Mục lục

CHUYÊN ĐỀ QUY HOẠCH ĐỘNG

A. PHẦN MỞ ĐẦU

1. Lý do chọn đề tài

Quy hoạch động là một thuật toán quen thuộc của các học sinh chuyên Tin. Bài toán sử dụng thuật toán này cũng xuất hiện khá nhiều ở các cuộc thi cấp quốc gia, quốc tế. Tuy nhiên, quy hoạch động có khá nhiều dạng bài không dễ, đòi hỏi học sinh có tư duy khá, sáng tạo tốt mới có thể giải được. Với cơ sở là những bài toán "kinh điển" trong quy hoạch động, người ra đề đã đưa thêm vào những giả thuyết khiến học sinh phải phân tích thật kỹ lưỡng, cẩn thận mới có thể đưa ra một thuật toán tối ưu. Vì vậy, để có thể giải nhuần nhuyễn một bài toán sử dụng quy hoạch động, học sinh phải cần nhiều thời gian giải nhiều dạng bài tập khác nhau. Với những lý do nêu trên, tôi lựa chọn chuyên đề "Quy hoạch động" để viết.

2. Mục đích của đề tài

Chuyên đề đưa ra nhiều dạng bài tập khác nhau, giúp người đọc có được những phân tích mới mẻ về dạng bài tập Quy hoạch động.

B. PHẦN NỘI DUNG

1. Phương pháp:

Để giải quyết một bài toán bằng phương pháp quy hoạch động, chúng ta cần tiến hành những công việc sau:

- Tìm nghiệm của các bài toán nhỏ nhất.
- Tìm ra công thức xây dựng nghiệm của bài toán con thông qua nghiệm của các bài toán con cỡ nhỏ hơn.
- Tạo ra một bảng lưu giữ các nghiệm của các bài toán con. Sau đó tính nghiệm của các bài toán con theo công thức đã tìm ra và lưu vào bảng.
- Từ các bài toán con đã giải để tìm nghiệm của bài toán.

2. Các dạng bài toán cơ bản của quy hoạch động:

2.1 Dãy con đơn điệu tăng dài nhất.

Bài toán: cho dãy số nguyên A = $a_1, a_2, \dots, a_n$. ($n \le 1000$, $-10000 \le a_i \le 10000$). Một dãy con của A là một cách chọn ra trong A một số phần tử giữ nguyên thứ tự. Yêu cầu tìm dãy con đơn điệu tăng của A có độ dài lớn nhất bắt đầu từ a_1.

Ý tưởng: Dãy con đơn điệu tăng dài nhất bắt đầu từ a_i sẽ được thành lập bằng cách lấy a_i ghép vào đầu một trong số những dãy con đơn điệu tăng dài nhất bắt đầu tại vị trí a_j mà $a_i < a_j$ ($i+1 \le j \le n+1$)

2.2 Dãy con chung dài nhất (LCS – Longest common subsequence)

Bài toán: cho hai số nguyên dương M, N ($0 < m,n \le 3000$) và hai dãy số nguyên $A_1, A_2, \dots, A_n$ và $B_1, B_2, \dots, B_n$. Tìm dãy dài nhất C là dãy con chung dài nhất của hai dãy A và B, nhận được từ A bằng cách xóa đi một số số hạng và cũng nhận được từ B bằng cách xóa đi một số số hạng.

Ý tưởng: Cần xây dựng mảng F[i][j] là độ dài của dãy con chung dài nhất của hai dãy A[1..i] và B[1..j], ta có thể tính được:

F[i][j] = max (F[i][j-1], F[i-1][j], F[i-1][j-1] + x)

Với x=0 nếu $A[i] \ne B[j]$

x=1 nếu A[i]=B[j]

2.3 Bài toán cái túi (Knapsack 1)

Bài toán: có N đồ vật, được đánh số từ 1, 2, … , N. Mỗi đồ vật I ($1 \le I \le N$) có sức nặng w_i và giá trị v_i. Một người có thể chọn một vài đồ vật trong N đồ vật và bỏ vào trong một cái túi. Sức chứa lớn nhất của túi là W. Hỏi người đó có thể chọn những đồ vật nào để được tổng giá trị lớn nhất?

Ràng buộc:

- $1 \le N \le 100$
- $1 \le W \le 10^5$
- $1 \le w_i \le W$
- $1 \le v_i \le 10^9$

Ý tưởng: Gọi F[i][j] là giá trị lớn nhất có thể có bằng cách chọn trong các gói 1, 2, … , I với giới hạn trọng lượng j, ta có thể tính được:

$F[i][j] = \max (F[i-1][j], F[i-1][j-w_i] + v_i)$

2.4 Bài toán cái túi (Knapsack 2)

Bài toán: có N đồ vật, được đánh số từ 1, 2, … , N. Mỗi đồ vật I ($1 \le I \le N$) có sức nặng w_i và giá trị v_i. Một người có thể chọn một vài đồ vật trong N đồ vật và bỏ vào trong một cái túi. Sức chứa lớn nhất của túi là W. Hỏi người đó có thể chọn những đồ vật nào để được tổng giá trị lớn nhất?

Ràng buộc:

- $1 \le N \le 100$
- $1 \le W \le 10^9$
- $1 \le w_i \le W$
- $1 \le v_i \le 10^3$

Ý tưởng: Gọi F[j] là tổng khối lượng tối thiểu của các đồ vật để có được giá trị j. Ta có thể tính được:

F[j]= min (F[j], F[j-v[i]]+w[i]), với $1 \le i \le n$

Kết quả bài toán sẽ là giá trị F[j] đầu tiên nhỏ hơn hoặc bằng W khi ta duyệt trong đoạn từ tổng giá trị của các đồ vật về 1.

2.5 Viên sỏi (K-Stones)

Bài toán: cho một tập A = $a_1, a_2, \ldots, a_n$ gồm N số nguyên dương. Bạn T và bạn J sẽ chơi một trò chơi sau. Ban đầu có một đống sỏi gồm K viên sỏi. Hai người chơi lần lượt thực hiện như sau, bắt đầu với T: chọn một số x trong tập A và lấy đi x viên sỏi trong đống sỏi. Người thua cuộc là người không thể chơi tiếp được nữa. Giả sử cách chơi hai người đều tối ưu, hãy xác định người chiến thắng.

Ràng buộc:

- $1 \le N \le 100$
- $1 \le K \le 10^5$
- $1 \le a_1, a_2, \ldots, a_n \le K$

Ý tưởng: Gọi F[i] là người chiến thắng khi bốc i viên sỏi, quy định 0 là J và 1 là T. Ta có thể tính được: khởi tạo ban đầu F[i] = 0 $1 \le i \le k$

F[i] = 1 nếu ($i \ge a[j]$ và F[i-a[j]]=0) với $1 \le j \le n$

2.6 Đồng xu (Coins)

Bài toán: có N đồng xu được đánh số từ 1, … , N. Khi đồng xu thứ I được tung lên, xác suất để xảy ra mặt ngửa là p_i và xác suất để xảy ra mặt sấp là $1 - p_i$. Bạn T thực hiện tung hết N đồng xu, hãy tìm xác suất để mặt ngửa xuất hiện nhiều hơn mặt sấp.

Ràng buộc:

- $1 \leq N \leq 2999$
- p_i là số thực và có hai chữ số thập phân.
- $0 < p_i < 1$

Ý tưởng: Gọi F[i][j] là xác suất xảy ra j mặt trên của i đồng xu ($j \leq i$). Ta có thể tính được:

F[i][0] = F[i-1][0]*(1-p[i]); với $1 \leq i \leq n$

F[i][j] = F[i-1][j-1]*p[i]+F[i-1][j]*(1-p[i]); với $1 \leq i \leq n$, $1 \leq j \leq i$.

2.7 Chia kẹo (Candies)

Bài toán: có N đứa bé, được đánh số từ 1, … , N. Chúng quyết định chia sẽ K viên kẹo cho nhau. Quy định rằng, đứa trẻ thứ i sẽ chỉ nhận được số kẹo trong khoảng từ 0 đến a_i viên kẹo, và không còn dư lại viên kẹo nào. Tìm số cách chia kẹo cho các bé, kết quả chia lấy dư cho $10^9 + 7$, hai cách chia được gọi là khác nhau nếu tồn tại một đứa bé nhận được số kẹo khác nhau.

Ràng buộc:

- $1 \leq N \leq 100$
- $0 \leq K \leq 10^5$
- $0 \leq a_i \leq K$

Ý tưởng: Gọi F[i][j] là số cách chia j viên kẹo cho i đứa bé. Ta có thể tính được như sau:

F[i][j] = F[i][j-1] + F[i-1][j] - F[i-1][j-a[i]-1]; nếu j-a[i]>0

F[i][j] = F[i-1][j] + F[i][j-1]; nếu j-a[i] $\leq$ 0

2.8 Slimes

Bài toán: cho N Slimes được xếp thành một hàng. Slime thứ i tính từ trái qua có kích thước là a_i. Bạn T cố gắng ghép các Slime lại với nhau thành một Slime lớn hơn. T sẽ thực hiện các thao tác sau cho đến khi còn một Slime: chọn hai Slime, ghép chúng lại với nhau thành một Slime mới, Slime mới này có kích thước là x + y, trong đó x và y là kích thước của hai slime ban đầu. Giá trị x + y được gọi là chi phí phát sinh. Vị trí của các Slime sau khi được ghép lại là không thay đổi. Hãy tìm cách ghép có tổng chi phí phát sinh là nhỏ nhất.

Ràng buộc:

- $2 \leq N \leq 400$
- $1 \leq a_i \leq 10^9$

Ý tưởng: Gọi F[i][j] là chi phí ghép slime i đến slime j. Tạo một mảng S[i] là tổng các kích thước của các Slime từ 1 đến i. Ta có thể tính được như sau:

F[i][j] = min(F[i][j], (F[l][k]+F[k+1][j]+(S[j]-S[i-1]))); với $n \geq i \geq 1$, $i+1 \leq j \leq n$, $i \leq k \leq j$

3. Một số bài tập:

3.1 Dãy con chung hoán vị: Nguồn https://lqdoj.edu.vn/problem/lcsperm

Bài toán: Cho số nguyên dương n. Một dãy a được gọi là siêu hoán vị hệ k nếu mỗi số nguyên 1, 2, …, n xuất hiện đúng k lần. Ví dụ, n = 3 và k = 2 thì (1,2,2,3,1,3) là một siêu hoán vị.

Cho a và b là hai siêu hoán vị hệ k. Hãy tìm dãy con chung dài nhất của chúng. Lưu ý rằng:

- Dãy C được gọi là dãy con của a nếu c có thể nhận được bằng cách bỏ đi vài số trong a và giữ nguyên thứ tự các số còn lại. Ví dụ, (1,3) là dãy con của (1,2,3) và (3,2) thì không phải.
- Dãy c được gọi là dãy con chung của a và b, nếu c là dãy con của a và c là dãy con của b.

Input:

- Dòng đầu tiên chứa hai số nguyên dương n, k ($1 \leq n \leq 10^5$, $1 \leq k \leq 5$).
- Dòng thứ hai chứa n*k số nguyên $a_1, a_2, \dots, a_{nk}$ ($1 \leq a_i \leq n$) và mỗi số từ 1 đến n xuất hiện đúng k lần.
- Dòng thứ ba chứa n*k số nguyên $b_1, b_2, \dots, b_{nk}$ ($1 \leq b_i \leq n$) và mỗi số từ 1 đến n xuất hiện đúng k lần.

Output:

- In ra một số nguyên dương là độ dài của dãy con chung dài nhất.

lcsperm.inp	lcsperm.out
3 2 1 2 3 1 2 3 2 1 3 3 2 1	3

Subtask:

- Có 20% test có n≤1000
- Có 40% test có k=1
- 40% test còn lại không có điều kiện gì thêm.

Đề xuất giải thuật:

- Subtask 1: n<=1000 và k<=5, vậy dãy này có 5000 phần tử. Ta có thể dễ dàng nhận ra đây là bài toán xâu con chung dài nhất. Công thức như sau:

f[i][j]=max(f[i-1][j],f[i][j-1],f[i-1][j-1]+1 nếu a[i]=b[j])

Độ phức tạp: O(n^2)

Subtask 2 và 3: Xét 2 dãy sau:

a: 1 2 3 1 2 3

b: 2 1 3 3 2 1

Mỗi phần tử của dãy a có thể kết hợp với các phần tử của dãy b như sau:

các vị trí của dãy a	1	2	3	4	5	6
các vị trí của dãy b	6,2	5,1	4,3	6,2	5,1	4,3

Gọi f(i,j) là độ dài dãy con chung dài nhất. Ở đây ta xét:

f(i,j): nếu chọn i ta sẽ xác định được j, tức là nếu i được chọn thì phần tử i của dãy a sẽ được ghép với phần tử j của dãy b.

Do vậy bản chất phần tử j sẽ không phải là một tham số độc lập mà là tham số bị phụ thuộc. Khi đó với f(i,j), ta xét trong đoạn 1→ i và 1→ j nhưng cả phần tử i và phần tử j đều được chọn, vậy việc tìm dãy con chung dài nhất chính là việc tìm dãy con tăng dài nhất ở dãy các vị trí có thể được chọn của dãy b (6,2,5,1,4,3,6,2,5,1,4,3).

Các vị trí tìm được của dãy b sắp giảm dần vì một số ở dãy trái chỉ kết hợp nhiều nhất một số dãy phải, ta giảm dần để khi tìm dãy con tăng không có trường hợp một số dãy trái kết hợp với nhiều hơn một số ở dãy phải.

Rõ ràng, nếu tìm được dãy con tăng trên dãy b thì ta có thể hoàn toàn nối các phần tử tìm được với các phần tử tương ứng của dãy a mà không bị cắt nhau.

Độ phức tạp: O(nlogn).

Code:

```
#include "bits/stdc++.h"
using namespace std;
#define name "lcsperm"
#define MAXN 3000005
long long n,k,a[MAXN],res[MAXN],cnt=0;
vector<long long> pos[MAXN];
vector<long long> save[MAXN];
long long dp[MAXN],b[MAXN],m;

void solve()
{
    cin>>n>>k;
    for(int i=1;i<=n*k;i++)
    {
        long long tmp;
        cin>>tmp;
        pos[tmp].push_back(i);
    }
    for(int i=1;i<=n*k;i++)
    {
        cin>>a[i];
        save[a[i]].push_back(pos[a[i]].back());
        pos[a[i]].pop_back();
    }
    for(int i=1;i<=n*k;i++)
        for(auto r:save[a[i]])
            res[++cnt]=r;
    for(int i=1;i<=cnt;i++)
    {
        dp[i]=lower_bound(b+1,b+m+1,res[i])-b;
        m=max(m,dp[i]);
        b[dp[i]]=res[i];
    }
    cout<<m;
}

int main()
{
    freopen(name".inp", "r", stdin);
    freopen(name".out", "w", stdout);
    solve();
}
```

Test:
https://drive.google.com/drive/folders/17HQmoHYWynv4eVZHbGsztbMFs7cXawtf?usp=sharing

3.2 Tặng quà (Nguồn: https://oj.vnoi.info/problem/voi21_noel)

Bài toán: Noel sắp tới, Ông Già Tuyết đã chuẩn bị 2n món quà dành cho các bạn nhỏ. Các món quà có màu sắc đôi một khác nhau và có mã màu từ 1 đến 2n. Khi cho các món quà vào túi, Ông đã đưa ra các món quà vào theo một thứ tự mà nếu lấy ra, các món quà sẽ có mã màu lần lượt là $c_1, c_2, \ldots, c_{2n}$ (dãy $c_1, c_2, \ldots, c_{2n}$ là một hoán vị của 1, 2, …, 2n).

Ông Già Tuyết dự định tặng quà cho m ($m \le n$) bạn nhỏ, mỗi bạn sẽ được nhận hai món quà sau hai lượt tặng. Các bạn nhỏ đứng thành một hàng và Ông sẽ đi từ đầu hàng đến cuối hàng để lần lượt tặng quà cho từng bạn. Khi đứng trước một bạn để tặng quà, Ông lần lượt lấy từng món quà ra cho tới khi lựa chọn được một món quà phù hợp và tặng bạn nhỏ, các món quà không được lựa chọn sẽ cất đi và không được dùng để tặng quà. Khi bạn nhỏ thứ m ở cuối hàng đã được nhận quà, Ông sẽ di chuyển về đầu hàng để tặng quà lượt thứ hai tương tự như lượt thứ nhất.

Ông được biết, các bạn nhỏ mong muốn nhận được hai món quà mà chênh lệch mã màu của hai món quà đó không vượt quá d. Với mong muốn mang lại nhiều niềm vui cho các bạn nhỏ, Ông quyết định việc tặng quà sẽ phải bảo đảm tất cả các bạn nhỏ đều nhận được hai món quà mà chênh lệch mã màu không vượt quá d.

Một cách hình thức, gọi m là số lượng bạn nhỏ được quà, Ông cần chọn ra dãy 2m chỉ số $1 \le i_1 < i_2 < \ldots < i_m < i_{(m+1)} < \ldots < i_{2m} \le 2n$ sao cho $\left|c_{i_k} - c_{i_{m+k}}\right| \le d$ với mọi $1 \le k \le m$.

Ông Già Tuyết biết rằng, có thể không tồn tại cách chọn được 2m chỉ số thỏa mãn, điều đó cũng có nghĩa là không thể tặng quà như mong muốn cho cả m bạn nhỏ. Do đó, với một số nguyên d và thứ tự các món quà lấy ra có mã màu lần lượt là $c_1, c_2, \ldots, c_{2m}$, Ông muốn tính số lượng nhiều nhất các bạn nhỏ có thể tặng quà.

Yêu cầu: Hãy giúp Ông Già Tuyết tính số lượng nhiều nhất các bạn nhỏ mà Ông có thể tặng quà đáp ứng điều kiện nêu trên.

Dữ liệu: vào từ đầu vào chuẩn:

- Dòng thứ nhất chứa hai số nguyên dương n và d (d≤5);
- Dòng thứ hai chứa 2n số nguyên dương $c_1, c_2, \ldots, c_{2n}$ là mã màu của các món quà lần lượt được lấy ra.

Các số trên cùng một dòng cách nhau bởi dấu cách.

Kết quả: Ghi ra đầu ra chuẩn một số nguyên duy nhất là số lượng nhiều nhất các bạn nhỏ mà Ông Già Tuyết có thể tặng quà.

Ràng buộc:

- Có 40% số test ứng với 40% số điểm của bài thỏa mãn: n≤10;
- 40% số test khác ứng với 40% số điểm của bài thỏa mãn: n≤100;
- 20% số test khác ứng với 20% số điểm của bài thỏa mãn: n≤1000;

Ví dụ:

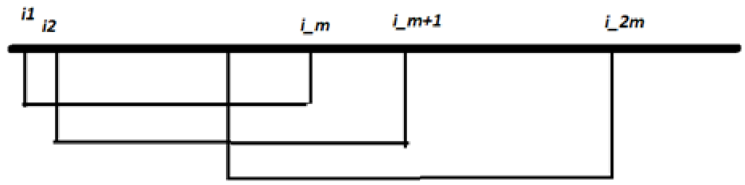
noel.inp	noel.out
3 1 1 5 6 3 4 2	2

Đề xuất giải thuật:

Tóm tắt đề bài: Cho dãy số $c_1, c_2, \ldots, c_{2n}$ và số nguyên d≤5.

Cần tìm ra các chỉ số $i_1 < i_2 < \cdots < i_{2m}$ sao cho $|c_{i_k} - c_{i_{m+k}}| \le$ d với mọi 1≤k≤m. In ra số m lớn nhất thỏa mản điều kiện trên.

- Subtask 1(n≤10): sinh nhị phân liệt kê tất cả các dãy số rồi tìm dãy số có độ dài lớn nhất thỏa điều kiện. Độ phức tạp: O(2^n)
- Subtask 2 (n≤100): duyệt trong đoạn từ 1 → 2n tìm điểm cắt để chia dãy thành hai dãy bên trái và bên phải. Sau đó ta tìm xâu con chung dài nhất của hai dãy.



Lưu ý rằng ở đây ta tìm độ chênh lệch của hai phần tử. Tức là hai chỉ số c_i và c_j thỏa mãn $|c_i - c_j| \le d$.

Công thức quy hoạch động: gọi dãy bên trái là a, dãy bên phải là b.

f[i][j]=f[i-1][j-1]+1 (nếu |a[i]-b[j]|≤d).

f[i][j]=max(f[i-1][j],f[i][j-1]) (trường hợp ngược lại)

Độ phức tạp: $O(n^3)$

- Subtask 3(n≤1000): sử dụng ý tưởng của bài toán dãy con chung hoán vị.

Khi tìm dãy con chung của 2 dãy với chênh lệch không quá d, d≤5, ta nhận thấy một phần tử ở trên sẽ ghép với không quá 2d+1 phần tử ở dưới, như vậy ta có thể chuyển tải về bài toán tìm dãy con tăng.

Độ phức tạp: O(n.n.(2d+1).log(n.(2d+1)).

Code:

```
#include<bits/stdc++.h>
using namespace std;
#define name "noel"
long long a[2005],n,d,dd[2005]={0},b[30005]={0},de,sm[30005],kq=0;
void enter(long long k)
{
    dd[a[k]]=k;
    for(long long i=k+1;i<=2*n;i++)
    {
        long long v[15];long long dem=0;long long u=a[i];
        for(long long j=1;j<=d;j++)
        {
            if(u-j>0&&dd[u-j]!=0)
            {
                dem++;v[dem]=dd[u-j];
            }
            if(u+j<=2*n&&dd[u+j]!=0)
            {
                dem++;v[dem]=dd[u+j];
            }
        }
        sort(v+1,v+dem+1,greater<long long>());
        for(long long j=1;j<=dem;j++)
        {
            de++;
```

```
            b[de]=v[j];
        }
    }
}
long long tknp(long long l,long long r,long long x)
{
    if(l==r) return l;
    if(r-l==1)
    {
        if(sm[r]>x) return r;
        else return l;
    }
    long long mid=(l+r)/2;
    if(x>=sm[mid]) return tknp(l,mid-1,x);
    else return tknp(mid,r,x);
}
void process(long long k)
{
    de=0;
    memset(sm,0,sizeof(sm));
    enter(k);
    long long m=1;
    for(long long i=de;i>=1;i--)
    {
        long long u=tknp(1,m,b[i]);
        sm[u+1]=max(sm[u+1],b[i]);
        m=max(m,u+1);
    }
    kq=max(kq,m-1);
}
void solve()
{
    sm[1]=2005;
    for(long long i=1;i<2*n;i++)
    process(i);
    cout<<kq;
}
int main()
{
    freopen(name".inp", "r", stdin);
    freopen(name".out", "w", stdout);
    cin>>n>>d;
    for(long long i=1;i<=2*n;i++)
    cin>>a[i];
    solve();
}
```

Test:

https://drive.google.com/drive/folders/17HQmoHYWynv4eVZHbGsztbMFs7cXawtf?usp=sharing

3.3 Tặng quà (Nguồn: https://lqdoj.edu.vn/problem/bonus2021)

Bài toán: Trong buổi giao lưu các thí sinh tham gia kỳ thi, thầy Nhỏ đã chuẩn bị 2n món quà dành cho các thí sinh đạt giải. Khi cho các món quà vào túi, thầy Nhỏ đã đưa các món quà vào theo một thứ tự mà nếu lấy ra, các món quà sẽ có mã màu lần lượt là c_1, c_2, …, c_{2n}.

Có m ($m \le n$) thí sinh đạt giải, mỗi bạn sẽ được nhận hai món quà sau hai lượt tặng. Các thí sinh đứng thành một hàng và thầy Nhỏ sẽ đi từ đầu hàng đến cuối hàng để lần lượt tặng quà cho từng bạn. Khi đứng trước một bạn để tặng quà, thầy Nhỏ lần lượt lấy từng món quà ra cho tới khi lựa chọn được một món quà phù hợp để tặng, các món quà không được lựa chọn sẽ cất đi và không được dùng để tặng quà. Khi bạn thứ m ở cuối hàng đã được nhận quà, thầy Nhỏ tiếp tục tặng quà lượt thứ hai tương tự như lượt thứ nhất nhưng bắt đầu từ bạn thứ m lùi về đầu hàng. Thầy Nhỏ được biết, các thí sinh mong muốn nhận được hai món quà mà chênh lệch mã màu của hai món quà đó không vượt quá d nên Thầy quyết định việc tặng quà sẽ phải bảo đảm tất cả thí sinh đều nhận được hai món quà mà chênh lệch mã màu không vượt quá d.

Một cách hình thức, gọi m là số lượng thí sinh được tặng quà, thầy Nhỏ cần chọn ra dãy 2m chỉ số $1 \le i_1 < i_2 < \ldots < i_m < i_{(m+1)} < \ldots < i_{2m} \le 2n$ sao cho $|c_{i_k} - c_{i_{2m+1-k}}| \le d$ với mọi $1 \le k \le m$.

Thầy Nhỏ biết rằng, có thể không tồn tại cách chọn được 2m chỉ số thỏa mãn, điều đó cũng có nghĩa là không thể tặng quà như mong muốn cho cả m thí sinh. Do đó, với một số nguyên d và thứ tự các món quà lấy ra có mã màu lần lượt là c_1, c_2, …, c_{2n}, thầy Nhỏ muốn tính số lượng nhiều nhất các bạn có thể tặng quà.

Yêu cầu: Hãy giúp thầy Nhỏ tính số lượng nhiều nhất các thí sinh mà thầy Nhỏ có thể tặng quà đáp ứng điều kiện nêu trên.

Dữ liệu: vào từ thiết bị vào chuẩn có khuôn dạng:

- Dòng thứ nhất chứa hai số nguyên dương n và d ($d \le 10^6$);
- Dòng thứ hai chứa 2n số nguyên dương c_1, c_2, …, c_{2n} là mã màu của các món quà lần lượt được lấy ra, các số không vượt quá 10^6.

Kết quả: Ghi ra thiết bị ra chuẩn một số nguyên duy nhất là số lượng nhiều nhất các thí sinh mà thầy Nhỏ có thể tặng quà.

Ràng buộc:

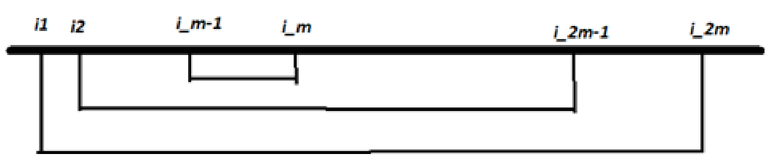
- Có 40% số test ứng với 40% số điểm của bài thỏa mãn: n≤10;
- 30% số test khác ứng với 30% số điểm của bài thỏa mãn: n≤150;
- 30% số test khác ứng với 30% số điểm của bài thỏa mãn: n≤2000;

Ví dụ:

bonus.inp	bonus.out
3 1 1 4 5 3 2 6	2

Đề xuất giải thuật:

Đây là bài toán tìm xâu con chung của 2 xâu. Điểm khác của bài toán này với bài toán Tặng quà (VOI 2021) đó là không phải so sánh 2 điểm đầu của 2 dãy mà là so sánh điểm đầu của dãy này và điểm cuối của dãy kia.



Gọi i là điểm đầu và j là điểm cuối, bài toán sẽ được quy về bài toán tìm dãy đối xứng dài nhất.

f(i,j) = max (f(i+1,j), f(i,j-1), f(i+1,j-1) + 1 nếu $|a_i - a_j| \le d$)

Độ phức tạp O(n^2)

Code:

```
#include "bits/stdc++.h"
using namespace std;
#define name "bonus"
#define MAXN 5005
long long n,d,a[MAXN],dp[MAXN][MAXN];

long long opt(int l,int r)
{
    if(l>=r) return 0;
    long long &ans=dp[l][r];
```

```
    if(dp[l][r]!=-1) return dp[l][r];
    if(max(a[l],a[r])-min(a[l],a[r])>d) return ans=max(opt(l+1,r),opt(l,r-1));
    else return ans=opt(l+1,r-1)+1;
}

void solve()
{
    cin>>n>>d;
    for(int i=1;i<=2*n;i++) cin>>a[i];
    memset(dp,-1,sizeof(dp));
    long long res=opt(1,2*n);
    cout<<res;
}

int main()
{
    freopen(name".inp", "r", stdin);
    freopen(name".out", "w", stdout);
    solve();
}
```

Test:

https://drive.google.com/drive/folders/17HQmoHYWynv4eVZHbGsztbMFs7cXawtf?usp=sharing

3.4 Dãy con (Nguồn: Thầy Đỗ Đức Đông)

Bài toán: Dãy Fibonacci là dãy vô hạn các số tự nhiên bắt đầu bằng hai phần tử 0 và 1, các phần tử tiếp theo được thiết lập theo quy tắc, mỗi phần tử sau bằng tổng hai phần tử trước nó.

Các phần tử đầu tiên của dãy Fibonacci là : 0,1,1,2,3,5,8,…

Với một dãy số nguyên không âm $a_1, a_2, \ldots, a_n$ và hai số nguyên P, Q, hãy đếm số số cặp chỉ số l, r thỏa mãn $P \le r - l + 1 \le Q$ và đoạn con S (l,r) là dãy có tính Fibonacci.

Dữ liệu: vào từ thiết bị nhập chuẩn theo khuôn dạng:

- Dòng đầu gồm ba số nguyên dương n, P, Q ($1 \le P \le Q \le n$);
- Dòng thứ hai gồm n số nguyên không âm $a_1, a_2, \ldots, a_n$ ($a_i \le 10^6$).

Kết quả: ghi ra thiết bị ra chuẩn một dòng chứa một số nguyên là số lượng cặp chỉ số l, r đếm được.

Ràng buộc:

- Có 30% số lượng test ứng với 30% số điểm có n = 1;
- Có 20% số lượng test khác ứng với 20% số điểm có $n \le 100$;

- Có 30% số lượng test khác ứng với 30% số điểm có $n \le 10^4$;
- Có 20% số lượng test còn lại ứng với 20% số điểm có $n \le 10^5$;

Ví dụ:

Fibseq.inp	Fibseq.out
1 1 1 13	1
5 1 3 1 1 8 1 1	7

Đề xuất giải thuật:

- Subtask 1: Với n=1, dãy số này chỉ có một số thì chỉ có một cặp số (1,1), ta chỉ cần kiểm tra xem số này có phải là số fibonacci hay không, nếu có in ra 1, ngược lại in ra 0.
- Subtask 2 và Subtask 3: Gọi mảng prefix là mảng cộng dồn từ 1 → n

  Với mỗi i duyệt từ Q → N, vì độ dài đoạn nằm trong đoạn P →Q nên với mỗi i ta xét các prefix trong đoạn i – Q +1 → i– P +1, duyệt j từ i – Q +1 → i– P +1 nếu prefix[i]- prefix[j] là số fibonacci thì kết quả tăng 1.

  Độ phức tạp là $O(n^2)$
- Subtask 4: Nhận xét dãy có 10^5 phần tử với giá trị lớn nhất của các phần tử là 10^6 thì tổng dãy lớn nhất là 10^{11}, mà số fibonacci thứ 60 là 956722026041, lớn hơn 10^{11}. Do đó ta có thể lật ngược bài toán.

  Xét với mỗi số fibonacci thì ta đếm có bao nhiêu đoạn có tổng chính bằng số fibonacci này có độ dài ≥ P và ≤ Q.

  Độ phức tạp: O(60*n)

Code:

```
#include "bits/stdc++.h"
using namespace std;
#define MAXN 500005
#define name "fibseq"
typedef long long ll;
int n,u,v,limit = 0;
ll f[MAXN], a[MAXN], ps[MAXN];
vector<int> pos[MAXN];
int BS(ll x)
{
    int l = 0, r = limit;
```

```
    while (l <= r) {
        int mid = (l + r) / 2;
        if (ps[mid] <= x)
            l = mid + 1;
        else r = mid - 1;
    }
    return (l - 1);
}
int BS_pos1(int x, int v)
{
    int l = 1, r = pos[v].size();
    while (l <= r) {
        int mid = (l + r) / 2;
        if (pos[v][mid - 1] < x)
            l = mid + 1;
        else r = mid - 1;
    }
    return (l - 1);
}
int BS_pos2(int x, int v)
{
    int l = 1, r = pos[v].size();
    while (l <= r) {
        int mid = (l + r) / 2;
        if (pos[v][mid - 1] <= x)
            l = mid + 1;
        else r = mid - 1;
    }
    return (l - 1);
}
void solve()
{
    cin >> n >> u >> v;
    for (int i = 1; i <= n; i++)
    {
        int x;
        cin >> x;
        a[i] = a[i - 1] + 1ll * x;
        ps[i] = a[i];
    }
    ps[0] = 0;
    for (int i = 1; i <= n; i++) if (ps[i] != ps[i - 1]) ps[++limit] = ps[i];
    f[0] = 0;
    f[1] = 1;
    int cnt;
    for (int i = 2; i; i++)
    {
        f[i] = f[i - 2] + f[i - 1];
        if (f[i] > a[n])
        {
            cnt = i;
            break;
        }
    }
    ll res = 0;
    pos[0].push_back(1);
    for (int i = 1; i <= n; i++)
```

```
    {
        int value = BS(a[i]);
        for (int j = 0; j <= cnt; j++)
        {
            if (f[j] > a[i]) break;
            if (j == 2) continue;
            ll needvalue = a[i] - f[j];
            int loc = BS(needvalue);
            if (loc < 0 || loc > limit || ps[loc] != needvalue) continue;
            int taken = BS_pos2(max(i - u + 1, loc),loc) - BS_pos1(max(i - v +
1, 1), loc);
            res = res + 1ll * taken;
        }
        pos[value].push_back(i + 1);
    }
    cout << res;
}
int main()
{
    freopen(name".inp", "r", stdin);
    freopen(name".out", "w", stdout);
    solve();
}
```

Test:
https://drive.google.com/drive/folders/17HQmoHYWynv4eVZHbGsztbMFs7cXawtf?usp=sharing

3.5 Chia kẹo (Nguồn: Thầy Đỗ Đức Đông)

Bài toán: Ba anh em An, Bình, Cường có n gói kẹo, gói thứ I có a_i cái kẹo. Cả ba quyết định chia n gói kẹo thành ba phần theo nguyên tắc:

- Không bóc các gói kẹo;
- Chia các gói kẹo thành ba phần, gọi $A \geq B \geq C$ là số kẹo tương ứng của ba phần, khi đó An sẽ nhận phần có A cái kẹo, Bình sẽ nhận phần có B cái kẹo, Cường sẽ nhận phần có C cái kẹo.

Cách chia để cả ba anh em vui nhất là cách chia có giá trị (A – C) nhỏ nhất.

Yêu cầu: Cho $a_1, a_2, \ldots, a_n$ là số kẹo của n gói kẹo, hãy tìm cách chia thỏa mãn để (A – C) đạt giá trị nhỏ nhất.

Dữ liệu: Vào từ thiết bị vào chuẩn:

- Dòng đầu chứa số nguyên n;
- Dòng thứ hai chứa n số nguyên dương $a_1, a_2, \ldots, a_n$ ($a_i \leq 10^9$) là số kẹo của n gói kẹo.

Kết quả: Ghi ra thiết bị ra chuẩn một dòng chứa một số là giá trị (A – C) nhỏ nhất tìm được.

Ràng buộc:

- Có 15% số lượng test ứng với 15% số điểm có n = 3;
- Có 35% số lượng test khác ứng với 35% số điểm có n ≤ 10;
- Có 25% số lượng test khác ứng với 25% số điểm có n ≤ 20;
- Có 25% số lượng test còn lại ứng với 25% số điểm có n ≤ 100 và tổng số kẹo trong n gói không vượt quá 1000.

Ví dụ:

sweets.inp	sweets.out
4 5 5 3 4	2

Đề xuất giải thuật:

- Subtask 1 (n=3): kết quả là: max(a[1],a[2],a[3]) – min(a[1],a[2],a[3])
- Subtask 2 (n≤10):

Sinh tam phân 1,2,3 với 1 là người A, 2 là người B, 3 là người C. Tính tổng A, B, C rồi lấy res=min(res,max(A,B,C)-min(A,B,C))

Độ phức tạp: $O(3^n)$

- Subtask 3 (n≤20): ta dùng bitmask

Chia ra làm 2 tập, 1 tập là B, tập còn lại là A+C. Sau đó ta tách A, C.

- Subtask 4 (n≤100 và tổng gói kẹo không vượt quá 1000):

Ý tưởng dùng quy hoạch trạng thái kết hợp bfs để đánh dấu lại các trạng thái có thể xảy ra.

Gọi f[i][x][y][z]: khi xét đến món đồ thứ i, thì A có x kẹo, B có y, C có z kẹo.

Thực hiện giảm chiều quy hoạch động, vì x+y+z=n ➔ z=n-x-y tức là z phụ thuộc vào x và y nên ta chỉ cần tìm được x,y là sẽ tìm được z ➔ f[i][x][y]

Khởi tạo f[0][0][0]=true.

Tạo một queue gồm 3 tham số là <i,x,y> như hàm f. Ta dùng queue để loang ra các trường hợp xét các trạng thái có thể xảy ra. Thuật toán thực hiện như sau:

+ pos=q.front().i, A=q.front().x, B=q.front().y q.pop()

+ nếu pos≥n thì ta bỏ qua, do ta đã xét hết n gói của trạng thái này.

+ từ f[i][A][B] ta có thể đi đến:

- f[i+1][A+a[i+1]][B] //chọn gói kẹo a[i+1] cho thằng A
- f[i+1][A][B+a[i+1]] //chọn gói kẹo a[i+1] cho thằng B
- f[i+1][A][B] //chọn gói kẹo a[i+1] cho thằng C

+ nếu f[i+1][A+a[i+1]][B]=false (tức chưa duyệt đến trạng thái này) và A+a[i+1]≤sum (tức là số kẹo của A không được vượt quá tổng số kẹo) thì ta đánh dấu f[i+1][A+a[i+1]][B]=true và q.push({i+1,A+a[i+1],B})

Lặp lại các thao tác trên cho đến khi queue rỗng.

Với mỗi f[n][i][j]=true (1≤i≤sum,1≤j≤sum) tức là xét hết n gói kẹo A được i gói, B được j gói và C được sum-i-j kết quả là min(max(i,j,sum-i-j)-min(i,j,sum-i-j))

Code:

```
#include<bits/stdc++.h>
using namespace std;
#define name "sweets"
long long n, a[105], b[25], f[1100005]={0}, su=0, s1,s2,s3,kq=1000000000000,
dp[105][1005]={0}, a2[105],A,B,C,dd[1005]={0}, dp1[105][1005]={0};
void enter()
{
    cin>>n;
    for(long long i=1;i<=n;i++)
    {
        cin>>a[i];su+=a[i];
    }
}
long long lt(long long i)
{
    if(i==0) return 1;
    if(i==1) return 2;
    long long x=lt(i/2);
    if(i%2==0) return x*x;
    return x*x*2;
}
void sol(long long t)
{
    long long sum=0;
    for(long long i=0;i<n;i++)
    {
        long long k=t^lt(i);
```

```
            if(k<t) sum+=a[i+1];
        }
        if(sum<=su/3) f[t]=sum;
        else
        {
            for(long long i=0;i<n;i++)
            {
                long long z=lt(i);
                f[t]=max(f[t],f[t^z]);
            }
            s1=f[t];s2=su-sum;s3=sum-f[t];
            long long z1=min(s1,min(s2,s3));
            long long z2=max(s1,max(s2,s3));
            kq=min(kq,z2-z1);
        }
    }
    void solve()
    {
        long long mw=0;
        dp[0][0]=1;
        for(long long i=1;i<=n;i++)
        {
            for(long long j=a[i];j<=su/3;j++)
            {
                for(long long z=0;z<i;z++)
                {
                    if(dp[z][j-a[i]]==1)
                    {
                        dp[i][j]=1;
                        mw=max(mw,j);break;
                    }
                }
            }
        }
        A=mw;
        while(mw!=0)
        {
            for(long long i=1;i<=n;i++)
            {
                if(dp[i][mw]==1)
                {
                    dd[i]=1;mw-=a[i];break;
                }
            }
        }
        long long n2=0;
        for(long long i=1;i<=n;i++)
        {
            if(dd[i]==0)
            {
                n2++;a2[n2]=a[i];
            }
        }
        su-=A;
        B=0;
        dp1[0][0]=1;
        for(long long i=1;i<=n2;i++)
```

```
    {
        for(long long j=a2[i];j<=su/2;j++)
        {
            for(long long z=0;z<i;z++)
            {
                if(dp1[z][j-a2[i]]==1)
                {
                    dp1[i][j]=1;
                    B=max(B,j);break;
                }
            }
        }
    }
    C=su-B;
    kq=C-A;
}
int main()
{
    freopen(name".inp", "r", stdin);
    freopen(name".out", "w", stdout);
    enter();
    if(n<=20)
        for(long long i=0;i<pow(2,n);i++)
            sol(i);
    else
    solve();
    cout<<kq;
}
```

Test:
https://drive.google.com/drive/folders/17HQmoHYWynv4eVZHbGsztbMFs7cXawtf?usp=sharing

3.6 Cắt xâu (Nguồn: Thầy Đỗ Đức Đông)

Bài toán: Hai xâu X và Y cùng độ dài n (chỉ gồm các kí tự 'a' đến 'z') được gọi là tương đồng bậc k nếu các kí tự tương ứng của hai xâu không cách nhau quá k vị trí trong bảng chữ. Cụ thể, với mọi i ($1 \le i \le n$), ta có, kí tự X_i (là kí tự thứ I của xâu X) và Y_i (là kí tự thứ I của xâu Y) có thứ tự chênh lệch không quá k (thứ tự của 'a' là 1, thứ tự của 'b' là 2, …, thứ tự của 'z' là 26). Trường hợp k bằng 0 thì xâu X bằng xâu Y.

Yêu cầu: cho hai xâu S_1 và S_2 độ dài bằng nhau (chỉ gồm các kí tự 'a' đến 'z'), hãy xác định số cách cắt S_2 thành ba xâu khác rỗng, mà từ đó có thể ghép thành xâu S mà xâu S tương đồng bậc k với xâu S_1. Hai cách cắt được gọi là khác nhau nếu tồn tại một vị trí cắt khác nhau.

Dữ liệu: vào từ thiết bị nhập chuẩn gồm ba dòng:

- Dòng thứ nhất chứa số nguyên k;
- Dòng thứ hai chứa xâu S_1;

- Dòng thứ ba chứa xâu S_2.

Kết quả: Ghi ra thiết bị chuẩn một dòng chứa một số nguyên là số cách cắt thỏa mãn.

Ràng buộc:

- Có 30% số test ứng với 30% số điểm có k = 0 và độ dài hai xâu không vượt quá 300;
- Có 40% số test khác ứng với 40% số điểm có k=0 và độ dài hai xâu không vượt quá 2000;
- Có 30% số test còn lại ứng với 30% số điểm có k≤0 và độ dài hai xâu không vượt quá 2000;

Ví dụ:

cutstrg.inp	cutstrg.out
0 beast betas	1
1 aaaaa bbbbb	6

Đề xuất giải thuật:

Tóm tắt đề: cho số K và 2 xâu S và T có độ dài bằng nhau, đếm xem có bao nhiêu cách chia xâu T thành ba phần sau đó ghép ba phần này lại thành một xâu. Gọi xâu sau khi ghép là T' thì T' phải đồng bậc k với xâu S (đồng bậc k nghĩa là với mọi i thì S[i]-T'[i]≤k ở đây a có bậc là 1, b có bậc là 2,…, z có bậc là 26)

Ý tưởng trâu: Ta sẽ viết một hàm kiểm tra xem hai xâu có đồng bậc K với nhau hay không. Duyệt từ 0→S.size()-1, nếu s[i] - T'[j]>k (s[i] ở đây là bậc của kí tự s[i]) thì return false, nếu duyệt hết S.size() thì return true.

Duyệt n^2 lần tìm ra hai điểm cắt ở xâu T để tạo ra 3 phần, khi cắt xâu T ra làm ba phần là A, B, C thì ta có 6 cách ghép là ABC,ACB,BAC,BCA,CAB,CBA, kiểm tra lần lượt 6 xâu này có đồng bậc K với xâu S hay không, nếu có thì tăng kết quả lên 1.

Độ phức tạp $O(n^3)$.

Tối ưu hóa độ phức tạp: ta nhận xét ở đây với các cặp xâu thì ta phải liên tục trả lời truy vấn trong đoạn từ i→j ở xâu S, có bằng đoạn từ u→v trong xâu T hay không?

Thế thì có cách nào trả lời các truy vấn này trong O(1) không?

Ta sẽ tạo trước 1 mảng f[i][j] là xâu con chung liên tiếp dài nhất kết thúc ở i trong xâu S và ở j trong xâu T là bao nhiêu, thế nhưng xâu con chung ở đây không phải là s[i]==t[j] mà là $|s[i] - t[j]| \leq$ k vì nó đồng bậc k nên ta dùng LCS mang tính tương đồng. Nếu $|s[i] - t[j]| \leq$ k thỏa thì f[i][j]=f[i-1][j-1]+1 còn không thì ta ngắt đi f[i][j]=0.

Ví dụ: ta muốn kiểm tra đoạn S[p→q] có đồng bậc 1 T[u→v] hay không //lưu ý độ dài đoạn bằng nhau.

Lúc này nếu f[q][v]≥q-p+1 thì có tương đồng, ngược lại thì không.

Code:

```
#include "bits/stdc++.h"
using namespace std;
#define name "cutstrg"
#define MAXN 2005
long long n,k,res=0;
string s,t;
long long dp[MAXN][MAXN];
int toInt(char x)
{
    return (int)(x-96);
}
void solve()
{
    cin>>k>>s>>t;
    n=s.size();
    s=" "+s; t=" "+t;

    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)
        {
            if(abs(toInt(s[i])-toInt(t[j]))<=k) dp[i][j]=dp[i-1][j-1]+1;
            else dp[i][j]=0;
        }

    for(int i=1;i<=n-2;i++)
        for(int j=i+1;j<=n-1;j++)
        {
            //ABC
            if(dp[i][i]>=i&&dp[j][j]>=j-i&&dp[n][n]>=n-j)
            {
                res++;
                continue;
            }
```

```
            //ACB
            if(dp[i][i]>=i&&dp[n-j+i][n]>=n-j&&dp[n][j]>=j-i)
            {
                res++;
                continue;
            }
            //BAC
            if(dp[j-i][j]>=j-i&&dp[j][i]>=i&&dp[n][n]>=n-j)
            {
                res++;
                continue;
            }
            //BCA
            if(dp[j-i][j]>=j-i&&dp[n-i][n]>=n-j&&dp[n][i]>=i)
            {
                res++;
                continue;
            }
            //CAB
            if(dp[n-j][n]>=n-j&&dp[n+i-j][i]>=i&&dp[n][j]>=j-i)
            {
                res++;
                continue;
            }
            //CBA
            if(dp[n-j][n]>=n-j&&dp[n-i][j]>=j-i&&dp[n][i]>=i)
            {
                res++;
                continue;
            }
        }
    cout<<res;
}
int main()
{
    freopen(name".inp", "r", stdin);
    freopen(name".out", "w", stdout);
    solve();
}
```

Test:

https://drive.google.com/drive/folders/17HQmoHYWynv4eVZHbGsztbMFs7cXawtf?usp=sharing

3.7 Mật khẩu (Nguồn https://lqdoj.edu.vn/problem/password03)

Bài toán: Do dịch Covid-19, hai bạn Hồng và Chi không được đi học và gặp nhau nhưng hai bạn vẫn thường xuyên nhắn tin cho nhau. Một lần, Hồng muốn gửi mật khẩu tham gia lớp học online do Chi nhưng không muốn em Phúc tò mò và biết được. Theo ý tưởng giấu tin trong ảnh, Hồng quyết định sẽ giấu mật khẩu vào trong đoạn văn bản gửi cho Chi. Cụ thể, với một văn bản mà Hồng gửi cho Chi được biểu diễn bằng xâu ký tự T

= $t_1t_2\ldots t_n$ (gồm n ký tự, mỗi ký tự thuộc ‘a’ đến ‘z’) và dãy số nguyên $a_1, a_2, \ldots, a_m$ ($1 \le a_1 < a_2 < \ldots < a_m \le n$) là dãy số mà hai bạn đã thống nhất thì mật khẩu là một xâu P = $t_{a1}t_{a2}\ldots t_{am}$, là xâu độ dài m nhận được bằng cách ghép lần lượt các ký tự ở các vị trí $a_1, a_2, \ldots, a_m$. Ví dụ, T=’missyouuu’ và dãy số $a_1 = 2$, $a_2 = 3$, $a_3=5$, $a_4=6$, $a_5=8$ thì mật khẩu là P=”isyou”.

Hồng nhanh chóng nhận ra rằng, với một xâu T và mật khẩu P sẽ tồn tại nhiều dãy số để xác định mật khẩu. Ví dụ, một dãy số khác $a_1 = 2$, $a_2 = 4$, $a_3=5$, $a_4=6$, $a_5=7$ cũng xác định được mật khẩu P=”isyou” trong xâu T=’missyouuu’.

Trong quá trình gửi, xâu T sẽ được mã hóa theo phương thức RLE (Run Length Encoding). Nghĩa là, một xâu T chỉ gồm các ký tự ‘a’ đến ‘z’ được mã hóa thành xâu TE (chỉ gồm các ký tự ‘a’ đến ‘z’ và ký tự ‘0’ đến ‘9’) bằng cách đi từ trái sang phải, mã hóa dãy các ký tự liên tiếp giống nhau trong T thành ký tự đại diện và số lượng.

Ví dụ, xâu T = ‘missyouuuuuuuuuu’ thì TE = ‘m1i1s2y1o1u10’.

Yêu cầu: Cho xâu TE (là mã hóa của xâu T) và xâu mật khẩu P, gọi R là số lượng dãy số khác nhau có thể xác định được mật khẩu P trong xâu T. Hãy tính R chia dư cho $10^9 + 7$.

Dữ liệu vào

- Dòng đầu chứa hai số nguyên dương n, m;
- Dòng thứ hai chứa một xâu là mã hóa của xâu T
- Dòng thứ ba chứa một xâu là xâu P.

Kết quả

- Ghi ra một số nguyên duy nhất là số R chia dư cho $10^9 + 7$

Ví dụ:

Password.inp	Password.out
9 5 m1i1s2y1o1u3 isyou	6

Ràng buộc:

- Có 20% số lượng test ứng với 20% số điểm thỏa mãn điều kiện: n≤20, m=1 ;

- Có 20% số lượng test khác ứng với 20% số điểm thỏa mãn điều kiện: $n \leq 20$, $m<n$;
- Có 20% số lượng test ứng khác với 20% số điểm thỏa mãn điều kiện: $n \leq 10^5$, $m=3$;
- Có 20% số lượng test khác ứng với 20% số điểm thỏa mãn điều kiện: $n \leq 10^5$, $m \leq 30$;
- Có 20% số lượng test còn lại ứng với 20% số điểm thỏa mãn điều kiện: : $n \leq 10^9$, $m \leq 30$;và xâu mã hóa của xâu T có độ dài không vượt quá 10^5

Đề xuất giải thuật:

Chia xâu T thành các khối có ký tự giống nhau, gọi f[i][j] là số cách tạo được j ký tự đầu tiên của xâu P bằng I khối đầu tiên của xâu T. Ta có f[0][0] =1.

Từ trạng thái f[i][j] ta tiến hành cập nhật cho các trạng thái quy hoạch động khác theo ý tưởng như sau: giả sử khố i+1 sẽ phủ tiếp k ký tự tiếp theo, điều kiện để có được điều này là các ký tự từ vị trí j+1 tới j+k của P phải bằng ký tự của khối i+1 của xâu T. Khi đó ta có f[i+1][j+k] += f[i][j]*C(k,w[i+1]) với w[i+1] là số ký tự của khối i+1 và C(k,w) là tổ hợp chập k của w phần tử.

Code:

```
#include "bits/stdc++.h"
using namespace std;
#define name "password"
#define MAXN 200005
#define MOD 1000000007
long long n,m;
string s,t;
vector<pair<char,long long>> a;
long long dp[50][MAXN];
void ChangeToVector()
{
    a.push_back({'@',0});

    for(int i=0;i<s.size();i++)
    {
        a.push_back({s[i],0});
        i++;
        while(isdigit(s[i]))
        {
            a.back().second=a.back().second*10+(int)(s[i]-48);
            i++;
        }
        i--;
    }
}
long long binPow(long long a,long long b)
{
```

```
    a%=MOD;
    long long res=1;
    while(b>0)
    {
        if(b&1) res=(res*a)%MOD;
        b>>=1;
        a=(a*a)%MOD;
    }
    return res;
}
void solve()
{
    cin>>n>>m>>s>>t;
    ChangeToVector();
    n=a.size()-1;
    t=" "+t;
    for(int i=0;i<=n;i++) dp[0][i]=1;
    for(int i=1;i<=m;i++)
    {
        for(int j=1;j<=n;j++)
        {
            dp[i][j]=dp[i][j-1];
            long long r=i,cnt=0,cbn=1;
            while(t[r]==a[j].first && cnt<a[j].second)
            {
                r--; cnt++;
                cbn=(cbn*(binPow(cnt,MOD-2)*(a[j].second-cnt+1)%MOD))%MOD;
                dp[i][j]=(dp[i][j]+dp[r][j-1]*cbn)%MOD;
            }
        }
    }
    cout<<dp[m][n];
}
int main()
{
    freopen(name".inp", "r", stdin);
    freopen(name".out", "w", stdout);
    solve();
}
```

Test:

https://drive.google.com/drive/folders/17HQmoHYWynv4eVZHbGsztbMFs7cXawtf?usp=sharing

3.8 RLELCS (Nguồn Thầy Đỗ Đức Đông)

Bài toán: Xét xâu S độ dài không vượt quá 10^{18} chỉ gồm các ký tự ‘a’ đến ‘z’ được mã hóa thành xâu S_E (chỉ gồm các ký tự ‘a’ đến ‘z’ và ký tự ‘0’ đến ‘9’) như sau: Đi từ trái qua phải, mã hóa dãy các ký tự liên tiếp bằng nhau trong S thành ký tự đại diện và số lượng. Độ dài các xâu mã hóa không vượt quá 1000.

Ví dụ, xâu S=aaabbbbaaaaaaaaaaz thì S_E = a3b4a10z1

Giải quyết hai vấn đề sau:

1) Cho xâu X được mã hóa thành X_E và xâu Y được mã hóa thành Y_E, hãy tìm xâu Z là xâu con chung dài nhất của X và Y. Đưa ra độ dài của xâu Z.

   Ví dụ X_E = a1b10, Y_E = b3c9b4 thì Z_E=b7

2) Cho xâu X được mã hóa thành X_E, xâu Y được mã hóa thành Y_E, tìm Z là xâu con liên tiếp của cả X và Y. Đưa ra Z_E là mã hóa của Z.

   Ví dụ: X_E=a10b2c3, Y_E=a5b2c10 thì Z_E=a5b2c3

Input

- Dòng 1: chứa xâu X_E là mã hóa của X.
- Dòng 2: chứa xâu Y_E là mã hóa của Y.

Output

- Dòng 1: ghi độ dài xâu con chung dài nhất của X và Y;
- Dòng 2: ghi độ dài xâu con liên tiếp của X và Y;

Ví dụ:

LCRLE.INP	LCRLE.OUT
a1b10 b3c9b4	7 4

Đề xuất giải thuật:

Đây là bài toán xâu nén, gồm 2 thành phần là số lượng kí tự và kí tự, ta có thể dùng pair<long long,char> lưu để tách xâu này ra.

Ví dụ h3v2 ta sẽ tách ra đưa vào dãy a có CTDL pair <long long,char> là a[0].first=3,a[0].second=h,a[1].first=2,a[1].second=v

- Vấn đề 1: xâu con chung dài nhất (không liên tiếp)

Ta gọi f[i][j] là kết quả tối ưu khi xét i kí tự đầu tiên của xâu X và j kí tự đầu tiên của xâu Y. Nếu ai=bj thì f[i][j]=f[i'][j']+min(leng_1,leng_2)

Với i' là vị trí xa i nhất về trước sao cho đoạn [i', i] có kí tự bằng kí tự X[i] và có độ dài là leng_1. Tương tự với j'.

Ta có dùng quy hoạch động đảo nhãn để duyệt i', tự cập nhật j'.

Độ phức tạp: $O(n^3)$

- Vấn đề 2: xâu con chung liên tiếp dài nhất

Gọi f[i][j] là kết quả tối ưu khi xét 2 xâu từ 1->i của xâu X và từ 1->j của xâu Y giải quyết + Với xâu thường: f[i][j] = max (f[i-1][j-1]+1 (nếu a[i]=b[j]), 0 (nếu a[i]!=b[j]))

+ Với xâu nén: Xét mỗi cặp (i, j) a[i].second=b[j].second ta duyệt ngược về đầu tìm phần tử xa nhất mà bằng a[i].second và b[j].second rồi kết quả là lấy min của 2 đoạn đó cộng với f[i-1][j-1]

Độ phức tạp: $O(n^2)$

Code:

```
#include "bits/stdc++.h"
using namespace std;
#define MAXN 1005
#define name "LCRLE"
typedef long long ll;
typedef pair<int, int> ii;
ii a[MAXN], b[MAXN];
int  Dp[MAXN][MAXN] , Dpcont[MAXN][MAXN];
vector<int> Pos[30];
int number=0,lena=0,lenb=0;
string s;
void solve()
{
    cin >> s;
    for(int i=1;i<=s.size();i++)
        if(s[i-1]>='a'&&s[i-1]<='z')
        {
            if(number)
            {
                a[lena].second=number;
                a[++lena].first=s[i-1]-'a'+1;
                number=0;
            }
            else a[++lena].first=s[i-1]-'a'+1;
        }
        else number=number*10+(s[i-1]-'0');
    a[lena].second=number;
    number=0;
    cin>>s;
    for(int i=1;i<=s.size();i++)
        if(s[i-1]>='a'&&s[i-1]<='z')
        {
            if(number)
            {
                b[lenb].second=number;
                b[++lenb].first=s[i-1]-'a'+1;
                number=0;
            }
            else b[++lenb].first=s[i-1]-'a'+1;
```

```
        }
        else number=number*10+(s[i-1]-'0');
    b[lenb].second=number;
    number=0;
    for(int i=1;i<=lenb;i++) Pos[b[i].first].push_back(i);
    for(int i=1;i<=lena;i++)
        for(int j=1;j<=lenb;j++) {
            if(a[i].first!=b[j].first)
            {
                Dp[i][j]=max(Dp[i-1][j],Dp[i][j-1]);
                continue;
            }
            int value=a[i].first,pointer;
            int suma=0,sumb=b[j].second;
            for(int k=0;k<Pos[value].size();k++)
                if(Pos[value][k]==j)
                {
                    pointer=k;
                    break;
                }
            for(int k=i;k;k--)
            {
                int curvalue=a[k].first;
                if(curvalue==value) suma+=a[k].second;
                while(pointer&&sumb<suma)
                {
                    Dp[i][j]=max(Dp[i][j],Dp[k-1][Pos[value][pointer]-
1]+sumb);
                    pointer--;
                    sumb+=b[Pos[value][pointer]].second;
                }
                Dp[i][j]=max(Dp[i][j],Dp[k-1][Pos[value][pointer]-
1]+min(suma,sumb));
            }
            Dp[i][j]=max({Dp[i][j],Dp[i-1][j],Dp[i][j-1]});
        }
    cout<<Dp[lena][lenb]<<'\n';
    for(int i=1;i<=lena;i++)
        for(int j=1;j<=lenb;j++)
        {
            if(a[i].first!=b[j].first) continue;
            if(a[i].second==b[j].second)
                Dpcont[i][j]=Dpcont[i-1][j-1]+a[i].second;
            else Dpcont[i][j]=min(a[i].second,b[j].second);
        }
    int maxrescont=0;
    for(int i=0;i<=lena;i++)
        for(int j=0;j<=lenb;j++)
        {
            If (a[i+1].first==b[j+1].first)
maxrescont=max(maxrescont,Dpcont[i][j]+min(a[i+1].second,b[j+1].second));
            else maxrescont=max(maxrescont,Dpcont[i][j]);
        }
    cout<<maxrescont;
}
int main()
{
```

```
    freopen(name".inp", "r", stdin);
    freopen(name".out", "w", stdout);
    solve();
}
```

Test:

https://drive.google.com/drive/folders/17HQmoHYWynv4eVZHbGsztbMFs7cXawtf?usp=sharing

3.9 RLESUBSTR (Nguồn Thầy Đỗ Đức Đông)

Bài toán: Xét xâu S độ dài không vượt quá 10^{18} chỉ gồm các ký tự 'a' đến 'z' được mã hóa thành xâu S_E (chỉ gồm các ký tự 'a' đến 'z' và ký tự '0' đến '9') như sau: Đi từ trái qua phải, mã hóa dãy các ký tự liên tiếp bằng nhau trong S thành ký tự đại diện và số lượng. Độ dài các xâu mã hóa không vượt quá 1000.

Ví dụ, xâu S=aaabbbbaaaaaaaaaaz thì S_E = a3b4a10z1

Yêu cầu: Cho xâu S được mã hóa thành S_E, đếm số lượng xâu khác nhau nhận được từ S bằng cách giữ nguyên hoặc xóa đi một số ký tự (đưa ra kết quả mod 111539786)

Ví dụ: S_E = a10 thì số lượng các xâu khác nhau nhận được từ S là 10.

Input

- dòng đầu chứa số T là số bộ dữ liệu;
- T dòng sau, mỗi dòng chứa xâu S_E là mã hóa của S.

Output

- Gồm T dòng, mỗi dòng là kết quả tương ứng với dữ liệu vào

Ví dụ:

RLESTR.INP	RLESTR.OUT
2 a10 b1a1b1	10 6

Đề xuất giải thuật:

Bài này ta cũng thực hiện dùng pair<char,long long> để tách xâu nén ra.

Ví dụ: a3c2b4f7 thì ta tách như sau:

a[i].first : a c b f

a[i].second : 3 2 4 7

Dùng vector<int> pos[MAXN], với pos[i] ta lưu lại vị trí của các kí tự i có trong pair trên 1≤i≤26 (ta ép kiểu kí tự i sang số nguyên)

Dùng một set để lưu lại các kí tự khác nhau có ở pair.

Khi đó với mỗi i ta tìm lần lượt kí tự có trong dãy có vị trí lớn hơn i đầu tiên, gọi vị trí đó là flag nếu như kí tự đó không có thì ta bỏ qua, nếu mà kí tự đó là a[i].first mà ta đang xét thì dp[flag]+=dp[i], ngược lại dp[flag]+=dp[i]*a[i].second.

Kết quả là tổng của các dp[i]*a[i].second với (i: 1→n)

Code:

```
#include<bits/stdc++.h>
using namespace std;
#define pcl pair<char,long long>
#define ve vector<pcl>
#define oo 111539786
#define name "rlestr"
ve rle(string s)
{
    long long i=0;long long t=0;pcl c;ve a;
    while(i<s.size())
    {
        if(s[i]>=97&&s[i]<=122)
        {
            if(i!=0)
            {
                a.push_back(c);
                c.first=s[i];
                t=0;
            }
            else
            {
                c.first=s[i];
            }
        }
        else
        {
            t=t*10+s[i]-48;
            c.second=t%oo;
        }
        i++;
    }
    a.push_back(c);
    return a;
}
void process()
{
    long long t=0;
    long long dd[30]={0},dp[505]={0},lt[505]={0};lt[0]=1;
    ve a;
```

```
    string s;
    cin>>s;
    a=rle(s);
    for(long long i=1;i<=a.size();i++)
    {
        char l=a[i-1].first;
        long long k=dd[l-96];
        dp[i]=lt[k];
        for(long long j=k+1;j<i;j++)
        dp[i]=(dp[i]+dp[j]+oo)%oo;
        dd[l-96]=i;
        lt[i]=dp[i];
        dp[i]=(dp[i]*(a[i-1].second+oo)%oo+00)%oo;
        t=(t+dp[i])%oo;
    }
    cout<<t<<"\n";
}
int main()
{
    freopen(name".inp", "r", stdin);
    freopen(name".out", "w", stdout);
    long long t;
    cin>>t;
    for(long long i=1;i<=t;i++)
    process();
}
```
Test:

https://drive.google.com/drive/folders/17HQmoHYWynv4eVZHbGsztbMFs7cXawtf?usp=sharing

3.10 Dãy số (Nguồn Thầy Đỗ Đức Đông)

Bài toán: Cho dãy số gồm n số nguyên $a_1, a_2, \ldots, a_n$. Một đoạn con của dãy đã cho là dãy $a_i,\ldots,a_j$ $(1 \le i \le j \le n)$, dãy có độ dài (j-i+1) và có trọng số bằng tổng $(a_i+\ldots+a_j)$. Yêu cầu: Tìm hai đoạn con không có phần tử chung, mỗi đoạn có độ dài là một số chia hết cho 3 và tổng trọng số của hai đoạn con là lớn nhất.

Input:

- Dòng đầu ghi số nguyên n $(n\ge6)$;
- Dòng thứ hai ghi n số nguyên $a_1, a_2,\ldots,a_n$ $(|a_i|\le10^9)$.

Output:

- Một số là tổng trọng số của hai đoạn con tìm được.

Ví dụ:

Seq.inp	Seq.out

11 -1 3 -1 -9 -1 1 1 1 1 1 -9	5

Subtask:

- Có 30% số test có n ≤ 20;
- Có 30% số test có n ≤ 200;
- Có 20% số test khác có n ≤ 2000;
- Có 20% số test còn lại có n ≤ 200000;

Đề xuất giải thuật:

Trước khi đến với bài này thì ta phải làm được bài toán nếu chỉ cần tìm 1 đoạn có độ dài chia hết cho 3 và tổng trọng số lớn nhất.

Ta sẽ có 1 mảng f gồm 3 phần tử là f[0], f[1], f[2]

Với f[i] là phần tử có giá trị nhỏ nhất sau khi cộng dồn và chỉ số của phần tử đó chia 3 dư i.

Khởi tạo f[0]=0, f[1]=INT_MAX, f[2]=INT_MAX.

Với mỗi i: res=max(res,a[i]-f[i%3]).

f[i%3]=min(f[i%3],a[i])

Quay lại với vấn đề chính, với hai dãy ta sẽ giải quyết như sau:

- Ý tưởng trâu: tìm điểm cắt chia dãy này ra làm hai, bên trái và bên phải, tìm kết quả của dãy bên trái như bài trên (tìm một dãy mà có độ dài chia hết cho 3) và trọng số lớn nhất gọi đó là res1. Tương tự, tìm kết quả bên phải, gọi đó là res2

Như vậy: res=max(res,res1+res2)

Độ phức tạp: $O(n^2)$.

- Ý tưởng tối ưu:

Ta sẽ gọi f1[i] là kết quả của dãy từ 1→i có một đoạn độ dài chia hết cho 3 và có trọng số lớn nhất, f2[i] là kết quả của dãy từ n→i có một đoạn độ dài chia hết cho 3 và có trọng số lớn nhất. Ta cũng làm như bài trên đưa về 1 dãy.

f[0]=0, f[1]=INT_MAX, f[2]=INT_MAX, res=INT_MIN

For(i,1,n)

{

res=max(res,a[i]-f[i%3]);

f1[i]=res;

f[i%3]=min(a[i],f[i%3]);

} // mảng a là mảng cộng dồn

Thêm 1 mảng b dùng để lưu các giá trị của dãy a ban đầu theo chiều ngược lại tức là b[1]=a[n], b[2]=a[n-1],…,b[n]=a[1] //lưu ý mảng a ban đầu trước khi cộng dồn.

Sau đó ta thực hiện cộng dồn mảng b.

For(i,n,1)

{

res=max(res,b[i]-f[i%3]);

f2[i]=res;

f[i%3]=min(b[i],f[i%3]);

}

Đảo ngược mảng f2, kết quả chính là: res=max(res,f1[i]+f2[i+1])

Độ phức tạp: O(n)

Code:

```
#include "bits/stdc++.h"
using namespace std;
#define name "seq"
#define MAXN 500005
long long pref[MAXN],tmp[MAXN],suf[MAXN],res=-999999999999;
long long dp[5];
long long n,a[MAXN],b[MAXN];
void solve()
{
    cin>>n;

    for(int i=1;i<=n;i++) cin>>a[i];

    int cnt=0;
    for(int i=n;i>=1;i--) b[++cnt]=a[i];

    for(int i=1;i<=n;i++) a[i]+=a[i-1];
    for(int i=1;i<=n;i++) b[i]+=b[i-1];

    res=-999999999999; dp[0]=0; dp[1]=999999999999; dp[2]=999999999999;
```

```
    for(int i=1;i<=n;i++)
    {
        res=max(res,a[i]-dp[i%3]);
        dp[i%3]=min(dp[i%3],a[i]);
        pref[i]=res;
    }

    res=-999999999999; dp[0]=0; dp[1]=999999999999; dp[2]=999999999999;

    for(int i=1;i<=n;i++)
    {
        res=max(res,b[i]-dp[i%3]);
        dp[i%3]=min(dp[i%3],b[i]);
        tmp[i]=res;
    }
    cnt=0;
    res=-999999999999;
    for(int i=n;i>=1;i--) suf[++cnt]=tmp[i];
    for(int i=1;i<n;i++) res=max(res,pref[i]+suf[i+1]);
    cout<<res;
}

int main()
{
    freopen(name".inp", "r", stdin);
    freopen(name".out", "w", stdout);
    solve();
}
```

Test:

https://drive.google.com/drive/folders/17HQmoHYWynv4eVZHbGsztbMFs7cXawtf?usp=sharing

C. PHẦN KẾT LUẬN

Quy hoạch động là một thuật toán quen thuộc và hiệu quả trong việc giải các bài toán giúp giảm độ phức tạp thuật toán. Vì vậy, quy hoạch động được sử dụng khá nhiều trong các đề thi các cấp. Việc phân tích đúng hướng thuật toán quy hoạch động giúp các em học sinh có thể đạt điểm cao trong các kì thi.

Với thời gian nghiên cứu có hạn, chuyên đề này chắc chắn không tránh khỏi những khiếm khuyết. Tôi mong nhận được sự đóng góp ý kiến của quý thầy cô để tài liệu về chuyên đề này hoàn thiện hơn.

Tôi xin trân trọng cảm ơn !

D. TÀI LIỆU THAM KHẢO

[1]. Hồ Sĩ Đàm (2016), Tài liệu giáo khoa chuyên Tin học quyển 1, Nhà xuất bản giáo dục Việt Nam.

[2]. https://atcoder.jp/contests/dp/tasks

[3]. https://lqdoj.edu.vn/