KĨ THUẬT 2 CON TRỎ

Nguồn: https://www.geeksforgeeks.org/two-pointers-technique

Người dịch: Nguyễn Như Thắng – GV Tin học trường THPT Chuyên tỉnh Lào Cai

I. Giới thiệu kĩ thuật 2 con trỏ.

Hai con trỏ thực sự là một kĩ thuật dễ dàng và hiệu quả dùng trong tìm kiếm các cặp thỏa mãn điều kiện nào đó trong mảng đã được sắp xếp.

Ví dụ: Cho một mảng đã sắp xếp A (được sắp xếp theo thứ tự tăng dần), có N số nguyên, hãy tìm xem tồn tại bất kì một cặp (A[i], A[j]) mà có tổng bằng X.

Với một mảng sắp xếp A (được sắp xếp theo thứ tự tăng dần), có N số nguyên, tìm xem có tồn tại bất kỳ cặp phần tử nào (A [i], A [j]) sao cho tổng của chúng bằng X

Hãy quan sát giải thuật sơ đẳng sau:

// Giải thuật sơ đẳng tìm cặp trong mảng A[0..N-1] mà có tổng bằng X cho trước

```
bool isPairSum(A[], N, X) {
  for (i = 0; i < N; i++) {
     for (j = 0; j < N; j++) {
        if (A[i] + A[j] == X)
           return true;                  // Tồn tại cặp thỏa mãn
        if (A[i] + A[j] > X)
           break;                        // Không tồn tại do mảng đã sắp xếp
     }
  }
    return false;   // Không tìm ra cặp mà có tổng cho trước.

}
```

Độ phức tạp của lời giải trên là $O(N^2)$.

Bây giờ chúng ta sẽ xem kĩ thuật 2 con trỏ hoạt động như thế nào. Ta lấy 2 con trỏ, một đặt tại phần tử đầu của dãy, một đặt tại phần tử cuối cùng của dãy. Và tính tổng của 2 phần tử mà 2 con trỏ đang trỏ đến. Nếu tổng của chúng nhỏ hơn X thì chúng ta cần dịch chuyển con trỏ bên bên trái sang bên phải, hoặc nếu tổng lớn hơn thì ta lại dịch chuyển con trỏ bên phải sang trái, để có được tổng gần X hơn. Cứ di chuyển như vậy đến khi nào được tổng bằng X thì dừng.

// Kĩ thuật 2 con trỏ để tìm cặp trong mảng A[0..N-1] có tổng cho trước.

```
bool isPairSum(A[], N, X){
   int i = 0; //con trỏ thứ nhất (con trỏ bên trái)
   int j = N - 1; //con trỏ thứ 2, con trỏ bên phải
    while (i < j) {
      if (A[i] + A[j] == X)   //Nếu tìm thấy
         return true;
          else if (A[i] + A[j] < X)
```

```
        //nếu tổng nhỏ hơn, di chuyển con trỏ bên trái tăng dần
         i++;
      else   j--;    //ngược lại, thì di chuyển con trỏ bên phải giảm dần
   }
   return false;  //không tim thấy
}
```

Minh họa như sau:

A[]={10,20,35,50,75,80} ; X=70

Bước 1: i=0, j=5➔A[i]+A[j]=10+80=90.
Từ A[i]+A[j]>X ➔ Giảm j

Bước 2: i=0; j=4➔A[i]+A[j]=10+75=85
Từ A[i]+A[j]>X➔ Giảm j

Bước 3: i=0; j=3➔A[i]+A[j]=10+50=60
Từ A[i]+A[j]<X➔ Tăng i

Bước 4: i=1; j=3➔A[i]+A[j]=20+50.
Từ A[i]+A[j]=X➔ Tìm thấy cặp phần tử thỏa mãn.

Độ phức tạp của lời giải trên là O(N).

Hoạt động như thế nào?: Thuật toán dựa trên mảng đầu vào đã được sắp xếp. Ta tính tổng của giá trị các đầu mút (tương ứng giá trị nhỏ nhất, lớn nhất) và điều kiện di chuyển cả hai con trỏ. Khi di chuyển con trỏ trái ***i*** với tổng **A[i]+A[j]<X**, ta không hề bỏ qua một cặp nào vì tổng của các cặp trước nó chắc chắn luôn nhỏ hơn **X**. Tương tự với con trỏ phải ***j***.

II. Một số bài tập sử dụng kĩ thuật 2 con trỏ.

2.1. Cho một mảng đã sắp xếp và số X, tìm cặp trong mảng mà có tổng gần X nhất

Ví dụ:

```
Input: arr[] = {10, 22, 28, 29, 30, 40}, X = 54
Output: 22 and 30

Input: arr[] = {1, 3, 4, 7, 10}, X = 15
Output: 4 and 10
```

Một lời giải đơn giản là thử moi cặp và tìm xem cặp nào gần X nhất (giá trị tuyệt đối của của hiệu A[i]+A[j]-X là nhỏ nhất). Sau đó in ra cặp gần nhất. Độ phức tạp của lời giải trên là $O(N^2)$.

Một lời giản có thể tìm ra cặp trong O(N). Ý tưởng tương tự với kĩ thuật 2 con trỏ đã nói trên. Sau đây là chi tiết thuật giải:
1, Khởi tạo biến DIFF là dương vô cùng (DIFF dùng để lưu độ lệch của cặp so với X). Chúng ta cần tìm giá trị nhỏ nhất của DIFF.
2. Khởi tao 2 biến chỉ số l,r trong dãy đã cho.
   a) Khởi tao con trỏ trái: l=0;

b) Khởi tao con trỏ phải: r=n-1

3. Lặp while (l<r)

a) Nếu (a) If abs(arr[l] + arr[r] - sum) < diff thì cập nhật lại giá trị DIFF

b) Else if(arr[l] + arr[r] < sum) then l++

c) Else r--

Sau đây là cài đặt thuật toán trên bằng C++, chương trình C++ đơn giản tìm cặp mà có tổng gần với giá trị X cho trước nhất:

```
#include <iostream>
#include <climits>
#include <cstdlib>
using namespace std;

// Prints the pair with sum closest to x
void printClosest(int arr[], int n, int x)
{
   int res_l, res_r;  // To store indexes of result pair
   int l = 0, r = n-1, diff = INT_MAX; // Khởi tạo con trỏ trái, phải, và khoảng cách đến X
   while (r > l)  {     // Khi vẫn còn phần tử nằm giữa l và r
         if (abs(arr[l] + arr[r] - x) < diff) {       // Kiểm tra cặp đang xét gần X hơn cặp trước
đó
         res_l = l;
         res_r = r;
         diff = abs(arr[l] + arr[r] - x);
      }
       if (arr[l] + arr[r] > x) // Nếu cặp l,r có tổng lớn hơn X thì dịch con trỏ phải sang trái
         r--;
      else // Ngược lại thì dịch con trỏ trái sang phải
         l++;
   }

   cout <<" Cặp gần nhất là: " << arr[res_l] << " và " << arr[res_r];
}
int main()
{
   int arr[] =  {10, 22, 28, 29, 30, 40}, x = 54;
   int n = sizeof(arr)/sizeof(arr[0]);
   printClosest(arr, n, x);
   return 0;
}
```

Output:

```
Cặp gần nhất là 22 và 33
```

Tương tự bài trên là Bài 1: SEQGAME thi HSG Duyên Hải lớp 10 năm 2018. Lời giải sau đã test AC và do người dịch tự code:

```
#include<bits/stdc++.h>
using namespace std;
const int N=1e5;
int x[N],y[N],s,m,n,k;
int main() {
   freopen("SEQGAME.inp","r",stdin);
   freopen("SEQGAME.out","w",stdout);
   scanf("%d%d%d",&m,&n,&k);
   for (int i=0; i<m; i++)
      scanf("%d",&x[i]);
   for (int i=0; i<n; i++)
      scanf("%d",&y[i]);
   sort(x,x+m);
   sort(y,y+n);
   while (k--) {
      scanf("%d",&s);
      int l=0,r=n-1,diff=INT_MAX;
      while (l<=m-1 && r>=0) {
         if (diff==0)
            break;
         if (abs(x[l]+y[r]-s)<diff)
            diff=abs(x[l]+y[r]-s);
         if (x[l]+y[r]>s)
            r--;
         else
            l++;
      }
      printf("%d\n",diff);
   }
}
```

Ngoài kiểu sử dụng 2 con trỏ chạy ngược chiều nhau như các ví dụ trên, chúng ta có thể cho chạy song song (gọi là dạng cửa số trượt – slide window) hoặc chạy theo một quy luật nào đó tùy theo điều kiện, yêu cầu của đề bài.

III. Đánh giá của người dịch:

Chuyên đề này chưa có tài liệu tiếng Việt nào được viết trên các diễn đàn như VNOI, một số trang quốc tế như CodeForce chưa đề cập rõ ràng. Tuy nhiên trang https://www.geeksforgeeks.org lại viết khá rõ, ngoài ra còn có bài viết trên https://tp-iiita.quora.com/The-Two-Pointer-Algorithm cũng khá hay.

Chuyên đề này đã được người dịch dạy cho học sinh đội tuyển HSG lớp 10 ngày 28/4/2018 và đã thu được kết quả: HS cơ bản đã nắm được và làm tốt dạng bài tương đương mức thi HSG Duyên Hải 2018. Và một số bài tập trên trang CodeForce, SPOJ.

Danh sách bài tập để học sinh luyện tập được phân loại theo độ khó

	Problem Name		Online Judge	Contest	Level
1	Complete the	Categories	Codeforces	Codeforces Round #372 (Div. 2)	1

	Word				
2	BerSU Ball	Categories	Codeforces	Codeforces Round #277.5 (Div. 2)	1
3	Kefa and Company	Categories	Codeforces	Codeforces Round #321 (Div. 2)	1
4	Sereja and Dima	Categories	Codeforces	Codeforces Round #223 (Div. 2)	1
5	Number of Ways	Categories	Codeforces	Codeforces Round #266 (Div. 2)	1
6	Roommate Agreement	Categories	SPOJ		1
7	Books	Categories	Codeforces	Codeforces Round #171 (Div. 2)	1
8	Guess a number!	Categories	Codeforces	Codeforces Round #241 (Div. 2)	2
9	Counting Kangaroos is Fun	Categories	Codeforces	Codeforces Round #219 (Div. 1) & Codeforces Round #219 (Div. 2)	2
10	Audition	Categories	SPOJ		2
11	Vasya and String	Categories	Codeforces	Codeforces Round #354 (Div. 2)	2
12	Alice, Bob and Chocolate	Categories	Codeforces	Codeforces Beta Round #6 (Div. 2 Only)	2
13	Queries about less or equal elements	Categories	Codeforces	Educational Codeforces Round 2	2
14	George and Round	Categories	Codeforces	Codeforces Round #227 (Div. 2)	2
15	DZY Loves Sequences	Categories	Codeforces	Codeforces Round #255 (Div. 1) & Codeforces Round #255 (Div. 2)	2
16	Escape from Stones	Categories	Codeforces	Codeforces Round #162 (Div. 1) & Codeforces Round #162 (Div. 2)	2
17	Points on Line	Categories	Codeforces	Codeforces Round #153 (Div. 1) & Codeforces Round #153 (Div. 2)	2
18	Kirill And The Game	Categories	Codeforces	Codeforces Round #430 (Div. 2)	2
19	Approximating a Constant Range	Categories	Codeforces	Codeforces Round #333 (Div. 2)	2
20	Another Problem on Strings	Categories	Codeforces	Codeforces Round #112 (Div. 2)	2

-----HẾT-----