

Chuyên đề 3

SẮP XẾP

Sắp xếp là quá trình bố trí lại vị trí các đối tượng của một danh sách theo một trật tự nhất định. Sắp xếp đóng vai trò rất quan trọng trong cuộc sống nói chung và trong tin học nói riêng, thử hình dung xem, một cuốn từ điển, nếu các từ không được sắp xếp theo thứ tự, sẽ khó khăn như thế nào trong việc tra cứu các từ. Theo D.Knuth thì 40% thời gian tính toán của máy tính là dành cho việc sắp xếp. Không phải ngẫu nhiên thuật toán sắp xếp nhanh (Quick Sort) được bình chọn là một trong 10 thuật toán tiêu biểu của thế kỉ 20.

Do đặc điểm dữ liệu (kiểu số hay phi số, kích thước bé hay lớn, lưu trữ ở bộ nhớ trong hay bộ nhớ ngoài, truy cập tuần tự hay ngẫu nhiên...) mà người ta có các thuật toán sắp xếp khác nhau. Trong chuyên đề này, chúng ta chỉ quan tâm đến các thuật toán sắp xếp trong trường hợp dữ liệu được lưu trữ ở bộ nhớ trong (nghĩa là toàn bộ dữ liệu cần sắp xếp phải được đưa vào bộ nhớ chính của máy tính).

1. Phát biểu bài toán

Giả sử các đối tượng cần sắp xếp được biểu diễn bởi bản ghi gồm một số trường. Một trong các trường đó được gọi là *khoá sắp xếp*. Kiểu của khoá là kiểu có thứ tự (chẳng hạn, kiểu số nguyên, kiểu số thực,...)

```
const
  MAX      = ...;
type
  object   = record
    key : keyType;
    [các trường khác]
  end;
  TArray  = array[1..MAX] of object;
var
  a        : TArray;
  n        : longint;
```

Bài toán sắp xếp được phát biểu như sau: Cho mảng a các đối tượng, cần sắp xếp lại các thành phần (phần tử) của mảng a để nhận được mảng a mới với các thành phần có các giá trị khoá tăng dần:

$$a[1].key \leq a[2].key \leq \dots \leq a[n].key$$

2. Các thuật toán sắp xếp thông dụng

Hai thuật toán hay được sử dụng nhiều trong thực tế đó là thuật toán sắp xếp nổi bọt (BUBBLE SORT) và thuật toán sắp xếp nhanh (QUICK SORT).

2.1 Thuật toán sắp xếp nổi bọt (Bubble Sort)

Ý tưởng cơ bản của thuật toán là tìm và đổi chỗ các cặp phần tử kề nhau sai thứ tự (phần tử đứng trước có khoá lớn hơn khoá của phần tử đứng sau) cho đến khi không tồn tại cặp nào sai thứ tự (dãy được sắp xếp).

Cụ thể:

- Lượt 1: ta xét từ cuối dãy, nếu gặp 2 phần tử kề nhau mà sai thứ tự thì đổi chỗ chúng cho nhau. Sau lượt 1, phần tử có khoá nhỏ thứ nhất được đưa về vị trí 1.
- Lượt 2: ta xét từ cuối dãy (chỉ đến phần tử thứ 2), nếu gặp 2 phần tử kề nhau mà sai thứ tự thì đổi chỗ chúng cho nhau. Sau lượt 2, phần tử có khoá nhỏ thứ hai được đưa về vị trí 2.
- Lượt i : ta xét từ cuối dãy về (chỉ đến phần tử thứ i , vì phần đầu dãy từ 1 đến $i-1$ đã được xếp đúng thứ tự), nếu gặp 2 phần tử kề nhau mà sai thứ tự thì đổi chỗ chúng cho nhau. Sau lượt i , phần tử có khoá nhỏ thứ i được đưa về vị trí i .

Xong lượt thứ $n-1$ thì dãy được sắp xếp xong.

```
procedure BoubleSort;
var i, j : integer;
    tmp : object;
begin
    for i := 1 to n-1 do
        for j := n downto i+1 do
            if a[j-1].key > a[j].key then
begin
    tmp := a[j];
```

```

    a[j] := a[j-1];
    a[j-1] := tmp;
end;
end;

```

Đánh giá độ phức tạp

Số phép toán so sánh $a[j - 1].key > a[j].key$ được dùng để đánh giá hiệu suất thuật toán về mặt thời gian cho thuật toán sắp xếp nổi bọt. Tại lượt thứ i ta cần $n - i$ phép so sánh. Như vậy tổng số phép so sánh cần thiết:

$$(n - 1) + (n - 2) + \dots + 1 = \frac{n(n - 1)}{2}$$

Thuật toán có độ phức tạp $O(N^2)$

Một thuật toán sắp xếp đơn giản, hay sử dụng khác cũng cho độ phức tạp $O(N^2)$

```

for i:=1 to n-1 do
    for j:=i+1 to n do
        if a[i].key>a[j].key then begin
            tmp:=a[i];
            a[i]:=a[j];
            a[j]:=tmp;
        end;
    
```

2.2. Thuật toán sắp xếp nhanh (Quick Sort)

Ý tưởng của thuật toán như sau: Để sắp xếp dãy coi như là sắp xếp đoạn từ chỉ số 1 đến chỉ số n . Để sắp xếp một đoạn trong dãy, nếu đoạn chỉ có một phần tử thì dãy đã được sắp xếp, ngược lại ta chọn một phần tử x trong đoạn đó làm "chốt", mọi phần tử có khoá nhỏ hơn khoá của "chốt" được xếp vào vị trí đúng trước chốt, mọi phần tử có khoá lớn hơn khoá của "chốt" được xếp vào vị trí đúng sau chốt. Sau phép hoán chuyển như vậy thì đoạn đang xét được chia làm hai đoạn mà mọi phần tử trong đoạn đầu đều có khoá \leq khoá của "chốt" và mọi phần tử trong đoạn sau đều có khoá \geq khoá của "chốt". Tiếp tục sắp xếp kiểu như vậy với 2 đoạn con, ta sẽ được đoạn đã cho được sắp xếp theo chiều tăng dần của khoá.

Cụ thể:

Giả sử phải sắp xếp đoạn có chỉ số từ L đến H:

- chọn x là một phần tử ngẫu nhiên trong đoạn $L..H$ (có thể chọn x là phần tử ở giữa đoạn, nghĩa là $x = a[(L+H) \text{ div } 2]$)
- cho i chạy từ L sang phải, j chạy từ H sang trái; nếu phát hiện một cặp ngược thứ tự: $i \leq j$ và $a[i].key \geq x.key \geq a[j].key$ thì đổi chỗ 2 phần tử đó; cho đến khi $i > j$. Lúc đó dãy ở tình trạng: khoá các phần tử đoạn $L..i \leq$ khoá của x ; khoá của các phần tử đoạn $j..H \geq$ khoá của x . Tiếp tục sắp xếp như vậy với 2 đoạn $L..j$ và $i..H$.

Thủ tục QuickSort(L,H) sau, sắp xếp đoạn từ L tới H , để sắp xếp dãy số ta gọi QuickSort($1,n$)

```

procedure QuickSort (L,H:longint);
var i,j      :longint;
x,tmp      :object;
begin
  i:=L;
  j:=H;
  //x:=a [random (H-L+1)+L];
  x:=a [ (L+H) div 2];
  repeat
    while a[i].key<x.key do inc(i);
    while a[j].key>x.key do dec(j);
    if i<=j then
      begin
        tmp:=a[i];
        a[i]:=a[j];
        a[j]:=tmp;
        inc(i);
        dec(j);
      end;
    until i>j;
    if L<j then QuickSort (L,j);
    if i<H then QuickSort (i,H);
end;
```

Đánh giá độ phức tạp

Việc chọn chốt để phân đoạn quyết định hiệu quả của thuật toán, nếu việc chọn chốt không tốt rất có thể việc phân đoạn bị suy biến thành trường hợp xấu (phân

thành hai đoạn mà số phần tử của hai đoạn chênh lệch nhiều) khiến Quick Sort hoạt động chậm. Các tính toán độ phức tạp chi tiết cho thấy thuật toán Quick sort:

- Có thời gian thực thi cỡ $O(n \log n)$ trong trường hợp trung bình.
- Có thời gian thực thi cỡ $O(n^2)$ trong trường hợp xấu nhất (2 đoạn được chia thành một đoạn $n-1$ và một đoạn 1 phần tử). Khả năng để xảy ra trường hợp này là rất ít, còn nếu chọn chốt ngẫu nhiên, hầu như sẽ không xảy ra.

2.3. Nhận xét

Nếu chương trình ít gọi tới thủ tục sắp xếp và chỉ trên tập dữ liệu nhỏ, thì việc sử dụng một thuật toán phức tạp (tuy có hiệu quả hơn) có thể không cần thiết, khi đó có thể sử dụng thuật toán đơn giản có độ phức tạp $O(N^2)$, dễ cài đặt. Tuy nhiên, vì độ phức tạp $O(n^2)$, nghĩa là thời gian thực hiện tăng lên gấp 4 khi số lượng phần tử tăng lên gấp đôi. Do đó, trong trường hợp sắp xếp trên tập dữ liệu lớn nên sử dụng thuật toán sắp xếp nhanh có độ phức tạp cỡ $O(n \log n)$.

3. Sắp xếp bằng đếm phân phối (Distribution Counting)

Trong trường hợp khoá các phần tử $a[1], a[2], \dots, a[n]$ là các số nguyên nằm trong khoảng từ 0 tới K ta có thuật toán *đơn giản* và *hiệu quả* như sau:

Xây dựng dãy $c[0], c[1], \dots, c[K]$, trong đó $c[V]$ là số lần xuất hiện khoá V trong dãy.

```
for V := 0 to K do c[V] := 0; {Khởi tạo dãy c}
for i := 1 to n do c[a[i].key] := c[a[i].key] + 1;
```

Như vậy, sau khi sắp xếp:

- Các phần tử có khoá bằng 0 đứng trong đoạn từ vị trí 1 tới vị trí $c[0]$.
- Các phần tử có khoá bằng 1 đứng trong đoạn từ vị trí $c[0] + 1$ tới vị trí $c[0] + c[1]$.
- Các phần tử có khoá bằng 2 đứng trong đoạn từ vị trí $c[0] + c[1] + 1$ tới vị trí $c[0] + c[1] + c[2]$.
- ...
- Các phần tử có khoá bằng V trong đoạn đứng từ vị trí $c[0] + c[1] + \dots + c[V - 1] + 1$ tới vị trí $c[0] + c[1] + \dots + c[V - 1] + c[V]$.
- ...
- Các phần tử có khoá bằng K trong đoạn đứng từ vị trí $c[0] + c[1] + \dots + c[K - 1] + 1$ tới vị trí $c[0] + c[1] + \dots + c[K]$.

Ví dụ: với dãy gồm 8 phần tử có dãy khoá bằng : 2, 0, 2, 5, 1, 2, 0, 3 ta có

$c[0]$	$c[1]$	$c[2]$	$c[3]$	$c[4]$	$c[5]$
2	1	3	1	0	1

Sau khi sắp xếp, các phần tử có khoá bằng 0 sẽ nằm từ vị trí 1 đến vị trí 2, phần tử có khoá bằng 1 nằm ở vị trí 3, các phần tử có khoá bằng 2 nằm từ vị trí 4 đến vị trí 6, phần tử có khoá bằng 3 nằm ở vị trí 7, các phần tử có khoá bằng 5 nằm ở vị trí 8.

Dãy khoá sau khi sắp xếp: 0, 0, 1, 2, 2, 2, 3, 5

Độ phức tạp của thuật toán là: $O(\max(N, K))$

4. Một số ví dụ ứng dụng thuật toán sắp xếp

Ví dụ 1: Giá trị nhỏ thứ k

Cho dãy a_1, a_2, \dots, a_n , các số đôi một khác nhau và số nguyên dương k ($1 \leq k \leq n$). Hãy đưa ra giá trị nhỏ thứ k trong dãy.

Ví dụ dãy gồm 5 phần tử: 5, 7, 1, 3, 4 và $k = 3$ thì giá trị nhỏ thứ k là 4.

Giải

Sắp xếp dãy theo giá trị tăng dần, số đứng thứ k của dãy là giá trị nhỏ thứ k . Nếu $n \leq 5000$ có thể sử dụng thuật toán sắp xếp nổi bọt, nhưng nếu $n > 5000$ thì nên sử dụng thuật toán sắp xếp nhanh.

Ta có thể tìm được giá trị nhỏ thứ k hiệu quả hơn (không cần phải sắp xếp lại cả dãy số) cụ thể:

+ Trong thuật toán sắp xếp nổi bọt ta chỉ cần sắp xếp đến phần tử thứ k , khi đó phần tử thứ k chính là phần tử có khoá nhỏ thứ k .

```

for i:=1 to k do // i chạy đến k
    for j:=n downto i+1 do
        if a[j-1]>a[j] then
            begin
                tmp:=a[j];
                a[j]:=a[j-1];
                a[j-1]:=tmp;
            end;
    
```

+ Trong thuật toán Quick Sort, ta thấy rằng:

- Nếu $k < L \leq H$ thì đoạn từ L đến H không cần sắp xếp vì đoạn này không ảnh hưởng đến vị trí thứ k .

- Nếu $L \leq H < k$ thì đoạn từ L đến H cũng không cần sắp xếp vì đoạn này không ảnh hưởng đến vị trí thứ k .
- Nếu $L \leq k \leq H$ thì ta sẽ xử lí tiếp trong đoạn này.

```

procedure QuickSort(L,H:longint);
var      i,j      :longint;
x,tmp    :longint;
begin
  if (L<=K) and (H>=K) then
    begin
      i:=L;
      j:=H;
      x:=a[(L+H) div 2];
      repeat
        while a[i]<x do inc(i);
        while a[j]>x do dec(j);
        if i<=j then
          begin
            tmp:=a[i];
            a[i]:=a[j];
            a[j]:=tmp;
            inc(i);
            dec(j);
          end;
        until i>j;
        if L<j then QuickSort(L,j);
        if i<H then QuickSort(i,H);
      end;
    end;
end;

```

Sau khi gọi và thực hiện thủ tục QuickSort(1,n) thì số đứng thứ k chính là giá trị nhỏ thứ k .

Chú ý: khi X là số nhỏ thứ $(N \text{ div } 2 + 1)$ của dãy a_1, a_2, \dots, a_n thì hàm

$$F(X) = |a_1 - X| + |a_2 - X| + \dots + |a_n - X|$$

đạt giá trị nhỏ nhất

Ví dụ 2: Tìm kiếm

Cho dãy đã được sắp tăng dần $a_1 \leq a_2 \leq \dots \leq a_n$ và số X . Hãy đưa ra chỉ số i mà $a_i = X$ hoặc đưa ra $i=0$ nếu không có phần tử nào có giá trị bằng X .

Giải

Thuật toán tìm kiếm nhị phân có thể tìm phần tử có giá trị bằng X trên mảng đã được sắp xếp một cách hiệu quả trong thời gian $O(\log n)$. Thuật toán như sau:

Giả sử cần tìm trong đoạn $a[L], a[L + 1], \dots, a[H]$ với giá trị cần tìm kiếm là X , trước hết ta xem xét với giá trị của phần tử nằm giữa dãy, $mid = (L + H) \text{ div } 2$

- Nếu $a[mid] < X$ thì có nghĩa là đoạn từ $a[L]$ tới $a[mid]$ chỉ chứa các phần tử có giá trị $< X$, ta tiếp hành tìm kiếm tiếp với đoạn từ $a[mid + 1]$ đến $a[H]$
- Nếu $a[mid] > X$ thì có nghĩa là đoạn từ $a[mid]$ tới $A[H]$ chỉ chứa các phần tử có giá trị $> X$, ta tiếp hành tìm kiếm tiếp với đoạn từ $a[L]$ đến $a[mid - 1]$
- Nếu $a[mid] = X$ thì việc tìm kiếm thành công (kết thúc quá trình tìm kiếm).

Quá trình tìm kiếm sẽ thất bại nếu đến một bước nào đó, đoạn tìm kiếm là rỗng ($L > H$)

```
function BinarySearch(X: longint): longint;
var L, H, mid: longint;
begin
  L := 1; H := n;
  while L ≤ H do
    begin
      mid := (L + H) div 2;
      if a[mid] = X then exit(mid);

      if a[mid] < X then L := mid + 1
      else H := mid - 1;
    end;
  exit(0);
end;
```

Ví dụ 3: *Thống kê*

Cho dãy a_1, a_2, \dots, a_n . Hãy đếm số lượng giá trị khác nhau có trong dãy và đưa ra số lần lặp của giá trị xuất hiện nhiều nhất.

Ví dụ: dãy gồm 8 số: 6, 7, 1, 7, 4, 6, 6, 8 thì dãy có 5 giá trị khác nhau và số lần lặp của giá trị xuất hiện nhiều nhất trong dãy là 3.

Giải

Các công việc trên sẽ được thực hiện đơn giản nếu mảng đã được sắp xếp, khi đó các phần tử có giá trị bằng nhau sẽ đứng cạnh nhau (liên tiếp nhau).

```
{ Hàm countValue trả về số giá trị khác nhau trong mảng a có n phần tử đã sắp xếp}
function countValue(a : TArray;n : longint):longint;
var i,count :longint;
begin
    count:=1;
    for i:=2 to n do
        if a[i-1]<>a[i] then inc(count);
    countValue := count;
end;

{ Hàm highestFrequency trả về số lần lặp của giá trị xuất hiện nhiều nhất
trong mảng a có n phần tử đã sắp xếp}
function highestFrequency(a:TArray; n:longint):longint;
var i,count,rslt      :longint;
begin
    rslt:=1;
    count:=1;
    for i:=2 to n do begin
        if a[i] <> a[i-1] then count:=1
        else inc(count);
        if count>rslt then rslt:=count;
    end;
    highestFrequency := rslt;
end;
```

Ví dụ 4. Xét dãy F gồm n ($2 < n \leq 10^6$) số nguyên $F = (f_1, f_2, \dots, f_n)$ định nghĩa như sau:

$$f_i = \begin{cases} 1, & \text{nếu } 1 \leq i \leq 2 \\ (f_{i-1} + f_{i-2}) \bmod 128, & \text{nếu } 2 < i \leq n \end{cases}$$

Hãy cho biết nếu sắp xếp dãy F theo thứ tự không giảm thì số thứ k ($k \leq n$) có giá trị là bao nhiêu?

Giải

Ta nhận thấy f_i có giá trị nguyên và $0 \leq f_i \leq 127$, ta sẽ sử dụng thuật toán đếm phân phối như sau:

- Xây dựng dãy F và dãy $c[0], c[1], \dots, c[127]$, trong đó $c[V]$ là số lần xuất hiện giá trị V trong dãy F .
- Giá trị thứ k của dãy F sau khi sắp xếp là giá trị P nhỏ nhất thoả mãn $c[0] + c[1] + \dots + c[P] \geq k$

Chú ý: Sử dụng $a \text{ and } (2^m - 1)$ thay cho $a \bmod (2^m)$, chương trình sẽ chạy nhanh hơn.

```

const maxValue           =128 - 1;
var      fi_2, fi_1, fi    :longint;
        i, v, n, k, cv, P :longint;
        c                 :array[0..maxValue]of longint;

BEGIN
  write('Nhập n, k:');readln(n,k);
  fi_1:=1; fi_2:=1;
  fillchar(c,sizeof(c),0);
  c[fi_1]:=c[fi_1]+1;
  c[fi_2]:=c[fi_2]+1;
  for i:=3 to n do begin
    fi:=(fi_1+fi_2) and maxValue;
    {write(fi:4);}
    c[fi]:=c[fi]+1;
    fi_2:=fi_1;
    fi_1:=fi;
  end;
  cv:=0;
  for v:=0 to maxValue do begin
    cv:=cv+c[v];
    if cv >= k then begin
      P:=v;
    end;
  end;
end.

```

```

        break;
    end;
end;
writeln(P)
END.
```

Ví dụ 5. Cho dãy gồm N ($N \leq 30000$) số tự nhiên không vượt quá 10^9 , tìm số tự nhiên nhỏ nhất không xuất hiện trong dãy.

Dữ liệu vào trong file SN.INP có dạng:

- Dòng đầu là số nguyên N
- Dòng thứ hai gồm N số

Kết quả ra file SN.OUT có dạng: số tự nhiên nhỏ nhất không xuất hiện trong dãy.

SN.INP	SN.OUT
5	2
5 0 3 1 4	

Giải

Ta có nhận xét sau: số tự nhiên nhỏ nhất không xuất hiện trong dãy sẽ nằm trong đoạn $[0, n]$. Do đó, ta sử dụng mảng $c:array[0..30000]of longint$; với $c[x]$ là số lần xuất hiện của x trong dãy, nếu $c[x]=0$ tức là x không xuất hiện trong dãy.

```

const      Limit      =30000;
           fi         ='SN.INP';
           fo         ='SN.OUT';
var       c          :array[0..Limit]of longint;
          n          :longint;
          i,x       :longint;
          f          :text;

BEGIN
  fillchar(c,sizeof(c),0);
  assign(f,fi); reset(f);
  readln(f,n);
  for i:=1 to n do begin
    read(f,x);
    if x<=n then inc(c[x]);
  end;
```

```

close(f);
for i:=0 to n do
  if c[i]=0 then begin
    x:=i;
    break;
  end;
  assign(f,fo); rewrite(f);
  write(f,x);
  close(f);
END.

```

Ví dụ 6. Cho xâu s (độ dài không vượt quá 10^6) chỉ gồm 2 kí tự 'A' và 'B'. Đếm số cách chọn cặp chỉ số (i,j) mà xâu con liên tiếp từ kí tự thứ i đến kí tự thứ j của xâu s có số lượng kí tự 'A' bằng số lượng kí tự 'B'.

Dữ liệu vào trong file “AB.INP” có dạng: gồm một dòng duy nhất chứa xâu s
Kết quả ra file “AB.OUT” có dạng: gồm một dòng duy nhất chứa một số là kết quả bài toán.

AB.INP	AB.OUT
ABAB	4

Giải

```

const MAX =1000000;
      fi = 'AB.INP';
      fo = 'AB.OUT';
var   s :ansistring;
      c :array[-MAX..MAX] of longint;
      f :text;
      i, sum :longint;
      count :int64;

BEGIN
  assign(f,fi); reset(f);
  read(f,s);
  close(f);
  fillchar(c,sizeof(c),0);
  c[0]:=1;
  sum:=0;

```

```

count:=0;
for i:=1 to length(s) do begin
  if s[i]='A' then sum:=sum - 1
  else sum:=sum + 1;
  count:=count + c[sum];
  inc(c[sum]);
end;
assign(f,fo); rewrite(f);
write(f,count);
close(f);
END.

```

Bài tập

3.1. Cho một danh sách n học sinh ($1 \leq n \leq 200$), mỗi học sinh có thông tin sau:

- Họ và tên: Là một xâu kí tự độ dài không quá 30 (các từ cách nhau một dấu cách)
- Điểm: Là một số thực

A) Đưa ra danh sách họ và tên đã sắp xếp theo thứ tự abc (ưu tiên tên, họ, điểm)

B) Có bao nhiêu tên khác nhau trong danh sách, liệt kê các tên đó.

C) Chọn những học sinh có thứ hạng 1, 2, 3 điểm cao nhất trong danh sách để trao học bổng, hãy cho biết tên những học sinh đó.

Ví dụ

Dữ liệu vào	Kết quả câu A	Kết quả câu B	Kết quả câu C
6 Vu Anh Quan 8.9 Nguyen Van Chung 8.7 Hoang Trong Quynh 8.5	Nguyen Van Chung Cong Hoang Dinh Quang Hoang Dinh Quang Huy Quan Quynh Vu Anh Quan Hoang Trong	5 Chung Hoang Huy Quan Quynh	Vu Anh Quan Dinh Quang Huy Dinh Quang Hoang Nguyen Van Chung

Dinh Hoang 8.7	Quang Huy 8.8	Quynh Cong Hoang 8.0		
----------------------	---------------------	----------------------------	--	--

3.2. Cho dãy số gồm N số nguyên $a_1 \leq a_2 \leq \dots \leq a_N$

A) Đưa ra thuật toán có độ phức tạp $O(N \log N)$ để tìm 2 chỉ số $i < j$ mà $a_i + a_j = 0$.

B) Đưa ra thuật toán có độ phức tạp $O(N^2 \log N)$ để tìm 3 chỉ số $i < j < k$ mà $a_i + a_j + a_k = 0$.

C) Đưa ra thuật toán có độ phức tạp $O(N)$ để tìm 2 chỉ số $i < j$ mà $a_i + a_j = 0$.

D) Đưa ra thuật toán có độ phức tạp $O(N^2)$ để tìm 3 chỉ số $i < j < k$ mà $a_i + a_j + a_k = 0$.

3.3. Cho một xâu s (độ dài không quá 200) chỉ gồm các kí tự 'a' đến 'z', đếm số lượng xâu con liên tiếp khác nhau nhận được từ xâu s.

Ví dụ: s='abab', ta có các xâu con liên tiếp khác nhau là:

'a', 'b', 'ab', 'ba', 'aba', 'bab', 'abab', số lượng xâu con liên tiếp khác nhau là 7.

3.4. Viết liên tiếp các số tự nhiên từ 1 đến N ta được một số nguyên M . Ví dụ $N=15$ ta có $M=123456789101112131415$. Hãy tìm cách xoá đi K chữ số của số M để nhận được số M' là lớn nhất.

3.5. Xét tập $F(N)$ tất cả các số hữu tỷ trong đoạn $[0,1]$ với mẫu số không vượt quá N ($1 < N \leq 100$).

Ví dụ tập $F(5)$: $0/1 \ 1/5 \ 1/4 \ 1/3 \ 2/5 \ 1/2 \ 3/5 \ 2/3 \ 3/4 \ 4/5 \ 1/1$

Sắp xếp các phân số trong tập $F(N)$ theo thứ tự tăng dần, đưa ra phân số thứ K .

3.6. Cho xâu s (độ dài không vượt quá 10^6) chỉ gồm các kí tự 'a' đến 'z',

A) Có bao nhiêu loại kí tự xuất hiện trong s

B) Đưa ra một kí tự xuất hiện nhiều nhất trong xâu s và số lần xuất hiện của kí tự đó.

3.7. Cho 2 dãy $a_1 \leq a_2 \leq \dots \leq a_n$ và $b_1 \leq b_2 \leq \dots \leq b_m$, hãy đưa ra thuật toán có độ phức tạp $O(n + m)$ để có dãy $c_1 \leq c_2 \leq \dots \leq c_{n+m}$ là dãy trộn của hai dãy trên.

3.8. Cho dãy số gồm n ($n \leq 10000$) số nguyên a_1, a_2, \dots, a_n ($|a_i| \leq 10^9$), tìm số nguyên X bất kì để $S = |a_1 - X| + |a_2 - X| + \dots + |a_n - X|$ đạt giá trị nhỏ nhất, có bao nhiêu giá trị nguyên khác nhau thoả mãn.

Ví dụ 1: dãy gồm 5 số 3, 1, 5, 4, 5, ta có duy nhất một giá trị $X = 4$ để S đạt giá trị nhỏ nhất bằng 6.

Ví dụ 2: dãy gồm 6 số 3, 1, 7, 2, 5, 7 ta có ba giá trị nguyên của X là 3, 4, 5 để S đạt giá trị nhỏ nhất bằng 13.

3.9. Cho N ($N \leq 10000$) điểm trên mặt phẳng Oxy, điểm thứ i có tọa độ là (x_i, y_i) . Ta định nghĩa khoảng cách giữa 2 điểm $P(x_P, y_P)$ và $Q(x_Q, y_Q)$ bằng $|x_P - x_Q| + |y_P - y_Q|$. Hãy tìm điểm A có tọa độ nguyên mà tổng khoảng cách (theo cách định nghĩa trên) từ A tới N điểm đã cho là nhỏ nhất ($|x_i|, |y_i|$ nguyên không vượt quá 10^9)

3.10. Cho N ($N \leq 10000$) đoạn thẳng trên trực số với các điểm đầu x_i và độ dài d_i ($|x_i|, d_i$ là những số nguyên và không vượt quá 10^9). Tính tổng độ dài trên trực số bị phủ bởi N đoạn trên.

Ví dụ: có 3 đoạn $x_1 = -5, d_1 = 10; x_2 = 0, d_2 = 6; x_3 = -100, d_3 = 10$ thì tổng độ dài trên trực số bị phủ bởi 3 đoạn trên là: 21

3.11. Cho N ($N \leq 300$) điểm trên mặt phẳng Oxy, điểm thứ i có tọa độ là (x_i, y_i) . Hãy đếm số cách chọn 4 điểm trong N điểm trên mà 4 điểm đó tạo thành 4 đỉnh của một hình chữ nhật. ($|x_i|, |y_i|$ nguyên không vượt quá 1000)

Ví dụ: có 5 điểm $(0, 0), (0, 1), (1, 0), (-1, 0), (0, -1)$ có duy nhất 1 cách chọn 4 điểm mà 4 điểm đó tạo thành 4 đỉnh của một hình chữ nhật.

3.12. Cho N ($N \leq 10000$) đoạn số nguyên $[a_i, b_i]$, hãy tìm một số mà số đó thuộc nhiều đoạn số nguyên nhất.

Ví dụ: có 5 đoạn $[0, 10], [2, 3], [4, 7], [3, 5], [5, 8]$, ta chọn số 5 thuộc 4 đoạn $[0, 10], [4, 7], [3, 5], [5, 8]$.

3.13. Cho dãy gồm N ($N \leq 10000$) số a_1, a_2, \dots, a_N . Hãy tìm dãy con liên tiếp dài nhất có tổng bằng 0. ($|a_i| \leq 10^9$)

Ví dụ: dãy gồm 5 số 2, 1, -2, 3, -2 thì dãy con liên tiếp dài nhất có tổng bằng 0 là: 1, -2, 3, -2

3.14. ESEQ

Cho dãy số nguyên A gồm N phần tử A_1, A_2, \dots, A_N , tìm số cặp chỉ số i, j thoả mãn:

$$\sum_{p=1}^i A_p = \sum_{q=j}^N A_q \text{ với } 1 \leq i < j \leq N$$

Dữ liệu vào trong file “ESEQ.INP” có dạng:

- Dòng đầu là số nguyên dương N ($2 \leq N \leq 10^5$)
- Dòng tiếp theo chứa N số nguyên A_1, A_2, \dots, A_N ($|A_i| < 10^9$), các số cách nhau một dấu cách.

Kết quả ra file “ESEQ.OUT” có dạng: gồm một số là số cặp tìm được.

ESEQ.INP	ESEQ.OUT
3	3
1 0 1	

3.15. GHÉP SỐ

Cho n số nguyên dương a_1, a_2, \dots, a_n ($1 < n \leq 100$), mỗi số không vượt quá 10^9 . Từ các số này người ta tạo ra một số nguyên mới bằng cách ghép tất cả các số đã cho, tức là viết liên tiếp các số đã cho với nhau. Ví dụ, với $n = 4$ và các số 123, 124, 56, 90 ta có thể tạo ra các số mới sau: 1231245690, 1241235690, 5612312490, 9012312456, 9056124123,... Có thể dễ dàng thấy rằng, với $n = 4$, ta có thể tạo ra 24 số mới. Trong trường hợp này, số lớn nhất có thể tạo ra là 9056124123.

Yêu cầu: Cho n và các số a_1, a_2, \dots, a_n . Hãy xác định số lớn nhất có thể tạo ra khi ghép các số đã cho thành một số mới.

Dữ liệu vào từ file văn bản NUMJOIN.INP có dạng:

- Dòng thứ nhất chứa số nguyên n ,
- Dòng thứ 2 chứa n số nguyên $a_1 \ a_2 \ \dots \ a_n$.

Kết quả ra file văn bản NUMJOIN.OUT gồm một dòng là số lớn nhất có thể tạo ra khi ghép các số đã cho thành một số mới.

3.16. GIÁ TRỊ NHỎ NHẤT

Cho bảng số A gồm MxN ô, mỗi ô chứa một số nguyên không âm (A_{ij}) có giá trị không vượt quá 10^9 . Xét hàng i và hàng j của bảng, ta cần xác định X_{ij} nguyên để:

$$S_{ij} = \left(\sum_{k=1}^N |A_{ik} - X_{ij}| \right) + \left(\sum_{k=1}^N |A_{jk} - X_{ij}| \right) \text{đạt giá trị nhỏ nhất.}$$

Tính $W = \sum_{i=1}^{M-1} \sum_{j=i+1}^M S_{ij}$

Dữ liệu vào trong file “WMT.INP” có dạng:

- Dòng đầu là 2 số nguyên dương M, N ($1 < M, N < 1001$)
- M dòng sau, mỗi dòng N số

Kết quả ra file “WMT.OUT” có dạng: gồm một số W

WMT.INP	WMT.OUT
2 3	5
2 3 1	
2 3 4	

3.17. DECIPHERING THE MAYAN WRITING (IOI 2006)

Công việc giải mã chữ viết của người MAIA là khó khăn hơn người ta tưởng nhiều. Trải qua hơn 200 năm mà người ta vẫn hiểu rất ít về các chữ viết này. Chỉ trong 3 thập niên gần đây do công nghệ phát triển việc giải mã này mới có nhiều tiến bộ.

Chữ viết Maia dựa trên các kí hiệu nhỏ gọi là nét vẽ, mỗi nét vẽ tương ứng với một âm giọng nói. Mỗi từ trong chữ viết Maia sẽ bao gồm một tập hợp các nét vẽ như vậy kết hợp lại với nhiều kiểu dáng khác nhau. Mỗi nét vẽ có thể hiểu là một kí tự ta hiểu ngày nay.

Một trong những vấn đề lớn khi giải mã chữ Maia là thứ tự đọc các nét vẽ. Do người Maia trình bày các nét vẽ này không theo thứ tự phát âm, mà theo cách thể hiện của chúng. Do vậy nhiều khi đã biết hết các nét vẽ của một từ rồi nhưng vẫn không thể tìm ra được chính xác cách ghi và đọc của từ này.

Các nhà khảo cổ đang đi tìm kiếm một từ đặc biệt W. Họ đã biết rõ tất cả các nét vẽ của từ này nhưng vẫn chưa biết các cách viết ra của từ này. Vì họ biết có các thí sinh IOI'06 sẽ đến nên muốn sự trợ giúp của các sinh viên này. Họ sẽ đưa ra toàn bộ g nét vẽ của từ W và dãy S tất cả các nét vẽ có

trong hang đá cổ. Bạn hãy giúp các nhà khảo cổ tính xem có bao nhiêu khả năng xuất hiện từ W trong hang đá.

Yêu cầu: Hãy viết chương trình, cho trước các kí tự của từ W và dãy S các nét vẽ trong hang đá, tính tổng số khả năng xuất hiện của từ W trong dãy S, nghĩa là số lần xuất hiện một hoán vị các kí tự của dãy g kí tự trong S.

Các ràng buộc

$1 \leq g \leq 3\ 000$, số nét vẽ trong W

$g \leq |S| \leq 3\ 000\ 000$, $|S|$ là số các nét vẽ của dãy S

Dữ liệu vào:

- Dòng 1: chứa 2 số g và $|S|$ cách nhau bởi dấu cách.
- Dòng 2: chứa g kí tự liền nhau là các nét vẽ của từ W. Các kí tự hợp lệ là 'a'-'z' và 'A'-'Z'. Các chữ in hoa và in thường là khác nhau.
- Dòng 3: Chứa $|S|$ kí tự là dãy các nét vẽ tìm thấy trong hang. Các kí tự hợp lệ là 'a'-'z' và 'A'-'Z'. Các chữ in hoa và in thường là khác nhau.

Kết quả ra:

Chứa đúng 1 số là khả năng xuất hiện của từ W trong dãy S.

Dữ liệu vào	Kết quả ra
4 11 cAda AbrAcadAbRa	2

3.18. TRÒ CHƠI VỚI DÃY SỐ (Học sinh giỏi quốc gia, 2007-2008)

Hai bạn học sinh trong lúc nhàn rỗi nghĩ ra trò chơi sau đây. Mỗi bạn chọn trước một dãy số gồm n số nguyên. Giả sử dãy số mà bạn thứ nhất chọn là: b_1, b_2, \dots, b_n

còn dãy số mà bạn thứ hai chọn là: c_1, c_2, \dots, c_n

Mỗi lượt chơi mỗi bạn đưa ra một số hạng trong dãy số của mình. Nếu bạn thứ nhất đưa ra số hạng b_i ($1 \leq i \leq n$), còn bạn thứ hai đưa ra số hạng c_j ($1 \leq j \leq n$) thì giá của lượt chơi đó sẽ là $|b_i + c_j|$.

Ví dụ: Giả sử dãy số bạn thứ nhất chọn là 1, -2; còn dãy số mà bạn thứ hai chọn là 2, 3. Khi đó các khả năng có thể của một lượt chơi là (1, 2), (1, 3), (-2, 2), (-2, 3). Như vậy, giá nhỏ nhất của một lượt chơi trong số các lượt chơi có thể là 0 tương ứng với giá của lượt chơi (-2, 2).

Yêu cầu: Hãy xác định giá nhỏ nhất của một lượt chơi trong số các lượt chơi có thể.

Dữ liệu vào:

Dòng đầu tiên chứa số nguyên dương n ($n \leq 10^5$)

Dòng thứ hai chứa dãy số nguyên b_1, b_2, \dots, b_n ($|b_i| \leq 10^9, i = 1, 2, \dots, n$)

Dòng thứ hai chứa dãy số nguyên c_1, c_2, \dots, c_n ($|c_i| \leq 10^9, i = 1, 2, \dots, n$)

Hai số liên tiếp trên một dòng được ghi cách nhau bởi dấu cách.

Kết quả ra:

Ghi ra giá nhỏ nhất tìm được.

Dữ liệu vào	Kết quả ra
2	0
1 -2	
2 3	

3.19. DÃY SỐ (Học sinh giỏi, Hà Nội 2008-2009)

Cho dãy số nguyên a_1, a_2, \dots, a_n . Số a_p ($1 \leq p \leq n$) được gọi là một số trung bình cộng trong dãy nếu tồn tại 3 chỉ số i, j, k ($1 \leq i, j, k \leq n$) đôi một khác nhau, sao cho $a_p = (a_i + a_j + a_k)/3$

Yêu cầu: Cho n và dãy số a_1, a_2, \dots, a_n . Hãy tìm số lượng các số trung bình cộng trong dãy.

Dữ liệu vào:

- Dòng đầu ghi số nguyên dương n ($3 \leq n \leq 1000$)
- Dòng thứ hai chứa n số nguyên a_i ($|a_i| < 10^8$)

Kết quả ra:

Số lượng các số trung bình cộng trong dãy.

Dữ liệu vào	Kết quả ra
5	2
4 3 6 3 5	

3.20. ĐỆM SỐ TAM GIÁC (Tin học trẻ, bảng B, năm 2009)

Cho ba số nguyên dương a, b, m và n ($m, n \leq 10000$) đoạn thẳng đánh số từ 1 tới n . Đoạn thẳng thứ i có độ dài d_i ($\forall i: 1 \leq i \leq n$), ở đây các độ dài (d_1, d_2, \dots, d_n) được cho như sau:

$$d_i = \begin{cases} b, & \text{nếu } i = 1 \\ (a \times d_{i-1} + b) \bmod m + 1, & \text{nếu } 1 < i \leq n \end{cases} \quad (*)$$

Hãy cho biết có bao nhiêu tam giác khác nhau có thể được tạo ra bằng cách lấy đúng ba đoạn trong số n đoạn thẳng đã cho làm ba cạnh (hai tam giác bằng nhau nếu chúng có ba cặp cạnh tương ứng bằng nhau, nếu không chúng được coi là khác nhau).

Ví dụ với $a = 6; b = 3; m = 4; n = 5$. Ta có 5 đoạn thẳng với độ dài của chúng tính theo công thức (*) là $(3, 2, 4, 4, 4)$. Với 5 đoạn thẳng này có thể tạo ra được 4 tam giác với độ dài các cạnh được chỉ ra như sau:

Tam giác 1: $(2, 3, 4)$

Tam giác 2: $(2, 4, 4)$

Tam giác 3: $(3, 4, 4)$

Tam giác 4: $(4, 4, 4)$