SỞ GIÁO DỤC VÀ ĐÀO TẠO NGHỆ AN

===== ▯ =====

ĐỀ CƯƠNG SÁNG KIẾN KINH NGHIỆM

SỬ DỤNG QUY HOẠCH ĐỘNG ĐỂ NÂNG CAO NĂNG LỰC GIẢI QUYẾT MỘT SỐ VẤN ĐỀ VỀ DÃY CON BẰNG NGÔN NGỮ LẬP TRÌNH C++

THUỘC MÔN: TIN HỌC

THÁNG 3/ 2022

I. PHẦN MỞ ĐẦU

1.1 Lý do chọn đề tài

Trong quá trình giảng dạy phát triển năng lực cho học sinh khá giỏi thường gặp rất nhiều bài toán về dãy con. Đây là dạng bài tập khó thường xuất hiện trong các đề thi học sinh giỏi môn Tin học. Rất nhiều học sinh khi gặp dạng bài tập dạng này thì khó tìm được cách giải tối ưu nên điểm không cao. Nguyên nhân có thể nhiều nhưng trong đó có hai nguyên nhân cơ bản là: chương trình cho kết quả output sai hoặc chương trình cho kết quả output đúng với các bộ input có dữ liệu nhỏ nhưng với những bộ input có dữ liệu lớn thì chương trình chạy quá thời gian quy định là 1giây/1test (mặc dù kết quả output vẫn đúng).

Trên thực tế đã có một số tài liệu đề cập đến các bài tập về dãy con, nhưng các tài liệu này mới chỉ đưa ra thuật toán và chương trình giải một số bài tập cụ thể làm ví dụ minh họa cho một kỹ thuật lập trình nào đó khi nghiên cứu mà chưa khái quát dạng, chưa phân tích sâu cách tư duy, cách lựa chọn và cài đặt chương trình tối ưu. Các chương trình mà một số tài liệu đưa ra rất khó hiểu và phức tạp không phù hợp năng lực học sinh Trường THPT Lê Viết Thuật. Khi nghiên cứu các tài liệu này, không chỉ học sinh mà ngay cả giáo chưa có kinh nghiệm cũng rất khó khăn?

Từ những lý do trên, chúng tôi chọn nghiên cứu đề tài: ***"Sử dụng quy hoạch động để nâng cao năng lực giải quyết một số vấn đề về dãy con bằng ngôn ngữ lập trình C++"***.

1.2. Mục đích nghiên cứu

Với mong muốn sử dụng quy hoạch động nâng cao năng lực giải quyết một số vấn đề về dãy con và hiểu biết sâu sắc hơn cách giải các bài tập dạng này, chúng tôi đã dày công nghiên cứu, phân dạng các bài tập dãy con, trăn trở để tìm ra nhiều cách làm khác nhau, đánh giá độ phức tạp, đo thời gian thực hiện chương trình, để so sánh tìm ra chương trình tối ưu nhất và dễ hiểu nhất trong các chương trình đã đưa ra. Từ đó nâng cao chất lượng bồi dưỡng học sinh giỏi môn Tin học.

1.3. Đối tượng nghiên cứu

Sáng kiến kinh nghiệm có đối tượng nghiên cứu là

- Một số bài toán về dãy con liên tiếp
- Một số bài toán về dãy con không liên tiếp

Được nghiên cứu ở nhiều cách làm, xét trên nhiều phương diện *(trong đó nhấn mạnh phương pháp quy hoạch động)* như: độ phức tạp, kết quả output, thời gian thực hiện chương trình.

1.4. Phương pháp nghiên cứu

Để trình bày sáng kiến kinh nghiệm này, chúng tôi đã sử dụng phối kết hợp nhiều phương pháp như: nghiên cứu tài liệu, thuyết trình, quan sát, điều tra cơ bản, thực nghiệm so sánh, phân tích kết quả thực nghiệm, … phù hợp với môn học thuộc lĩnh vực Tin học, Toán học.

Trong từng phần chúng tôi sắp xếp và trình bày các bài tập từ dễ đến khó, đồng thời thông qua từng bài tập chúng tôi cố gắng phân tích nhằm đưa ra một số định hướng lời giải bài toán để rèn luyện cho học sinh có kinh nghiệm, kỹ năng vận dụng một số bài toán tương tự nhau, hướng tới sự phát triển năng lực cho học sinh.

II . NỘI DUNG NGHIÊN CỨU

2.1. Cơ sở lý luận

Nếu học sinh biết vận dụng phương pháp quy hoạch động vào việc giải quyết các bài toán về dãy con nói riêng và các bài tập lập trình nói chung thì chất lượng học sinh giỏi sẽ được nâng cao.

2.2. Thực trạng trước khi nghiên cứu

Các năm học trước chúng tôi cũng đã trực tiếp giảng dạy cho đội tuyển học sinh giỏi các cấp về chuyên đề dãy con, tuy nhiên việc dạy chuyên đề này chủ yếu dựa trên những kiến thức cơ bản của sách giáo khoa, tài liệu tham khảo chưa chú trọng nhiều đến việc nghiên cứu kiến thức Toán học để vận dụng giải quyết các bài toán.

Chính vì vậy nên các em chủ yếu chỉ biết giải quyết các bài toán mà thầy, cô đã dạy mà không hiểu bản chất thật của bài toán, khi gặp các bài toán cùng dạng nhưng có khác chút ít thì gặp phải rất nhiều khó khăn.

Kết quả của thực trạng: Trên cơ sở nhiều năm được phân công dạy khối lớp 11, trường THPT Lê Viết Thuật, chúng tôi đã lưu lại kết quả học tập và sự tiến bộ của học sinh ở mỗi năm học ở một số lớp để có sự đối chiếu và rút kinh nghiệm.

- Bảng số liệu kết quả đạt được khi chưa thực hiện đề tài: năm học 2019 - 2020

STT	Lớp	Sĩ số	Giỏi	Khá	Trung bình	Không đạt yêu cầu
1	$11T_1$	35	3%	29%	57%	11%
2	$11A_1$	40		13%	63%	25%
3	$11A_2$	38		6%	50%	44%

- Khi thực nghiệm qua các đối tượng học sinh đã nêu trên, đa số các em còn lúng túng trước những bài toán lập trình cơ bản. Phần lớn các em còn chưa hứng thú với các bài toán lập trình đặc biệt là với ngôn ngữ pascal.

Vì vậy trong quá trình giảng dạy chúng tôi đúc rút ra một số kinh nghiệm để giúp các học sinh tiếp cận nội dung này dễ dàng hơn, tạo nhiều đam mê cho học sinh. Để rèn năng lực và kỹ năng lập trình cho học sinh khá, giỏi môn Tin học, có rất nhiều cách mà giáo viên có thể áp dụng đối với các đối tượng học sinh khác nhau. Thông thường khi cho một bài toán tin học có dạng tương tự hoặc dạng mở rộng từ một bài toán cơ bản nào đó trong sách giáo khoa, hoặc một bài toán cơ bản nào đó mà các em biết thì các em có thể xây dựng và có hứng thú để xây dựng thuật toán cho bài toán đặt ra. Vì vậy giáo viên có thể chọn các bài tập cơ bản từ đó mở rộng và phát triển để rèn luyện kỹ năng lập trình cho học sinh. Dĩ nhiên cách làm này không mới với giáo viên nhưng cách chọn các bài toán cơ bản như thế nào để học sinh có thể vận dụng và gây được hưng thú cho học sinh đó lại là điều đáng quan tâm. Và chúng tôi đã hoàn toàn thay thế ngôn ngữ lập trình pascal bằng ngôn ngứ lập trình C++ và ngôn ngữ lập trình Python để tạo thuận lợi cho các em trong việc cài đặt chương trình.

2.3. Các biện pháp sử dụng để giải quyết vấn đề

2.3.1. Cơ sở lý thuyết

Khi nào thì chúng ta cần đến quy hoạch động? Đó là một câu hỏi rất khó trả lời. Không có một công thức nào cho các bài toán như vậy.

Tuy nhiên, có một số tính chất của bài toán mà bạn có thể nghĩ đến quy hoạch động. Dưới đây là hai tính chất nổi bật nhất trong số chúng:

Bài toán có các bài toán con gối nhau

Bài toán có cấu trúc con tối ưu

Thường thì một bài toán có đủ cả hai tính chất này, chúng ta có thể dùng quy hoạch động được. Một câu hỏi rất thú vị là không dùng quy hoạch động có được không? Câu trả lời là có, nhưng nếu bạn đi thi code thì kết quả không cao.

a. Dãy con liên tiếp

Dãy con liên tiếp là dãy gồm các phần tử liên tiếp thuộc một dãy cho trước.

Ví dụ: Cho dãy A gồm 4 số nguyên {5,3,4,-4}. Dãy số {4}; {3,4}; {5,3,4}; {5,3,4,-4}; … được gọi là các dãy con liên tiếp của dãy A.

b. Dãy con không liên tiếp

Dãy con có thể chọn không liên tiếp là dãy thu được sau khi xóa một số phần tử (có thể không xóa phần tử nào) của một dãy cho trước và giữ nguyên thứ tự các phần tử còn lại trong dãy.

Ví dụ: Cho dãy B gồm 6 số nguyên {3,5,-8,7,24,4}. Dãy số {3}; {3,5}; {-8,7}; {7,24,4}; {3,1,2,-6,9}; … được gọi là các dãy con có thể chọn không liên tiếp của dãy A.

c. Mô hình về dãy con

Cho dãy a1,a2,..an. Hãy tìm một dãy con tăng có nhiều phần tử nhất của dãy.

Đặc trưng:

i) Các phần tử trong dãy kết quả chỉ xuất hiện 1 lần. Vì vậy phương pháp làm là ta sẽ dùng vòng For duyệt qua các phần tử trong dãy.

ii) Thứ tự của các phần tử được chọn phải được giữ nguyên so với dãy ban đầu. Đặc trưng này có thể mất đi trong một số bài toán khác tùy vào yêu cầu cụ thể.

2.3.2. Độ phức tạp của thuật toán

Giả sử ta có hai thuật toán P1 và P2 với thời gian thực hiện tương ứng là $T_1(n) = 100n^2$ (với tỷ suất tăng là n^2) và $T_2(n) = 5n^3$ (với tỷ suất tăng là n^3). Khi $n > 20$ thì $T_1 < T_2$. Sở dĩ như vậy là do tỷ suất tăng của T_1 nhỏ hơn tỷ suất tăng của T_2. Như vậy một cách hợp lý là ta xét tỷ suất tăng của hàm thời gian thực hiện chương trình thay vì xét chính bản thân thời gian thực hiện. Cho một hàm T(n), T(n) gọi là có độ phức tạp f(n) nếu tồn tại các hằng C, N_0 sao cho $T(n) \leq Cf(n)$ với mọi $n \geq N_0$ (tức là T(n) có tỷ suất tăng là f(n)) và kí hiệu T(n) là O(f(n)) (đọc là "ô của f(n)").

Các hàm thể hiện độ phức tạp có các dạng thường gặp sau: $\log_2 n$, n, $n\log_2 n$, n^2, n^3, 2^n, n!, n^n. Trong cách viết, ta thường dùng logn thay thế cho $\log_2 n$ cho gọn.

Khi ta nói đến độ phức tạp của thuật toán là ta nói đến hiệu quả thời gian thực hiện chương trình nên có thể xem việc xác định thời gian thực hiện chương trình chính là xác định độ phức tạp của thuật toán.

2.3.3. Phương pháp lựa chọn và cài đặt chương trình tối ưu khi giải một số dạng bài tập về dãy con

Đối với mỗi dạng bài tập về dãy con chúng tôi đưa ra một bài toán cơ bản, từ mỗi bài toán cơ bản, trình bày từ 1 hoặc 2 cách giải (cả cách làm của học sinh và cách làm của giáo viên định hướng cho học sinh làm). Với phương châm " mưa dầm thấm lâu" chúng tôi không hướng dẫn học sinh cách làm tối ưu ngay mà khi phát vấn một dạng bài tập mới mà chúng tôi yêu cầu học sinh làm theo các trình tự sau:

Bước 1: Xác định bài toán

Bước 2: Suy nghĩ tìm ra thuật toán, viết chương trình, tính độ phức tạp (Có thể nhiều cách).

Bước 3: Trao đổi cách làm của mình với bạn để tìm cái hay cái dở.

Bước 4: Sử dụng phần mềm *Themis-chấm bài tự động* để chấm cách làm của mình (với 10 bộ test hoặc nhiều hơn mà giáo viên đã xây dựng sẵn, mỗi bộ test cấu hình là 1 điểm, thời gian chạy không quá 1 giây).

Bước 5: Nhận xét sự tối ưu của thuật toán.

Bước 6: Giáo viên định hướng cách làm tối ưu hơn (nếu có).

Bước 7: Sử dụng phần mềm Themis để chấm tất cả các cách đã viết chương trình.

Bước 8: Dựa vào kết quả, lựa chọn chương trình có độ phức tạp nhỏ nhất, thời gian thực hiện mỗi test nhỏ nhất và chương trình ngắn gọn dễ hiểu nhất.

Bước 9: Lập trình giải các bài tập tương tương với cách đã lựa chọn.

2.4 Các bài toán về dãy con liên tiếp

Các dãy con không chung nhau bất kỳ phần tử nào của dãy ban đầu nghĩa là những phần tử của dãy ban đầu đã thuộc dãy con thỏa mãn này thì không thuộc các dãy con thỏa mãn khác.

Ví dụ: Dãy A gồm 7 phần tử {2, 5, -9, -6, 0, -7, -5}. Dãy con {-9, -6}; {-7, -5} là các dãy con liên tiếp không chung nhau bất kỳ phần tử nào của dãy A.

Lưu ý: Dạng bài tập này áp dụng cho cả trường hợp một phần tử đầu của dãy này trùng với một phần tử cuối của dãy kia.

Bài tập 1: **(Bài toán cơ bản)**

Cho một dãy A gồm N số nguyên (hoặc số thực) $\{a_1, a_2,\ldots, a_N\}$. Dãy con a_i, $a_{i+1},\ldots, a_j$ $(1\le i\le j\le N)$ là dãy được tạo từ các phần tử liên tiếp của dãy A bắt đầu từ phần tử i và kết thúc ở phần tử j. Hãy tìm độ dài dãy con, số lượng dãy con, liệt kê chỉ số các dãy con, liệt kê giá trị các phần tử dãy con thõa mãn một điều kiện nào đó. (Độ dài dãy con là số lượng phần tử dãy con)

Để giải dạng bài tập này ta có thể sử dụng nhiều thuật toán như: thuật toán vét cạn các dãy con hoặc duyệt qua các phần tử của dãy hoặc sử dụng phương pháp quy hoạch động. Đối với dạng bài tập này chúng tôi định hướng cho học sinh lựa chọn thuật toán duyệt qua các phần tử của dãy hoặc quy hoạch động.

Mô hình thuật toán:

Cách 1. Sử dụng phương pháp duyệt qua các phần tử của dãy:

- Duyệt qua tất cả các phần tử của dãy nếu:

+ Thỏa mãn điều kiện, tăng độ dài thêm 1, ngược lại:

Nếu dãy con đang xét cần lưu thì: lưu lại độ dài, chỉ số đầu của dãy, xác định lại độ dài, chỉ số đầu của dãy mới.

Nếu dãy con đang xét không cần lưu thì: lưu lại độ dài, chỉ số đầu của dãy mới.

Cách 2. Sử dụng phương pháp quy hoạch động.

- Gọi L[i] là độ dài dãy con thỏa mãn điều kiện có phần tử cuối là a[i], i=1..n

- Gán giá trị độ dài dãy con trong trường hợp đơn giản: L[0]=0; L[1]=1.

- Tính L[i] nhờ các giá trị bài toán con đã tính từ trước như L[i-1], L[i-2],...

- Kết quả bài toán là sự tổng hợp kết quả từ các bài toán con L[i] (i=1,2,...,n). Từ đó ta có bài tập 1.2 như sau:

Bài tập 1.2: Cho một dãy A gồm N số nguyên $\{a_1, a_2,\ldots, a_N\}$. Dãy con a_i, $a_{i+1},\ldots, a_j (1\le i\le j\le N)$ là dãy được tạo từ các phần tử liên tiếp của dãy A bắt đầu từ phần tử i và kết thúc ở phần tử j.

Yêu cầu: Hãy tìm độ dài và liệt kê giá trị mỗi phần tử của dãy con dài nhất tạo thành cấp số cộng có công sai d.

Dữ liệu vào: File văn bản dayconcsc.inp gồm:

- Dòng đầu ghi giá trị N, d ($2 \le N \le 10^8$; $0 \le d \le 500$).

- Dòng sau gồm N số nguyên$\{a_1, a_2, \ldots, a_N\}$ ($-10^6 \le a_i \le 10^6$) mỗi số cách nhau một dấu cách.

Dữ liệu ra: File văn bản dayconcsc.out gồm

- Dòng đầu ghi độ dài dãy con dài nhất

- Dòng tiếp theo ghi giá trị các phần tử dãy con.

(Chú ý: Nếu không có dãy con nào thỏa mãn thì ghi 0)

Ví dụ:

Dayconcsc.inp	**Dayconcsc.out**
9 4 1 7 6 10 14 6 2 6 10	3 2 6 10 14 2 6 10

Cách 1: Khi gặp bài toán này thông thường học sinh sẽ sử dụng phương pháp vét cạn các dãy con như sau:

Mô hình thuật toán:

```
for (int i=1; i<=n; i++)
   for (int  j=1; j<=n-i+1; j++)
      {
        Xét tất cả các dãy con bắt đầu từ vị  trí i có độ dài j
      }
```

hoặc

```
   for (int k=1; k<= n; k++)
         for (int j=1; j<=n-k+1; j++)
            {
               j:=i+k-1;
               Xét tất cả các dãy con bắt đầu từ vị trí i đến vị trí j với độ dài k
            }
```

Code tham khảo:

```
#include <bits/stdc++.h>
#define N 10001
#define ll long long
using namespace std;
ll a[N], cs[N];
ll n, dmax, d;
//In day con
void inday(ll m, ll l)
{
   for (int i=m; i<=m+l-1; i++)
       cout<<a[i]<<" ";
   cout<<'\n';
}
//Kiem tra day cap so cong
bool kt(ll m,ll l)
{
   for (int i=m; i<=m+l-2; i++)
       if (a[i+1]-a[i] != d) return false;
   return true;
}
int main()
{
   freopen("dayconcsc.inp","r", stdin);
   freopen("dayconcsc.out","w", stdout);
   cin>>n>>d;
   for (int i=1; i<=n; i++) cin>>a[i];
   ///Tim do dai va chi so dau day con thoa man
   dmax=0;
   int k=0;
   for (int i=1; i<=n-1; i++)
```

```
        for (int j=2; j<=n-i+1; j++)
            if (kt(i,j)==true)
                {
                    if (j>dmax){dmax=j; k=0;}
                    if (j==dmax) {k+=1; cs[k]=i;}
                }
    ///In ket qua
    if (dmax==0) cout<<0;
    else
    {
        cout<<dmax<<" "<<k<<'\n';
        for (int i=1; i<=k; i++) inday(cs[i],dmax);
    }
    return 0;
}
```

Sử dụng phần mềm Themis – chấm bài tự động. Ta đo được thời gian thực hiện mỗi test cụ thể như sau:

Cách 1	Độ phức tạp	Test01 (giây)	Test02 (giây)	Test03 (giây)	Test04 (giây)	Test05 (giây)	Test06 (giây)	Test07 (giây)	Test08 (giây)	Test09 (giây)	Test10 (giây)
	0(nlogn)	0.2435	0.2213	0.3553	0.3791	7.9341	93.9473	>100	>100	>100	>100

Với cách này chỉ đạt được 60% số test, vì một số test có dữ liệu lớn ($n>10^6$) chạy quá thời gian.

Cách 2: Dùng quy hoạch động

Mô hình thuật toán

- Gọi L[i] là độ dài dãy con tạo thành cấp số cộng công sai d có chỉ số cuối là i.
- Ta dễ dàng nhận thấy bài toán cơ sở : L[1]=1 ;
- Công thức quy hoạch động là :

Nếu a[i+1]-a[i]=d thì L[i+1]=L[i], ngược lại L[i+1]=1.

- Kết quả bài toán: Max(L[i+1]) với i=1,2,...,n-1.

Code tham khảo:

```
#include <bits/stdc++.h>
#define N 10001
#define ll long long
using namespace std;
ll a[N], L[N], cs[N];
ll n, dmax=0, d, k=1;
int main()
{
   freopen("dayconcsc.inp","r", stdin);
   freopen("dayconcsc.out","w", stdout);
   cin>>n>>d;
   for (int i=1; i<=n; i++) cin>>a[i];
   L[1]=1;
   for (int i=1; i<n; i++)
   {
      if (a[i+1]-a[i]==d) L[i+1]=L[i]+1;
      else L[i+1]=1;
         if (L[i+1]>dmax) {dmax=L[i+1]; k=1; cs[k]=i+1;}
         else
            if (L[i+1]==dmax) {k++; cs[k]=i+1;}
   }
   ///In ket qua
   if (dmax==1) cout<<0;
   else
   {
      cout<<dmax<<" "<<k<<'\n';
      for (int j=1; j<=k; j++)
      {
         for (int i=cs[j]-dmax+1; i<=cs[j]; i++) cout<<a[i]<<" ";
         cout<<'\n';
```

```
        }
    }
    return 0;
}
```

Sử dụng phần mềm Themis – chấm bài tự động. Ta đo được thời gian thực hiện mỗi test cụ thể như sau:

Cách 2	Độ phức tạp	Test01 (giây)	Test02 (giây)	Test03 (giây)	Test04 (giây)	Test05 (giây)	Test06 (giây)	Test07 (giây)	Test08 (giây)	Test09 (giây)	Test10 (giây)
	0(n)	0.0739	0.0603	0.0679	0.0788	0.0657	0.0699	0.2045	0.2057	0.1262	0.1102

Với cách này thì đạt được 100% số test.

So sánh kết quả từ 2 bảng trên và kết quả chấm điểm *bài tập 1.2* bằng phần mềm Themis của 3 cách trên như sau *(mỗi test đúng và thời gian chạy không quá 1 giây được 1 điểm)*. Dễ dàng nhận thấy cách 2 là tối ưu hơn cả mà chương trình ngắn gọn dễ cài đặt phù hợp với năng lực học sinh. Do vậy giải các bài tập dạng này ta nên lựa ***chọn Cách thứ 2***. Cách này có thể lấy được điểm với dãy có số phần tử lớn lên đến $n = 10^8$.

Tương tự bài tập 1.2 ta thay đổi tính chất của dãy con ta có bài tập 1.3

Bài tập 1.3: Cho một dãy A gồm N số nguyên $\{a_1, a_2,\ldots, a_N\}$. Dãy con liên tiếp các phần tử $a_i, a_{i+1},\ldots, a_j$ $(1\le i\le j)$ thỏa mãn điều kiện $a_i<a_{i+1}<\ldots<a_j$ được gọi là dãy con đơn điệu tăng của dãy A.

Yêu cầu: Hãy tìm độ dài và chỉ số dãy con liên tiếp đơn điệu tăng dài nhất.

Dữ liệu vào: File văn bản daycontang.inp gồm:

- Dòng đầu ghi giá trị N $(1\le N\le 10000)$.

- Dòng sau gồm N số nguyên $\{a_1, a_2,\ldots, a_N\}$ $(-10^6\le a_i\le 10^6)$ mỗi số cách nhau một dấu cách.

Dữ liệu ra: File văn bản daycontang.out gồm

- Dòng đầu ghi độ dài dãy con tăng dài nhất (nếu có nhiều dãy con tăng dài nhất thì ghi ra dãy đầu tiên)

- Dòng tiếp theo ghi chỉ số các phần tử dãy con.

Ví dụ:

Daycontang.inp	**Daycontang.out**

12 5 2 3 8 9 10 8 6 7 11 20 33	5 2 3 4 5 6

Thuật toán: Tương tự *bài tập 1.3* chỉ thay điều kiện: a[i-1]<a[i]. Áp dụng cách 2 là tối ưu hơn cả.

Cài đặt chương trình:

{Sử dụng phương pháp quy hoạch động}

```
#include <bits/stdc++.h>
#define N 10001
#define ll long long
using namespace std;
ll a[N], L[N], Truoc[N];
ll n;
void QHD()
{
   fill(L,L+N,0);
   L[1]=1;
   Truoc[1]=-1;
   for (int i=2; i<=n; i++)
   {
      L[i]=1;
      Truoc[i]=-1;
      if (a[i-1]<a[i])
      if (L[i]<L[i-1]+1) {L[i]=L[i-1]+1; Truoc[i]=i-1;}
   }
}
void TruyVet(ll i)
{
   if (Truoc[i]==-1) cout<<i;
   else
   {
      TruyVet(Truoc[i]);
```

```
        cout<<" "<<i;
    }
}
void InKQ()
{
    int MaxL=-1, k=-1;
    for (int i=1; i<=n; i++)
    {
        if (L[i]>MaxL) {MaxL=L[i]; k=i;}
    }
    cout<<MaxL<<'\n';
    TruyVet(k);
}
int main()
{
    freopen("daycontang.inp","r", stdin);
    freopen("daycontang.out","w", stdout);
    cin>>n;
    for (int i=1; i<=n; i++) cin>>a[i];
    QHD();
    InKQ();
    return 0;
}
```

Từ bài tập 1.3 ta thay đổi tính chất các phần tử của dãy con ta có bài tập 1.4 như sau:

Bài tập 1.4*:* Cho một dãy A gồm N số nguyên $\{a_1, a_2,\ldots, a_N\}$. Dãy con a_i, $a_{i+1},\ldots, a_j (1\le i\le j\le N)$ là dãy được tạo từ các phần tử liên tiếp của dãy A bắt đầu từ phần tử i và kết thúc ở phần tử j.

Yêu cầu: Hãy tìm dãy con liên tiếp có số phần tử dương nhiều nhất.

Dữ liệu vào: File văn bản dayconduong.inp gồm:

- Dòng đầu ghi giá trị N ($2\le N\le 10000$).

- Dòng sau gồm N số nguyên{a_1, a_2,…, a_N} ($-10^6 \le a_i \le 10^6$) mỗi số cách nhau một dấu cách.

Dữ liệu ra: File văn bản dayconduong.out gồm

- Dòng đầu ghi độ dài dãy con có số lượng phần tử dương dài nhất
- Dòng tiếp theo ghi giá trị các phần tử dãy con.

(Chú ý: Nếu không có dãy con nào thỏa mãn thì ghi 0)

Ví dụ:

dayconduong.inp	**dayconduong.out**
9 1 3 1 -2 -5 0 1 17 12	3 1 3 1

Hướng dẫn thuật toán: Tương tự *bài tập 1.3* áp dụng cách 3 chỉ thay:

- Khởi tạo: mảng L=0;
- Công thức quy hoạch động:

```
Nếu a[i]>0 thì
    {
        Truoc[i]=-1;
        L[i]=1;
        Nếu  (L[i]<L[i-1]+1) thì
            {
                L[i]=L[i-1]+1;
                Truoc[i]=i-1;
            }
    }
```

Code tham khảo:

```
#include <bits/stdc++.h>
#define N 10001
#define ll long long
using namespace std;
ll a[N], L[N], Truoc[N];
ll n;
void QHD()
```

```
{
   fill(L,L+N,0);

   for (int i=1; i<=n; i++)
   {
      if (a[i]>0)
      {
         Truoc[i]=-1;
         L[i]=1;
         if (L[i]<L[i-1]+1) {L[i]=L[i-1]+1; Truoc[i]=i-1;}
      }

   }
}
void TruyVet(ll i)
{
   if (Truoc[i]==-1) cout<<a[i];
   else
   {
      TruyVet(Truoc[i]);
      cout<<" "<<a[i];
   }
}
void InKQ()
{
   int MaxL=0, k=0;
   for (int i=1; i<=n; i++)
   {
      if (L[i]>MaxL) {MaxL=L[i]; k=i;}
   }
   cout<<MaxL<<'\n';
```

```
    TruyVet(k);
}
int main()
{
    freopen("dayconduong.inp","r", stdin);
    freopen("dayconduong.out","w", stdout);
    cin>>n;
    for (int i=1; i<=n; i++) cin>>a[i];
    QHD();
    InKQ();
    return 0;
}
```

Các dãy con có thể chung nhau phần tử của dãy ban đầu nghĩa là những phần tử của dãy mẹ đã thuộc dãy con thỏa mãn này thì vẫn có thể thuộc hoặc không thuộc các dãy con thỏa mãn khác.

Ví dụ: Dãy ban đầu gồm 7 phần tử {1,2,3,6,9,-6,8}. Dãy con {**1**,**2**,**3**}; {**1**,**2**,**3**,6}; {-6,8} là các dãy con có thể chung nhau phần tử của dãy ban đầu từ đó ta có bài tập 1.5 như sau:

Bài tập 1.5: Cho một dãy A gồm N số nguyên $\{a_1, a_2,\ldots, a_N\}$. Dãy con $a_i, a_{i+1}, \ldots, a_j (1\le i\le j\le N)$ là dãy được tạo từ các phần tử liên tiếp của dãy A bắt đầu từ phần tử i và kết thúc ở phần tử j. Tìm các dãy con thõa mãn một điều kiện nào đó.

Để giải dạng bài tập này ta có thể sử dụng thuật toán vét cạn các dãy con hoặc sử dụng phương pháp quy hoạch động. Đối với dạng bài tập này chúng tôi định hướng cho học sinh lựa chọn thuật toán quy hoạch động.

Mô hình thuật toán:

- Gọi L[i] là giá trị dãy con (tùy điều kiện bài toán) từ phần tử thứ 1 đến phần tử thứ i

- Lập công thức tính giá trị dãy con từ i đến j theo L[i] và L[j].

- Xét tất cả các cặp số (i, j) bằng hai vòng lặp sau:

```
for (int i=1; i< n; i++)
{
    for (int  j=i; j<=n; j++)
      {
```

Xét các dãy con từ phần tử i đến j thỏa mãn điều kiện thì tăng số dãy, lưu chỉ số đầu, chỉ số cuối

}

}

Bài tập 1.6: Cho một dãy A gồm N số nguyên $\{a_1, a_2, \ldots, a_N\}$. Dãy con a_i, a_{i+1}, …, $a_j (1 \le i \le j \le N)$ là dãy được tạo từ các phần tử liên tiếp của dãy A bắt đầu từ phần tử i và kết thúc ở phần tử j.

Yêu cầu: Hãy tìm dãy con liên tiếp có tổng lớn nhất.

Dữ liệu vào: File văn bản tonglt.inp gồm:

- Dòng đầu ghi giá trị N ($1 \le N \le 10000$).
- Dòng sau gồm N số nguyên $\{a_1, a_2, \ldots, a_N\}$ ($-10^6 \le a_i \le 10^6$) mỗi số cách nhau một dấu cách.

Dữ liệu ra: File văn bản tonglt.out gồm

- Dòng đầu ghi tổng các phần tử dãy con và số lượng dãy con.
- Dòng tiếp theo ghi giá trị các phần tử dãy con.

Ví dụ:

Tonglt.inp	**Tonglt.out**
13 12 -34 14 11 9 -8 15 11 -7 -56 17 16 19	52 2 14 11 9 -8 15 11 17 16 19

Cách 1: Khi gặp bài toán này thông thường học sinh sẽ sử dụng phương pháp vét cạn các dãy con như sau:

Mô hình thuật toán:

Code tham khảo:

```
#include <bits/stdc++.h>
#define N 10001
#define ll long long
using namespace std;
ll a[N], dau[N], cuoi[N];
ll n;
ll tong(ll m, ll l)
{
```

```
    ll t=0;
    for (int i=m; i<=m+l-1; i++)
        t+=a[i];
    return t;
}
int main()
{
    freopen("tonglt.inp","r", stdin);
    freopen("tonglt.out","w", stdout);
    cin>>n;
    for (int i=1; i<=n; i++) cin>>a[i];
    ll tmax=a[1];
    int k=0;
    for (int i=1; i<=n-1; i++)
        for (int j=1; j<=n-i+1; j++)
        {
            ll t=tong(i,j);
            if (t>tmax) {tmax=t; k=0;}
            if (t==tmax) {k+=1;dau[k]=i; cuoi[k]=i+j-1;}
        }
    cout<<tmax<<" "<<k<<'\n';
    for (int i=1; i<=k; i++)
    {
        for (int j=dau[i]; j<=cuoi[i]; j++) cout<<a[j]<<" ";
        cout<<'\n';
    }
    return 0;
}
```

Sử dụng phần mềm Themis. Ta đo được thời gian thực hiện mỗi test cụ thể như sau:

Cách	Độ phức	Test0	Test0	Test0	Test0	Test0	Test0	Test0	Test0	Test0	Test1

1	tạp	1 (giây)	2 (giây)	3 (giây)	4 (giây)	5 (giây)	6 (giây)	7 (giây)	8 (giây)	9 (giây)	0 (giây)
	$0(n^2 \log n)$	0.0745	0.3465	2.5423	53.666	>100	>100	>100	>100	>100	>100

Với cách này thì đạt được 20% số test. Vì một số test có dữ liệu lớn chạy quá thời gian.

Cách 2: Sử dụng phương pháp quy hoạch động.

Mô hình thuật toán:

- Gọi L[i] là tổng tất cả các phần tử từ 1 đến i.
- Như vậy dãy con liên tiếp từ i đến j có tổng là: L[j]-L[i-1].
- Xét tất cả các dãy con từ i đến j bằng 2 vòng lặp:

```
for i:= 1 to n-1 do
    for j:= i to n do
        begin
            {Xét L[j]-L[i-1] thỏa mãn điều kiện thì tăng số dãy, lưu chỉ số đầu, chỉ số cuối}
        end;
```

Code tham khảo:

```
#include <bits/stdc++.h>
#define N 10001
#define ll long long
using namespace std;
ll a[N], L[N], dau[N], cuoi[N];
ll n;
int main()
{
   freopen("tonglt.inp","r", stdin);
   freopen("tonglt.out","w", stdout);
   cin>>n;
   for (int i=1; i<=n; i++) cin>>a[i];
```

```
L[0]=0;
for (int i=1; i<=n; i++) L[i]=L[i-1]+a[i];
ll tmax=a[1];
int k=0;
for (int i=1; i<=n-1; i++)
   for (int j=i; j<=n; j++)
   {
      if (L[j]-L[i-1]>tmax) {tmax=L[j]-L[i-1]; k=0;}
      if (L[j]-L[i-1]==tmax) {k++; dau[k]=i; cuoi[k]=j;}
   }
cout<<tmax<<" "<<k<<'\n';
for (int i=1; i<=k; i++)
{
   for (int j=dau[i]; j<=cuoi[i]; j++)
      cout<<a[j]<<" ";
   cout<<'\n';
}
return 0;
}
```

Sử dụng phần mềm Themis. Ta đo được thời gian thực hiện mỗi test cụ thể như sau:

Cách 2	Độ phức tạp	Test01 (giây)	Test02 (giây)	Test03 (giây)	Test04 (giây)	Test05 (giây)	Test06 (giây)	Test07 (giây)	Test08 (giây)	Test09 (giây)	Test10 (giây)
	$0(n^2)$	0.0556	0.0886	0.1123	0.1312	0.1764	0.2754	0.3352	0.3733	0.4373	0.5364

Với cách này thì đạt được 100% số Test. Do đó nên sử dụng cách thứ 2

So sánh kết quả từ 2 bảng trên và kết quả chấm điểm *bài toán 1.6* bằng phần mềm Themis của 2 cách trên như sau *(mỗi test đúng và thời gian chạy không quá 1 giây/ 1 test được 1 điểm)***.** Dễ dàng nhận thấy cách 2 là tối ưu hơn mà chương trình ngắn gọn dễ cài đặt phù hợp với năng lực học sinh trường chúng tôi. Do vậy giải các bài tập dạng này ta nên lựa ***chọn cách 2.***

Bài tập 1.7: Cho một dãy A gồm N số nguyên $\{a_1, a_2, \ldots, a_N\}$ và một số nguyên K. Dãy con $a_i, a_{i+1}, \ldots, a_j$ $(1 \le i \le j \le N)$ là dãy được tạo từ các phần tử liên tiếp của dãy A bắt đầu từ phần tử i và kết thúc ở phần tử j.

Yêu cầu: Hãy tìm dãy con liên tiếp có tổng các phần tử chia hết cho K.

Dữ liệu vào: File văn bản dayconchiam.inp gồm:

- Dòng đầu ghi hai số nguyên N và M ($1 \le N \le 10000$; $1 \le M \le 1000$).

- Dòng sau gồm N số nguyên$\{a_1, a_2, \ldots, a_N\}$ ($10^6 \le a_i \le 10^6$, $a_i <> 0$) mỗi số cách nhau một dấu cách.

Dữ liệu ra: File văn bản dayconchiam.out gồm

- Dòng đầu ghi số lượng dãy con thỏa mãn.

- Dòng tiếp theo ghi giá trị các phần tử dãy con thỏa mãn.

Ví dụ:

dayconchiak.inp	dayconchiak.out
4 3 1 3 2 5	2 1 3 2 3

Thuật toán:

Tương tự cách 2 *bài tập 1.6* chỉ thay điều kiện bằng L[j]-L[i-1] mod k =0.

Code tham khảo:

{Sử dụng phương pháp quy hoạch động}

```
#include <bits/stdc++.h>
#define N 10001
#define ll long long
using namespace std;
ll a[N], L[N], dau[N], cuoi[N];
ll n, m;
int main()
{
   freopen("dayconchiak.inp","r", stdin);
   freopen("dayconchiak.out","w", stdout);
   cin>>n>>m;
```

```
    for (int i=1; i<=n; i++) cin>>a[i];
    L[0]=0;
    for (int i=1; i<=n; i++) L[i]=L[i-1]+a[i];
    ll tmax=a[1];
    int k=0;
    for (int i=1; i<=n-1; i++)
        for (int j=i; j<=n; j++)
        {
            if ((L[j]-L[i-1])% m == 0) {k++; dau[k]=i; cuoi[k]=j;}
        }
    cout<<k<<'\n';
    for (int i=1; i<=k; i++)
    {
        for (int j=dau[i]; j<=cuoi[i]; j++)
            cout<<a[j]<<" ";
        cout<<'\n';
    }
    return 0;
}
```

2.5 Các bài toán về dãy con không liên tiếp

Bài tập 2: (Bài toán cơ bản)

Cho một dãy A gồm N số nguyên $\{a_1, a_2,\ldots, a_N\}$. Dãy con $a_i, a_{i+1},\ldots, a_j (1\le i\le j\le N)$ là dãy thu được khi ta xóa một số phần tử (có thể không xóa phần tử nào và không được xóa hết) của một dãy cho trước và giữ nguyên thứ tự các phần tử còn lại trong dãy. Hãy tìm các dãy con thỏa mãn một điều kiện nào đó.

Đối với dạng bài tập này chúng tôi định hướng cho học sinh lựa chọn thuật toán quy hoạch động là tối ưu hơn cả.

Thuật toán chung

- Phân rã bài toán
- Gán giá trị cho bài toán cơ sở.
- Tính giá trị cho bài toán thứ i nhờ các bài toán đã tính trước đó.
- Kết quả bài toán là sự tổng hợp từ các bài toán ban đầu.

Bài tập 2.1: Dãy con không giảm

Cho một dãy A gồm N số nguyên $\{a_1, a_2,\ldots, a_N\}$. Dãy con $a_i\le a_{i+1}\le\ldots\le a_j (1\le i\le j\le N)$ được gọi là dãy con không giảm của dãy A. Lưu ý các phần tử của dãy con có thể chọn liên tiếp hoặc không liên tiếp từ các phần tử dãy A nhưng phải theo đúng thứ tự. Độ dài của dãy con là số lượng phần tử của dãy con đó.

Yêu cầu: Tìm độ dài lớn nhất của dãy con không giảm.

***Dữ liệu vào*:** File văn bản dayconkogiam.inp gồm 2 dòng:

- Dòng đầu chứa một số nguyên dương N ($1\le N\le 10^4$).
- Dòng thứ hai chứa N số nguyên dương a_i ($-10^5\le a_i\le 10^5$), giữa hai số cách nhau bởi một dấu cách.

Dữ liệu ra: File văn bản dayconkogiam.out gồm 2 dòng:

Dòng thứ nhất chứa số nguyên dương là số lượng dãy con tăng dài nhất

Dòng thứ 2: là dãy con tăng dài nhất

Ví dụ:

Dayconkogiam.inp	**Dayconkogiam.out**
8 5 1 6 4 5 2 1 7	4 1 4 5 7

Thuật toán:

- Không mất tính tổng quát ta gán: a[0]= ∞; a[n+1] = +∞;
- Gọi F[i] là độ dài dãy con không giảm có phần tử cuối là a[i]. (i = 1 đến n+1)
- Dễ dàng nhận thấy: F[0] = 1;
- Ta có công thức quy hoạch động:

  F[i] = max(F[i], F[j] + 1) nếu a[j] <= a[i] (j = 1..i-1)
- Kết quả bài toán là max(F[n+1] - 2). (Trừ 2 phần tử a[0] = -∞ và a[n+1] = +∞)

Code tham khảo:

```
#include <bits/stdc++.h>
#define N 1002
#define inf 999999999
using namespace std;
int a[N], F[N];
int n, res=0;
int main()
{
   freopen("dayconkogiam.inp","r", stdin);
   freopen("dayconkogiam.out","w", stdout);
   cin>>n;
   for (int i=1; i<=n; i++) cin>>a[i];
   a[0]=-inf; a[n+1]=inf;
   F[0]=1;
   for (int i=1; i<=n+1; i++)
      for (int j=0; j<i; j++)
         if (a[i] > a[j] )
            F[i]=max(F[i], F[j]+1);
   res=F[n+1]-2;
   cout<<res<<'\n';
//Truy vet
   vector <int> kq;
   int i=n+1;
```

```
    while (i>0)
    {
      for (int j=i-1; j>=0; j--)
          if (a[i]>a[j] && F[i]==F[j]+1)
      {
          kq.push_back(a[j]);
          i=j;
          break;
      }
    }
    for (int i=kq.size()-2; i>=0; i--) cout<<kq[i]<<" ";
    return 0;
}
```

Sử dụng phần mềm Themis. Ta đo được thời gian thực hiện mỗi test:

Độ phức tạp	Test01 (giây)	Test02 (giây)	Test03 (giây)	Test04 (giây)	Test05 (giây)	Test06 (giây)	Test07 (giây)	Test08 (giây)	Test09 (giây)	Test10 (giây)
$0(n^2)$	0.067	0.055	0.056	0.056	0.061	0.192	0.26	0.288	0.301	0.377

Từ bảng trên và kết quả chấm điểm *bài toán trên* bằng phần mềm Themis:

Ta dễ dàng nhận thấy cách làm này rất tối ưu, thời gian thực hiện chương trình ngắn. Đây là cách làm đạt được 100% số test.

Lưu ý: Các bài tập dãy con về dạng này rất khó nên học sinh phải linh hoạt trong cách giải và nên sử dụng phương pháp quy hoạch động thì chương trình sẽ tối ưu hơn. Từ bài tập trên ta thay đổi tính chất của dãy con thì ta có các bài toán như sau:

Bài tập 2.2: Bố trí phòng họp (mất tính thứ tự so với dãy ban đầu)

Có n cuộc họp, cuộc họp thứ i bắt đầu vào thời điểm a_i và kết thúc ở thời điểm b_i. Do chỉ có một phòng hội thảo nên 2 cuộc họp bất kì sẽ được cùng bố trí phục vụ nếu khoảng thời gian làm việc của chúng chỉ giao nhau tại đầu mút. Hãy bố trí phòng họp để phục vụ được nhiều cuộc họp nhất.

Dữ liệu: từ file PHONGHOP.INP.

- Dòng đầu là số n ($1 \le N \le 1000$) là số cuộc họp.

- N dũng tiếp theo mỗi dũng ghi 2 số a_i và b_i ($1 \le i \le N$) cách nhau một dấu trắng là thời điểm ban đầu và kết thúc cuộc họp thứ i.

Kết quả: PHONGHOP.OUT

- Dòng thứ nhất ghi số lượng lớn nhất các cuộc họp
- Dòng thứ hai ghi số hiệu của cuộc họp.

Ví dụ:

PHONGHOP.INP	**PHONGHOP.OUT**
7 1 4 2 5 1 7 6 8 4 10 5 8 12 16	3 1 4 7

Hướng dẫn: Sắp xếp các cuộc họp tăng dần theo thời điểm kết thúc (b_i). Thế thì cuộc họp i sẽ bố trí được sau cuộc họp j nếu và chỉ nếu j<i và b_j<=a_i. Yêu cầu bố trí được nhiều cuộc họp nhất có thể đưa về việc tìm dãy các cuộc họp dài nhất thoả mãn điều kiện trên.

Code tham khảo:

```
#include <bits/stdc++.h>
#define N 1002
#define inf int (1e9)
struct hop
{
   int d, c, id;
};
using namespace std;
hop a[N];
int F[N], vet[N];
int n;
bool cmp(hop x, hop y)
{
```

```
    return (x.c<=y.c);
}
int main()
{
    freopen("phonghop.inp", "r", stdin);
    freopen ("phonghop.out", "w", stdout);
    cin>>n;
    int maxy=0;
    for (int i=1; i<=n; i++)
    {
        int x, y;
        cin>>x>>y;
        a[i].d=x;
        a[i].c=y;
        a[i].id=i;
        maxy=max(maxy,y);
    }
    sort(a+1,a+n+1,cmp);
    a[n+1].c=inf; a[n+1].d=inf; a[n+1].id=n+1;
    F[0]=1;
    for (int i=1; i<=n+1; i++)
        for (int j=0; j<i; j++)
        {
            if (a[j].c<=a[i].d && F[i]<F[j]+1)
            {
                F[i]=F[j]+1;
                vet[i]=j;
            }
        }
    cout<<F[n+1]-2<<'\n';
    vector <int> kq;
```

```
    int j=vet[n+1];
    while (j>0)
    {
       kq.push_back(a[j].id);
       j=vet[j];
    }
    for (int i=kq.size()-1; i>=0; i--) cout<<kq[i]<<" ";
    return 0;
}
```

Bài tập 2.3: Cho thuê máy

Trung tâm tính toán hiệu năng cao nhận được đơn đặt hàng của n khách hàng. Khách hàng i muốn sử dụng máy trong khoảng thời gian từ a_i đến b_i và trả tiền thuê là c_i.

Hãy bố trí lịch thuê máy để tổng số tiền thu được là lớn nhất mà thời gian sử dụng máy của 2 khách hàng bất kì được phục vụ đều không giao nhau (cả trung tâm chỉ có một máy cho thuê).

Dữ liệu: vào từ file văn bản **THUEMAY.INP**.

- Dòng đầu là số N ($0 < N \leq 1000$)
- N dòng tiếp theo dòng thứ i ghi 3 số A[i], B[i], C[i] cách nhau bởi một dấu cách.

Kết quả: ghi ra file **THUEMAY.OUT**

- Dòng đầu tiên ghi hai sô nguyên theo thứ tự là số lượng khách hàng được phục vụ và tổng tiền thu được
- Dòng thứ hai là chỉ số của các khách hàng được phục vụ

Ví dụ :

THUEMAY.INP	THUEMAY.OUT
4 400 821 800 200 513 500 100 325 200 600 900 600	2 1100 2 4

Hướng dẫn: Tương tự như bài toán a), nếu sắp xếp các đơn đặt hàng theo thời điểm kết thúc, ta sẽ đưa được bài toán b) về bài toán tìm dãy con có tổng lớn nhất. Bài toán này là biến thể của bài toán tìm dãy con tăng dài nhất, ta có thể cài đặt bằng đoạn chương trình như sau:

```
#include <bits/stdc++.h>
#define N 1002
#define inf int (1e9)
struct thue
{
   int d, c, id, t;
};
using namespace std;
thue a[N];
int F[N], vet[N];
int n;
bool cmp(thue x, thue y)
{
   return (x.c<=y.c);
}
int main()
{
   freopen("thuemay.inp", "r", stdin);
   freopen ("thuemay.out", "w", stdout);
   cin>>n;
   int maxy=0;
   for (int i=1; i<=n; i++)
   {
      int x, y, z;
      cin>>x>>y>>z;
      a[i].d=x;
      a[i].c=y;
      a[i].t=z;
```

```
        a[i].id=i;
    }
    sort(a+1,a+n+1,cmp);
    a[n+1].c=inf; a[n+1].d=inf; a[n+1].t=0; a[n+1].id=n+1;
    F[0]=0;
    for (int i=1; i<=n+1; i++)
        for (int j=0; j<i; j++)
        {
            if (a[j].c<=a[i].d && F[i]<F[j]+a[i].t)
            {
                F[i]=F[j]+a[i].t;
                vet[i]=j;
            }
        }
    vector <int> kq;
    int j=vet[n+1];
    while (j>0)
    {
        kq.push_back(a[j].id);
        j=vet[j];
    }
    cout<<kq.size()<<" "<<F[n]<<'\n';
    for (int i=kq.size()-1; i>=0; i--) cout<<kq[i]<<" ";
    return 0;
}
```

Bài tập 2.4: Dãy tam giác bao nhau

Cho n tam giác trên mặt phẳng. Tam giác i bao tam giác j nếu 3 đỉnh của tam giác j đều nằm trong tam giác i (có thể nằm trên cạnh). Hãy tìm dãy tam giác bao nhau có nhiều tam giác nhất.

Dữ liệu: vào từ file văn bản **TAMGIAC.INP.**

- Dòng đầu là số N ($0 < N \leq 1000$)

- N dòng tiếp theo dòng thứ i ghi 6 số X_{Ai}, Y_{Ai}, X_{Bi}, Y_{Bi}, X_{Ci}, Y_{Ci} cách nhau bởi một dấu cách là toạ độ của 3 đỉnh của tam giác thứ i.

Kết quả: ghi ra file **TAMGIAC.OUT.**

- Dòng đầu tiên ghi sô nguyên là số tam giác bao nhau nhiều nhất.
- Dòng thứ hai ghi chỉ số của các tam giác bao nhau, từ tam giác nhỏ nhất đến tam giác lớn nhất.

Ví dụ :

TAMGIAC.INP	TAMGIAC.OUT
3 1 1 2 4 5 2 2 2 2 3 3 3 0 3 2 4 5 2	2 2 2 2 3 3 3 ->1 1 2 4 5 2

Hướng dẫn: Sắp xếp các tam giác tăng dần về diện tích. Khi đó tam giác i sẽ bao tam giác j nếu j < i và 3 đỉnh của i nằm trong j. Từ đó đưa bài toán về tìm dãy tăng dài nhất. Việc kiểm tra điểm M có nằm trong tam giác ABC không có thể dựa trên phương pháp tính diện tích: điểm M nằm trong nếu S(ABC) = S(ABM) + S(ACM) + S(BCM). Bài toán có một số biến thể khác như tìm dãy hình tam giác, hình chữ nhật… bao nhau có tổng diện tích lớn nhất.

Code tham khảo:

```
#include <bits/stdc++.h>
#define Nmax 1002
#define inf int (1e9)
struct dinh
{
   int x, y;
};
using namespace std;
dinh a[Nmax], b[Nmax], c[Nmax];
int F[Nmax], vet[Nmax];
float S[Nmax];
int n;
float dt(dinh A, dinh B, dinh C)
```

```
{
   float AB=sqrt(pow((B.x-A.x),2)+pow((B.y-A.y),2));
   float BC=sqrt(pow((C.x-B.x),2)+pow((B.y-C.y),2));
   float AC=sqrt(pow((C.x-A.x),2)+pow((C.y-A.y),2));
   float p=(AB+BC+AC)/2;
   return sqrt(p*(p-AB)*(p-BC)*(p-AC));
}
void hoandoi(float &A, float &B)
{
   float tg=A;
   A=B;
   B=tg;
}
void hoandoi1(dinh &A, dinh &B)
{
   dinh tg=A;
   A=B;
   B=tg;
}
void SX(int T, int P)
{
   float x=S[(T+P)/2];
   int i=T;
   int j=P;
   while (i<j)
   {
      while (S[i]<x) i++;
      while (S[j]>x) j--;
      if (i<=j)
      {
         hoandoi(S[i],S[j]);
```

```
            hoandoi1(a[i],a[j]);
            hoandoi1(b[i],b[j]);
            hoandoi1(c[i],c[j]);
            i++;
            j--;
        }
    }
    if (T<j) SX(T,j);
    if (i<P) SX(i,P);
}
bool kt(dinh A, dinh B, dinh C, dinh M, dinh N, dinh P)
    {
        bool kt1=true;
        float s=dt(A,B,C);
        if (abs(s-(dt(A,B,M)+dt(A,C,M)+dt(B,C,M)))>0.0001) {kt1=false;
exit;}
        if (abs(s-(dt(A,B,N)+dt(A,C,N)+dt(B,C,N)))>0.0001) {kt1=false; exit;}
        if (abs(s-(dt(A,B,P)+dt(A,C,P)+dt(B,C,P)))>0.0001) {kt1=false; exit;}
        return kt1;
    }
void  QHD()
    {
        F[1]=1;
        vet[1]=-1;
        for (int i=2; i<=n; i++)
        {
            F[i]=1;
            vet[i]=-1;
            for (int j=1; j<=i-1; j++)
                if (kt(a[i],b[i],c[i],a[j],b[j],c[j]))
                    if (F[i]<F[j]+1)
```

```
                {
                    F[i]=F[j]+1;
                    vet[i]=j;
                }
        }
    }
void TruyVet(int i)
    {
        if (vet[i]==-1)
        {
            cout<<a[i].x<<" "<<a[i].y<<" ";
            cout<<b[i].x<<" "<<b[i].y<<" ";
            cout<<c[i].x<<" "<<c[i].y<<" ";
        }
        else
        {
            TruyVet(vet[i]);
            cout<<"->"<<a[i].x<<" "<<a[i].y<<" ";
            cout<<b[i].x<<" "<<b[i].y<<" ";
            cout<<c[i].x<<" "<<c[i].y<<" ";
        }
    }
void InKQ()
    {
        float maxL=-1;
        int k=-1;
        for (int i=1; i<=n; i++)
        {
            if (F[i]>maxL)
            {
                maxL=F[i];
```

```
                k=i;
              }
            }
            cout<<maxL<<'\n';
            TruyVet(k);
        }
    int main()
    {
        freopen("tamgiac.inp", "r", stdin);
        freopen("tamgiac.out", "w", stdout);
        cin>>n;
        for (int i=1; i<=n; i++)
        {
            cin>>a[i].x>>a[i].y>>b[i].x>>b[i].y>>c[i].x>>c[i].y;
            S[i]=dt(a[i],b[i],c[i]);
        }
        SX(1,n);
        QHD();
        InKQ();
        return 0;
    }
```

Bài tập 2.5: **Dãy số WAVIO**

Dãy số Wavio là dãy số nguyên thỏa mãn các tính chất : các phần tử đầu sắp xếp thành 1 dãy tăng dần đến 1 phần tử đỉnh sau đó giảm dần. Ví dụ dãy số 1 2 3 4 5 2 1 là 1 dãy Wavio độ dài 7. Cho 1 dãy gồm N số nguyên, hãy chỉ ra một dãy con Wavio có độ dài lớn nhất trích ra từ dãy đó.

Ví dụ :

WAVIO.INP	WAVIO.OUT
10 5 3 4 9 2 6 1 7 8 4	6

Hướng dẫn: F1[i] là mảng ghi độ dài lớn nhất của 1 dãy con tăng dần trích ra từ dãy N phần tử kể từ phần tử 1 đến phần tử a_i. F2[i] : mảng ghi độ dài lớn nhất

của dãy con giảm dần trích ra từ dãy N phần tử kể từ phần tử aN đến ai. Ta tìm phần tử j trong 2 mảng F1, F2 thỏa mãn F1[j]+F2[j] lớn nhất.

Code tham khảo:

```
#include <bits/stdc++.h>
#define N 1002
using namespace std;
int a[N], F1[N], F2[N];
int n, res=0;
int main()
{
   freopen("wavio.inp","r", stdin);
   freopen("wavio.out","w", stdout);
   cin>>n;
   for (int i=1; i<=n; i++) cin>>a[i];
   F1[1]=1;
   for (int i=2; i<=n+1; i++)
      for (int j=1; j<i; j++)
         if (a[i] > a[j] )
            F1[i]=max(F1[i], F1[j]+1);
   for (int i=1; i<=n; i++) cin>>a[i];
   F2[n]=1;
   for (int i=n-1; i>0; i--)
      for (int j=n; j>i; j--)
         if (a[i] > a[j] )
            F2[i]=max(F2[i], F2[j]+1);

   for (int i=1; i<=n; i++)
      res=max(res,F1[i]+F2[i]);
   cout<<res<<'\n';
   return 0;
}
```

Bài tập 2.6: Xếp các khối đá :

Cho N khối đá (N≤5000) Các khối đá đều có dạng hình hộp chữ nhật và được đặc trưng bởi 3 kích thước: dài, rộng, cao. Một cách xây dựng tháp là một cách đặt một số các khối đá trong các khối đá đã cho chồng lên nhau theo quy tắc:

• Chiều cao mỗi khối đá là kích thước nhỏ nhất trong 3 kích thước.

• Các mép của khối đá được đặt song song với nhau sao cho không có phần nào của khối trên nằm chìa ra ngoài khối dưới.

Hãy chỉ ra cách để xây dựng được một cái tháp sao cho chiều cao của cái tháp là cao nhất Dữ liệu vào XEPDA.INP có cấu trúc như sau :

Dòng đầu là số N.

N dòng sau dòng i ghi 3 số nguyên ≤ 255 là 3 kích thước của khối đá i .

Dữ liệu ra : XEPDA.OUT ghi theo quy cách :

Dòng đầu ghi chiều cao lớn nhất của tháp.

Các dòng sau ghi thứ tự các khối được chọn từ đáy lên đỉnh, mỗi khối đá ghi 4 số T, D, R, C trong đó T là số thứ tự của mỗi khối đá. D, R, C là kích thước của khối đá tương ứng.

Ví dụ :

XEPDA.INP	XEPDA.OUT
5 2 2 3 4 1 1 3 4 5 7 3 3 9 8 1	6 5 9 8 1 4 7 3 3 1 3 2 2

Hướng dẫn:

- Sắp xếp các khối đá theo chiều tăng dần chiều dài và chiều rộng.
- Gọi Li là chiều cao lớn nhất các khối đá xếp đc từ 1-> ai
- Quy hoạch động giống dẫy đơn điệu dài nhất

Code tham khảo:

```
#include <bits/stdc++.h>
#define N 1002
#define inf int (1e9)
struct da
```

```
{
   int t, d, r, c;
};
using namespace std;
da a[N];
int F[N], vet[N];
int n;
bool cmp(da x, da y)
{
   return (x.d>=y.d);
}
int main()
{
   freopen("xepda.inp", "r", stdin);
   freopen ("xepda1.out", "w", stdout);
   cin>>n;
   int maxy=0;
   for (int i=1; i<=n; i++)
   {
      int x, y, z;
      cin>>x>>y>>z;
      a[i].d=max(x,max(y,z));
      a[i].c=min(x,min(y,z));
      if (( x<= y && y<=z) || (z<= y && y<=x)) a[i].r=y;
      if (( y<= x && x<=z) || (z<= x && x<=y)) a[i].r=x;
      if (( x<= z && z<=y) || (y<= z && z<=x)) a[i].r=z;
      a[i].t=i;
   }
   sort(a+1,a+n+1,cmp);
   a[n+1].d=0; a[n+1].r=0; a[n+1].c=inf; a[n+1].t=n+1;
   a[0].d=inf; a[0].r=inf; a[0].c=0; a[0].t=0;
```

```
    F[0]=0;
    for (int i=1; i<=n+1; i++)
       for (int j=0; j<i; j++)
       {
          if (a[j].r>=a[i].r && F[i]<F[j]+a[i].c)
          {
             F[i]=F[j]+a[i].c;
             vet[i]=j;
          }
       }
    vector <int> kq;
    int j=vet[n+1];
    while (j>0)
    {
       kq.push_back(j);
       j=vet[j];
    }
    cout<<F[n+1]-inf<<'\n';
    for (int i=kq.size()-1; i>=0; i--) cout<<a[kq[i]].t<<" "<<a[kq[i]].d<<"
"<<a[kq[i]].r<<" "<<a[kq[i]].c<<'\n';
    return 0;
}
```

Bài bài tập 2.7: Dãy đổi dấu

Cho dãy a1, a2,…an. Hãy dãy con đổi dấu dài nhất của dãy đó. Dãy con con đổi dấu ai1,ai2,…aik phải thoả mãn các điều kiện sau:

• ai1<ai2>ai3<… hoặc ai1>ai2<ai3>…

• Các chỉ số phải cách nhau ít nhất L: i2–i1≥L, i3–i2≥L….

• Chênh lệch giữa 2 phần tử liên tiếp nhỏ hơn U: |ai1–ai2|≤U, |ai2–ai3|≤U…

Input :

– Dòng đầu tiên ghi ba số n, l, u.
– Dòng thứ hai ghi n số là các phần tử của dãy A.

Output :

– Ghi độ dài lớn nhất của dãy con đổi dấu.

Ví dụ

DAYDOIDAU.INP	DAYDOIDAU.OUT
5 2 2 5 4 3 2 4	3

Hướng dẫn: Gọi L(i) là số phần tử của dãy con đổi dấu có phần tử cuối cùng là ai và phần tử cuối cùng lớn hơn phần tử đứng trước. Tương tự, P(i) là số phần tử của dãy con đổi dấu có phần tử cuối cùng là ai và phần tử cuối cùng nhỏ hơn phần tử đứng trước.

Ta dễ dàng suy ra:

• L(i) = max(1, P(j)+1): j≤i–L và ai–U≤aj<aj≤ai+U

• P(i) = max(1, L(j)+1): j≤i–L và ai<aj≤ai+U.

Code tham khảo:

```
#include <bits/stdc++.h>
#define Nmax 1002
#define inf int (1e9)
using namespace std;
int a[Nmax], F[Nmax], P[Nmax];
int n, l, u;
int main()
{
   freopen("daydoidau.inp", "r", stdin);
   freopen("daydoidau.out", "w", stdout);
   cin>>n>>l>>u;
   for (int i=1; i<=n; i++) cin>>a[i];
   F[1]=1; P[1]=1;
   int k=F[1];
   for (int i=2; i<=n; i++)
   {
      F[i]=1; P[i]=1;
      for (int j=1; j<=i-1; j++)
```

```
            if (abs(a[i]-a[j])<=u)
            {
               if ((a[i]>a[j]) && (i-j>=l) && (F[i]<P[j]+1))
                  F[i]=P[j]+1;
               if ((a[i]<a[j]) && (i-j>=l) && (P[i]<F[j]+1))
                  P[i]=F[j]+1;
            }
         if ((k<F[i]) || (k<P[i])) k=max(F[i],P[i]);
      }
      cout<<k;
      return 0;
   }
```

III. KẾT LUẬN VÀ KIẾN NGHỊ

1. Kết quả nghiên cứu :

Trong năm học 2020 - 2021 Chúng tôi đã ứng dụng đề tài nghiên cứu của mình đối với một số lớp khối 11 ở trường THPT Lê Viết Thuật và đã tổng hợp số liệu về kết quả đạt được của học sinh như sau:

STT	Lớp	Sĩ số	Giỏi	Khá	Trung bình	Không đạt yêu cầu
1	$11T_1$	35	17%	54%	29%	
2	$11A_1$	40	12%	43%	42%	3%
3	$11A_2$	38	10%	38%	47%	5%

Sau một nhiều năm áp dụng dạy học sinh khối 11 áp dụng cách làm này chúng tôi nhận thấy:

- Các em khắc sâu kiến thức cơ bản của sách giáo khoa. Kỹ năng lập trình của các em tăng lên đáng kể, đặc biệt là hứng thú học tập, các em định hướng rõ hơn về một dạng bài tập, tạo cho các em tâm lý không sợ khó khi gặp bài tập dạng này.

- Nhiều học sinh đã biết vận dụng dạng toán quy hoạch động giải quyết các bài toán về dãy con và mở rộng tìm hiểu các mô hình khác của quy hoạch động, một số em có thể tự tìm được lời giải được một số bài toán khác khó hơn và trong các kì thi học sinh giỏi vừa qua các em đã giành được nhiều kết quả tốt. Điều đó cho thấy hiệu quả của cách rèn luyện kỹ năng lập trình bằng việc mở rộng bài toán cơ bản.

- Từ bài toán cơ bản chúng ta chỉ cần thay đổi các tham số hoặc thêm vào các tham số khác nhau thì được các bài toán khác nhau qua đó các em hiểu rõ hơn quy hoạch động cũng như độ phức tạp của thuật toán.

Như vậy, việc sử dụng phương pháp quy hoạch động thông qua việc mở rộng bài toán dãy con, giúp học sinh phát triển năng lực tư duy, kỹ năng về lập trình. Đồng thời nâng cao việc yêu thích học tin học đối với một bộ phận học sinh, trong đó có một số em có khả năng tìm hiểu sâu hơn về các dạng bài toán lập trình.

2. Kiến nghị, đề xuất :

Sau khi thực hiện đề tài này chúng tôi xin mạnh dạn đưa ra một số đề xuất như sau :

- Để học sinh thực sự hiểu rõ phương pháp quy hoạch động trong lập trình về bài toán dãy con đối với học sinh lớp 11 thì cần tăng cường hơn nữa lượng thời gian trong phân phối chương trình để học sinh rèn luyện các dạng bài tập.

- Giáo viên cần khai thác các bài toán cơ bản trong sách giáo khoa và từ đó mở

rộng ra các bài toán tương tự khác.

- Với đối tượng học sinh khá giỏi thì có thể khai thác sâu hơn một số bài toán khó và các mô hình khác về quy hoạch động

3. Kết Luận

Với cách làm này thì phát triển được năng lực, kỹ năng lập trình của học sinh, từ đó giúp các em hứng thú để tiếp tục tìm hiểu và giải quyết các bài toán khác, các thầy, cô có thể áp dụng cách làm này với nhiều dạng bài tập khác nhau để thấy được hiệu quả. Chúng tôi hy vọng các thầy cô có thể tạo được niềm đam mê cho học sinh và tạo ra những học sinh có tư duy kỹ năng lập trình đó cũng là mong muốn của chúng tôi khi viết sáng kiến kinh nghiệm này.

Trên đây là sáng kiến nghiệm của bản thân chúng tôi trong quá trình giảng dạy cũng như bồi dưỡng học sinh khá giỏi. Mặc dù đã rất cố gắng nhưng không thể tránh khỏi những sai sót, rất mong được sự góp ý kiến, phê bình, phản hồi của các đồng nghiệp (ĐT: Hoàng Xuân Thắng: 0976124889 – Nguyễn Đình Lợi: 0919529318).

Chúng tôi chân thành cảm ơn Ban giám hiệu trường THPT Lê Viết Thuật và các thầy, cô trong tổ Toán – Tin đã động viên giúp đỡ, đã có những ý kiến đóng góp sâu sắc để chúng tôi hoàn thành bài viết này.

Chúng tôi xin chân thành cảm ơn!

Nhóm tác giả

Hoàng Xuân Thắng

Nguyễn Đình Lợi