr[], n): //*Độ phức tạp* $O(n^2)$.

Bước 1 (Khởi tạo):

Max = Arr[0]; // *Chọn Max là phần tử đầu tiên*.

Bước 2 (lặp):

```
for (i=1; i<n; i++) { //Duyệt từ vị trí 1 đến n-1
        S = 0; //Gọi S là tổng liên tục của I số
        for ( j =0; j<=n; j++) { //tính tổng của Arr[0],..,Arr[i].
                S = S + Arr[j];
                        if (Max < S ) // Nếu Max nhỏ hơn
                                Max = S;
        }
}
```

Bước 3 (Trả lại kết quả):

Return(Max).

Thuật toán 2. Devide-Conquer (Arr[], n): //*Độ phức tạp* O(nlog(n).

```
int maxCrossingSum(int arr[], int l, int m, int h) {
        int sum = 0, left_sum = INT_MIN, right_sum = INT_MIN;
        for (int i = m; i >= l; i--)  { //Tìm tổng dãy con từ l đến m
                sum = sum + arr[i];
                if (sum > left_sum)    left_sum = sum;
        }
        sum = 0;
        for (int i = m+1; i <= h; i++)  { // //Tìm tổng dãy con từ m+1 đến h
                sum = sum + arr[i];
                if (sum > right_sum)    right_sum = sum;
        }
        return left_sum + right_sum; //Trả lại kết quả
}

int maxSubArraySum(int arr[], int l, int h) {
        if (l == h)  return arr[l];
        int m = (l + h)/2; // Tìm điểm ở giữa
        return max(maxSubArraySum(arr, l, m),
                        maxSubArraySum(arr, m+1, h),
                        maxCrossingSum(arr, l, m, h));
}
```

BÀI 1. LŨY THỪA

Cho số nguyên dương N và K. Hãy tính N^K modulo 10^9+7.

Input:

- Dòng đầu tiên là số lượng bộ test T ($T \leq 20$).
- Mỗi test gồm 1 số nguyên N và K ($1 \leq N \leq 1000$, $1 \leq K \leq 10^9$).

Output:

- Với mỗi test, in ra đáp án trên một dòng.

Ví dụ:

Input:	Output
2 2 3 4 2	8 16

BÀI 13. LŨY THỪA ĐẢO

Cho mảng số N. Ta gọi số đảo của N là R. Hãy tìm lũy thừa R của N. Đưa ra kết quả của bài toán dưới dạng modulo với $10^9 + 7$.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm là số N được ghi trên một dòng.
- T, N thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10^{10}$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input	Output
2 2 12	4 864354781

BÀI 10. SỐ FIBONACCI THỨ N

Dãy số Fibonacci được xác định bằng công thức như sau:

F[0] = 0, F[1] = 1;

F[n] = F[n-1] + F[n-2] với mọi n >= 2.

Các phần tử đầu tiên của dãy số là 0, 1, 1, 2, 3, 5, 8, ...

Nhiệm vụ của bạn là hãy xác định số Fibonaci thứ n. Do đáp số có thể rất lớn, in ra kết quả theo modulo 10^9+7.

Input:

Dòng đầu tiên là số lượng bộ test T ($T \leq 1000$).

Mỗi test bắt gồm một số nguyên N ($1 \leq N \leq 10^9$).

Output:

Với mỗi test, in ra đáp án trên một dòng.

Ví dụ:

Input:	**Output**
3 2 6 20	1 8 6765

BÀI 11. LŨY THỪA MA TRẬN

Cho ma trận vuông A kích thước N x N. Nhiệm vụ của bạn là hãy tính ma trận X = A^K với K là số nguyên cho trước. Đáp số có thể rất lớn, hãy in ra kết quả theo modulo 10^9+7.

Input:

Dòng đầu tiên là số lượng bộ test T ($T \leq 100$).

Mỗi test bắt gồm một số nguyên N và K ($1 \leq N \leq 10$, $1 \leq K \leq 10^9$) là kích thước của ma trận và số mũ.

Output:

Với mỗi test, in ra kết quả của ma trận X.

Ví dụ:

Input:	Output
2 2 5 1 1 1 0 3 1000000000 1 2 3 4 5 6 7 8 9	8 5 5 3 597240088 35500972 473761863 781257150 154135232 527013321 965274212 272769492 580264779

BÀI 5. DÃY XÂU FIBONACI

Một dãy xâu ký tự G chỉ bao gồm các chữ cái A và B được gọi là dãy xâu Fibonacci nếu thỏa mãn tính chất: $G(1) = A; G(2) = B; G(n) = G(n-2)+G(n-1)$. Với phép cộng (+) là phép nối hai xâu với nhau. Bài toán đặt ra là tìm ký tự ở vị trí thứ i (tính từ 1) của xâu Fibonacci thứ n.

Dữ liệu vào: Dòng 1 ghi số bộ test. Mỗi bộ test ghi trên một dòng 2 số nguyên N và i ($1<N<93$). Số i đảm bảo trong phạm vi của xâu G(N) và không quá 18 chữ số. **Kết quả:** Ghi ra màn hình kết quả tương ứng với từng bộ test.

Input	Output
2 6 4 8 19	A B

BÀI 6. HỆ CƠ SỐ K

Cho hai số A, B ở hệ cơ số K. Hãy tính tổng hai số đó ở hệ cơ số K.

Input: Chỉ có 1 dòng ghi 2 số K,A,B

($2 \le K \le 10$; A và B nếu biểu diễn trong hệ cơ số 10 đều nhỏ hơn 10^9)

Output: In ra tổng của A và B trong hệ cơ số K

Ví dụ:

Input	Output
2 1 10	11

BÀI 16. TÍCH HAI SỐ NHỊ PHÂN

Cho hai xâu nhị phân biểu diễn hai số. Nhiệm vụ của bạn là đưa ra tích của hai số. Ví dụ với xâu S1="1100" và S2="1010" ta sẽ có kết quả là 120.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm 2 hai xâu nhị phân S1, S2 được viết trên một dòng.
- T, S1, S2 thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq$ length(S1), length(S2) ≤ 30.

Output:

- Đưa ra tích của mỗi test theo từng dòng.

Input	Output
2 1100 01 01 01	12 1

BÀI 2. TÌM KIẾM NHỊ PHÂN

Cho dãy số A[] gồm có N phần tử đã được sắp xếp tăng dần và số K.

Nhiệm vụ của bạn là kiểm tra xem số K có xuất hiện trong dãy số hay không. Nếu có hãy in ra vị trí trong dãy A[], nếu không in ra "NO".

Input:

Dòng đầu tiên là số lượng bộ test T ($T \leq 10$).

Mỗi test bắt đầu bằng số nguyên N và K ($N \leq 100\,000$, $0 \leq K \leq 10^6$).

Dòng tiếp theo gồm N số nguyên A[i] ($0 \leq A[i] \leq 10^6$), các phần tử là riêng biệt.

Output:

Với mỗi test in ra trên một dòng đáp án tìm được.

Test ví dụ:

Input:	Output
2 5 3 1 2 3 4 5 6 5 0 1 2 3 9 10	3 NO

BÀI 19. QUAY DÃY SỐ 1

Cho mảng đã được sắp xếp A[] gồm N phần tử. Các phần tử của mảng A[] có thể giống nhau. Mảng A[] đã được quay vòng phải K lần. Hãy tìm số lần quay vòng K.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng thứ nhất đưa vào số M, N, K; dòng tiếp theo đưa vào N số của mảng A[]các số được viết cách nhau một vài khoảng trống.
- T, N, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10^7$; $0 \leq A[i] \leq 10^{18}$.

Output:

- Đưa ra số lần quay vòng K của mỗi test theo từng dòng.

Input	Output
2 5 5 1 2 3 4 5 1 2 3 4 5	1 0

BÀI 17. TÍNH FLOOR(X)

Cho mảng đã được sắp xếp A[] gồm N phần tử không có hai phần tử giống nhau và số X. Nhiệm vụ của bạn là tìm floor(X). Trong đó, K=floor(X) là phần tử lớn nhất trong mảng A[] lớn hơn hoặc bằng X.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng thứ nhất đưa vào số N là số phần tử của mảng A[] và số X; dòng tiếp theo đưa vào N số của mảng A[]; các số được viết cách nhau một vài khoảng trống.
- T, N, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10^7$; $1 \leq A[i] \leq 10^{18}$.

Output:

- Đưa ra vị trí của floor(X) trong mảng A[] hoặc -1 nếu không tồn tại floor(X) của mỗi test theo từng dòng.

Input	Output
3 7 0 1 2 8 10 11 12 19 7 5 1 2 8 10 11 12 19 7 10 1 2 8 10 11 12 19	-1 2 4

BÀI 20. QUAY DÃY SỐ 2

Cho hai mảng đã được sắp xếp A[] gồm N phần tử. Các phần tử của mảng A[] đều khác nhau. Mảng A[] đã được quay vòng với số lần không biết trước. Hãy tìm phần tử nhỏ nhất của mảng A[].

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng thứ nhất đưa vào số M, N, K; dòng tiếp theo đưa vào N số của mảng A[]các số được viết cách nhau một vài khoảng trống.
- T, N, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10^7$; $0 \leq A[i] \leq 10^{18}$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input	Output
2 5 4 5 1 2 3 6 10 20 30 40 50 5	1 5

BÀI 21. PHẦN TỬ KHÁC NHAU.

Cho hai mảng đã được sắp xếp A[] và B[] gồm N và N-1 phần tử. Các phần tử của mảng A[] chỉ khác mảng B[] một phần tử duy nhất. Hãy tìm vị trí của phần tử khác nhau giữa A[] và B[].

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm 3 dòng: dòng thứ nhất đưa vào số N; dòng tiếp theo đưa vào N số của mảng A[];dòng tiếp theo đưa vào N-1 số của mảng B[]; các số được viết cách nhau một vài khoảng trống.
- T, N, A[i], B[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10^7$; $0 \leq A[i] \leq 10^{18}$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input	Output
2 7 2 4 6 8 9 10 12 2 4 6 8 10 12 6 3 5 7 9 11 13 3 5 7 11 13	5 4

BÀI 12. CẶP NGHỊCH THẾ

Cho mảng A[] gồm N phần. Ta gọi cặp nghịch thế của mảng A[] là số các cặp i, j sao cho i<j và A[i]>A[j]. Đối với mảng đã được sắp xếp thì số cặp nghịch thế bằng 0. Mảng đã sắp theo thứ tự giảm dần có số đảo ngược cực đại. Nhiệm vụ của bạn là hãy đưa ra số cặp nghịch thế của mảng A[] gồm N phần tử.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất đưa vào số N tương ứng với số phần tử của mảng A[]; phần thứ 2 là N số của mảng A[]; các số được viết cách nhau một vài khoảng trống.
- T, N, A[i] thỏa mãn ràng buộc: $1 \le T \le 100$; $1 \le N \le 10^7$; $1 \le A[i] \le 10^{18}$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input	Output
2 5 2 4 1 3 5 5 5 4 3 2 1	3 10

BÀI 15. DÃY CON LIÊN TIẾP CÓ TỔNG LỚN NHẤT

Cho mảng A[] gồm N số có cả các số âm và số dương. Nhiệm vụ của bạn là tìm mảng con liên tục có tổng lớn nhất của mảng. Ví dụ với mảng A[]={-2, -5, 6, -2,-3, 1, 5, -6} ta có kết quả là 7 tương ứng với dãy con {6, -2, -3, 1, 5}.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng thứ nhất đưa vào hai số N tương ứng với số phần tử của mảng; dòng tiếp theo đưa vào N số A[i]; các số được viết cách nhau một vài khoảng trống.
- T, N, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 100$; $-100 \leq A[i] \leq 100$.

Output:

- Đưa ra tổng con liên tục lớn nhất của mỗi test theo từng dòng.

Input	Output
1 8 -2 -5 6 -2 -3 1 5 -6	7

BÀI 9. CẶP ĐIỂM GẦN NHẤT

Cho N điểm trên mặt phẳng tọa độ Oxy. Bạn cần tìm khoảng cách ngắn nhất giữa hai điểm trong số N điểm đã cho.

Input:

Dòng đầu tiên là số lượng bộ test T ($T \leq 20$).

Mỗi test bắt đầu bởi một số nguyên N ($1 \leq N \leq 100\,000$).

N dòng tiếp theo, mỗi dòng gồm 2 số nguyên X[i], Y[i] ($-10^6 \leq X[i], Y[i] \leq 10^6$).

Output:

Với mỗi test, in ra đáp án trên một dòng với độ chính xác 6 chữ số sau dấu phảy.

Ví dụ:

Input:	Output
2 6 2 3 12 30 40 50 5 1 12 10 3 4 3 0 0 3 0 4 0	1.414214 1.000000