



# LUYỆN TẬP: PHÂN TÍCH VÀ THIẾT KẾ THUẬT TOÁN

Nhóm 11: Vũ Quốc Minh Đăng - 19520448

GVMH: Nguyễn Thanh Sơn

# BÀI 1: KHỞI ĐỘNG

2

# ĐỀ BÀI

$P(n,k)$  là số phép biểu diễn các tập con có thứ tự gồm  $k$  phần tử của tập gồm  $n$  phần tử. Số  $P(n,k)$  được định nghĩa theo công thức sau:

$$P(n, k) = \begin{cases} 0 & \text{nếu } k > n \\ \frac{n!}{(n - k)!} = n \cdot (n - 1) \dots (n - k + 1) & \text{nếu } k \leq n \end{cases}$$

Cho 2 số  $n, k$ . Tính  $P(n,k)$  theo modulo  $10^9 + 7$

**Input:**

- Dòng đầu tiên đưa vào số lượng bộ test T
- T dòng tiếp theo đưa vào các bộ test, mỗi bộ gồm 2 số nguyên n, k được viết trên 1 dòng
- T, n, k thỏa mãn ràng buộc:  $1 \leq T \leq 100$ ,  $1 \leq n, k \leq 1000$

**Output:**

Đưa ra kết quả mỗi test theo từng dòng

INPUT	OUTPUT
2	
5 2	20
4 2	12

# ABSTRACTION:

Tính  $\frac{n!}{(n-k)!} \bmod 10^9 + 7$

# PATTERN RECOGNITION:

Chia để trị + Áp dụng định lí nhỏ Fermat

# ALGORITHM DESIGN:

Do đề bài là phân số nên không mod được, phải chuyển thành dạng tích của 2 số.

Áp dụng định lý nhỏ Fermat: *p là một số nguyên tố, thì với số nguyên a bất kỳ, sẽ chia hết p. Bằng kí hiệu đồng dư ta có:*

$$a^p \equiv a \pmod{p}.$$

$$\text{Hay: } a^{p-1} \equiv 1 \pmod{p}.$$

$$\Rightarrow (n - k)^{(10^9 + 6)} \equiv 1 \pmod{10^9 + 7}$$

$$\Rightarrow (n - k)^{(10^9 + 5)} \equiv \frac{1}{(n-k)!} \pmod{10^9 + 7}$$

$$\Rightarrow \frac{n!}{(n-k)!} \pmod{10^9 + 7} = n! * (n-k)!^{(10^9 + 5)}$$

Do  $(n-k)!^{(10^9 + 5)}$  là quá lớn nên phải dùng chia để trị để tính

Có:  $a^n = a * a * \dots * a$  ( $n$  số  $a$ )

$$\begin{cases} [ a * a * \dots * a (\frac{n}{2} \text{ số } a) ] * [ a * a * \dots * a (\frac{n}{2} \text{ số } a) ] * a & (\text{nếu } n \text{ lẻ}) \\ [ a * a * \dots * a (\frac{n}{2} \text{ số } a) ] * [ a * a * \dots * a (\frac{n}{2} \text{ số } a) ] & (\text{nếu } n \text{ chẵn}) \end{cases}$$

$\Rightarrow$  Ta có thể tính  $a^n$  nhờ  $a^{\frac{n}{2}}$

Tương tự ta có thể tính  $a^{\frac{n}{2}}$  nhờ  $a^{\frac{n}{4}}$

$\Rightarrow$  Sử dụng đệ quy

## Code minh họa (C++)

```
#include <bits/stdc++.h>
using namespace std;
long long mod = 1000000007;
long long f[1005];
long long mul(long long x, long long y)
{
    if(y==0) return 1;
    long long tmp = mul(x,y/2);
    tmp=(tmp*tmp)%mod;
    if(y%2==1) tmp=(tmp*x)%mod;
    return tmp%mod;
}

long long solve(long long n, long long k)
{
    long long res;
    res = f[n] % mod;
    res*=mul(f[n-k],mod-2);
    return res%mod;
}

int main()
{
    int t;
    cin>>t;
    f[0]=1;
    for(long long i=1;i<=1000;i++) f[i]=(f[i-1]*i)%mod;
    while(t--)
    {
        int n, k;
        cin>>n>>k;
        cout<<solve(n,k)<<endl;
    }
}
```

# BÀI 2: ĐỊ TRUYỀN GIỀN

9

# ĐỀ BÀI

Con người có 4 loại ADN: A, X, T, G. Giả sử đoạn gien quy định màu da của con người là một chuỗi N ADN kết hợp từ 4 loại ADN trên ( $1 \leq N \leq 20$ ). Ví dụ một đoạn gien có 8 ADN là: AATXGGGT. Các ADN trong đoạn gien được đánh số từ 1 đến N.

Đoạn gien quy định màu da của thế hệ con cũng là một đoạn N ADN kết hợp từ gien của bố và gien của mẹ. Trong đó ADN thứ i ( $1 \leq i \leq N$ ) được hình thành bằng cách lấy ADN thứ i tương ứng của gien bố hoặc gien mẹ. Ví dụ:

- Gien của bố: AATX
- Gien của mẹ: GATT
- Gien của con chỉ có thể là 4 trường hợp sau: AATX, AATT, GATX, GATT.

Cho trước gien của bố và gien của mẹ, bạn hãy viết chương trình liệt kê các khả năng có thể xảy ra của gien thế hệ con.

## Dữ liệu vào:

- Dòng thứ nhất: là số N biểu thị số ADN trong đoạn gien của bố và mẹ. ( $1 \leq N \leq 20$ )
- Dòng thứ hai: đoạn gien của bố.
- Dòng thứ ba: đoạn gien của mẹ (hai đoạn gien này có chiều dài bằng N và chỉ gồm các ký tự A, X, T G)

## Dữ liệu ra :

- Dòng đầu tiên: ghi số K là tổng số khả năng có thể xảy ra của đoạn gien thế hệ con.
- Trong K dòng tiếp theo, mỗi dòng liệt kê một khả năng của gien theo thứ tự từ điển (tức từ A → Z).

INPUT	OUTPUT
2	4
AT	AT
GX	AX
	GT
	GX

INPUT	OUTPUT
3	4
AXT	AXA
GXA	AXT
	GXA
	GXT

INPUT	OUTPUT
4	4
AATX	AATT
GATT	AATX
	GATT
	GATX

# ABSTRACTION:

Tìm tất cả trường hợp chuỗi có thể xảy ra dựa vào 2 chuỗi đầu vào

# PATTERN RECOGNITION:

Xử lý chuỗi

# ALGORITHM DESIGN:

Tạo một mảng exstr[20], mỗi phần tử là 1 vector<char> để lưu các kí tự của chuỗi cha và chuỗi mẹ  
Biến tong là số trường hợp có thể xảy ra

Chạy vòng for từ 0 đến n:

Tại vòng lặp i:

- Nếu dad[i] == mom[i], push\_back vào vector exstr[i]
- Nếu dad[i] != mom[i], push\_back lần lượt dad[i] và mom[i] vào vector exstr[i] theo thứ tự tăng dần
- Số trường hợp tổng \*= exstr[i].size

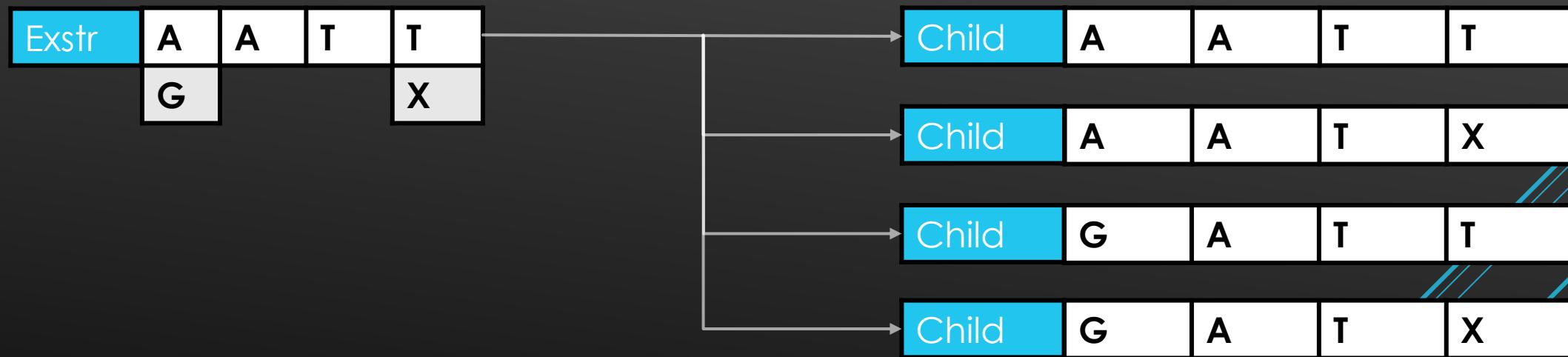
Hết vòng for => có mảng vector exstr như hình, tong = 4

Dad	A	A	T	X
Mom	G	A	T	T

Exstr	A	A	T	T
	G			X

# ALGORITHM DESIGN:

Tạo 1 hàm xuất ra kết quả (child) từ mảng vector exstr



Code minh họa  
(C++)

```
#include <bits/stdc++.h>
using namespace std;
int n, tong;
string dad, mom, child;
vector<char> exstr[20];
void dp(int x)
{
    for(int i=0;i<exstr[x].size();i++)
    {
        child[x]=exstr[x][i];
        if(x+1<n) dp(x+1);
        if(x+1==n) cout<<child<<endl;
    }
}
int main()
{
    cin>>n;
    cin>>dad>>mom;
    for(int i=1;i<=n;i++) child+="*";
    for(int i=0;i<n;i++)
    if(dad[i]==mom[i]) exstr[i].push_back(char(dad[i]));
    else exstr[i].push_back(char(min(dad[i], mom[i])));
        |exstr[i].push_back(char(max(dad[i], mom[i]))));
    tong=1;
    for(int i=0;i<n;i++) tong*=exstr[i].size();
    cout<<tong<<endl;
    dp(0);
}
```

# BÀI 3: ĐỒN CÂY

17

# ĐỀ BÀI

Hùng đang làm việc trong Công ty cao su X. Công ty có rừng cao su rất rộng, với những hàng cây cao su trồng cách đều thẳng tắp. Theo định kỳ, người ta thường phải chặt hạ cả hàng cây cao su đã hết hạn khai thác để trồng thay thế bằng hàng cây mới. Hùng phát hiện ra một bài toán tin học liên quan đến vấn đề này: Một nhóm công nhân được giao nhiệm vụ chặt hạ hàng cây gồm  $n$  cây được trồng dọc theo một đường thẳng với khoảng cách cố định giữa hai cây liên tiếp. Nếu các công nhân cưa đổ một cây, họ có thể cho nó đổ về phía bên trái hoặc bên phải dọc theo hàng cây. Một cây khi đổ có thể lật đổ cây khác bị nó rơi vào và có thể làm đổ nhiều cây khác, theo hiệu ứng lan truyền domino. Sau khi khảo sát kỹ, Hùng đã mô tả được hiệu ứng lan truyền domino như sau: Giả sử các cây trên hàng cây được đánh số từ 1 đến  $n$ , từ trái qua phải và chiều cao của cây  $i$  là  $h_i$  ( $1 \leq i \leq n$ )

- Nếu cây  $i$  đổ về bên trái thì tất cả các cây  $j$  với  $i - h_i < j < i$  cũng sẽ đổ;
  - Nếu cây  $i$  đổ về bên phải thì tất cả các cây  $j$  với  $i < j < i + h_j$  cũng sẽ đổ;
  - Mỗi cây chỉ đổ một lần về bên trái hoặc bên phải.
- Do đó bài toán đặt ra đối với Hùng là: Xác định số lượng nhỏ nhất các cây mà các công nhân cần cưa đổ để đảm bảo hạ đổ toàn bộ hàng cây.

- **Yêu cầu:** Giúp Hùng giải quyết bài toán đặt ra.
- **Input**
  - Dòng đầu tiên ghi số nguyên dương  $n$ ;
  - Dòng thứ hai chứa  $n$  số nguyên dương  $h_1, h_2, \dots, h_n$  được ghi cách nhau bởi dấu cách, mỗi số không vượt quá  $10^6$ .
- **Output**
  - Dòng đầu tiên ghi số nguyên dương  $k$  là số lượng cây mà các công nhân cần cưa đổ;
  - Dòng thứ hai ghi dãy số nguyên  $c_1, c_2, \dots, c_k$  trong đó  $|c_j| (1 \leq j \leq k)$  là dãy chỉ số của các cây theo thứ tự các công nhân phải lần lượt cưa đổ,  $c_j$  là số dương nếu cây cần cho đổ về bên phải và là số âm nếu cây cần cho đổ về bên trái.
- Nếu có nhiều cách thì chỉ cần đưa ra một cách tùy ý.

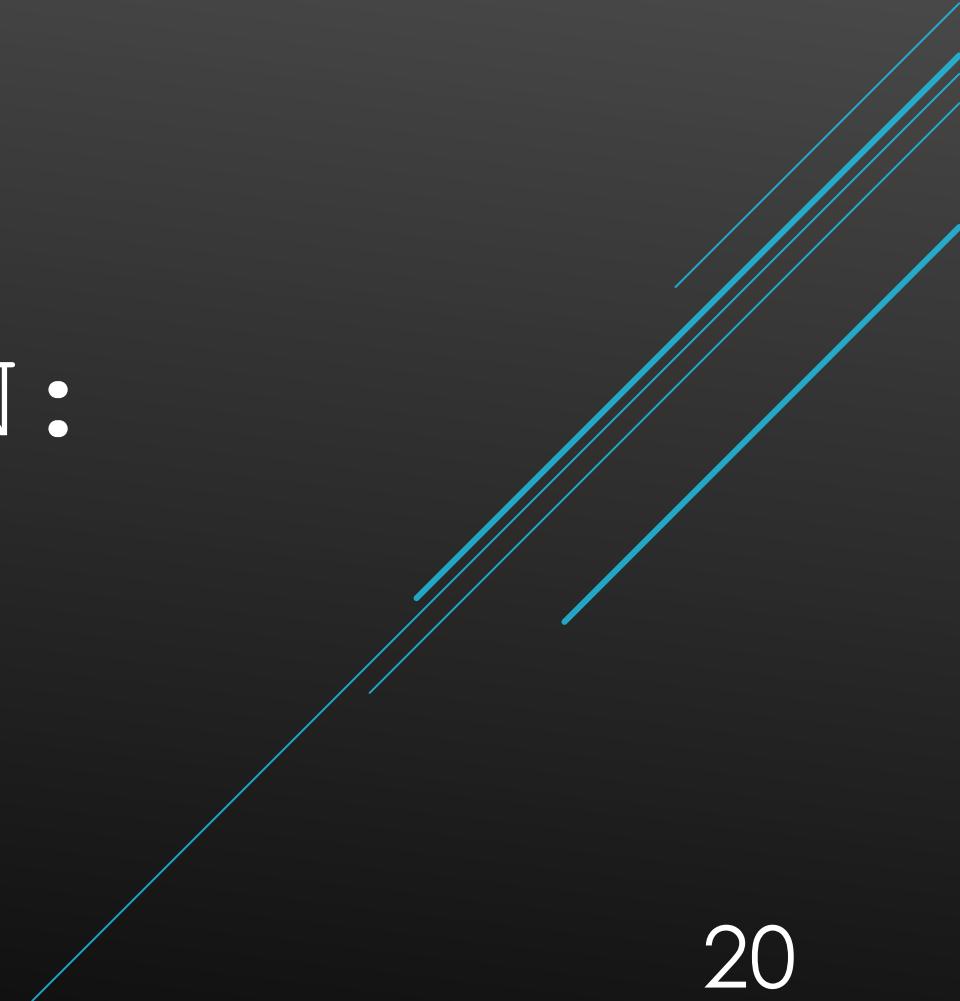
INPUT	OUTPUT
5 1 2 3 1 1	2 3 -2

# ABSTRACTION:

Tìm kiếm tuyến tính

# PATTERN RECOGNITION:

Quy hoạch động



# ALGORITHM DESIGN:

## Bước 1: Chuẩn bị

Ta sẽ xây dựng hai mảng  $L[]$  và  $R[]$ , trong đó  $L[i]$  là vị trí  $j$  nhỏ nhất mà bị cây  $i$  làm đổ nếu đáy về bên trái, tương tự với  $R$ .

$$L[i] = \min[i, \min(L[j]) \text{ với } i - h[i] < j < i]$$

Để tính  $L[]$  ta duy trì một stack chứa các chỉ số tăng dần. Trước khi thêm một cây  $i$  mới vào, các cây bị nó trực tiếp làm đổ sẽ bị pop ra, đồng thời ta cập nhật  $L[i]$ .

## Bước 2: Quy hoạch động

Gọi  $F(i)$  là số cây cần phải đỗ nhỏ nhất để các cây có chỉ số  $1 \dots i$  đều đỗ.

Để tính  $F(i)$  cần xét 2 trường hợp:

- Nếu ta đỗ cây  $i$  qua trái:  $F(i)=\min[F(j-1)+1]$  với  $L[i] \leq j \leq i$  (1)
- Nếu cây  $i$  bị đỗ qua phải bởi cây  $j$   $F(i)=\min[F(j-1)+1]$  với  $1 \leq j \leq i$  và  $R[j] \geq i$  (2)

Có thể dễ dàng tính các  $F[]$  trong  $O(N^2)$ . Có thể dùng các cấu trúc dữ liệu quản lí đoạn để giảm xuống  $O(N \log N)$ .

Ta có thể sử dụng stack để giảm độ phức tạp xuống  $O(N)$ .

$$F(i) = \min[F(j-1)+1] \text{ với } L[i] \leq j \leq i \quad (1)$$

Để xử lí (1), ta cài đặt được ngắn gọn như thế này:

$$F[L[i] - 1] = \min[F(j - 1) + 1] \text{ với } L[i] \leq j \leq i$$

$$F(i) = \min[F(j-1)+1] \text{ với } 1 \leq j \leq i \text{ và } R[j] \geq i \quad (2)$$

Để xử lí (2) ta sẽ sử dụng một *stack* để lưu các vị trí có  $R[j]$  giảm dần, đồng thời luôn duy trì sao cho giá trị ở *top* của *stack* luôn là tốt nhất. Chú ý là với  $j < i$  và  $R[j] \geq i$  thì  $R[j] \geq R[i]$ . Như vậy nếu tại mỗi bước ta *pop* các vị trí  $j$  có  $R[j] < i$  ra khỏi stack, thì sẽ luôn duy trì được tính chất của *stack* vì lúc này đảm bảo được  $R[i]$  là nhỏ hơn các  $R[j]$  đang ở trong stack, đồng thời nếu  $F(i-1)$  không tốt bằng giá trị ở đầu *stack* thì ta sẽ không đẩy  $i$  vào (để đảm bảo giá trị ở top luôn là tốt nhất).

```

#include <bits/stdc++.h>
using namespace std;

const int N = 4e6 + 6;

int n;
int a[N];
int L[N], R[N];
int dp[N], trace[N];

void initialize() {
    vector<int> S;
    for (int i = 1; i <= n; ++i) {
        L[i] = i;
        while (!S.empty() && S.back() > i - a[i])
            L[i] = min(L[i], L[S.back()]), S.pop_back();
        S.push_back(i);
    }
    S.clear();
    for (int i = n; i >= 1; --i) {
        R[i] = i;
        while (!S.empty() && S.back() < i + a[i])
            R[i] = max(R[i], R[S.back()]), S.pop_back();
        S.push_back(i);
    }
}

```

```

void solve() {
    for (int i = 1; i <= n; ++i) dp[i] = i, trace[i] = -i;
    vector<int> S;
    for (int i = 1; i <= n; ++i) {
        if (dp[i] > dp[L[i] - 1] + 1)
            dp[i] = dp[L[i] - 1] + 1, trace[i] = -(L[i]);
        while (!S.empty() && R[S.back()] < i) S.pop_back();
        if (!S.empty() && dp[i] > dp[S.back() - 1] + 1) {
            dp[i] = dp[S.back() - 1] + 1;
            trace[i] = S.back();
        }
        if (S.empty() || (dp[S.back() - 1] > dp[i - 1]))
            S.push_back(i);
    }
    cout << dp[n] << endl;
    for (int i = n; i; i = abs(trace[i]) - 1)
        cout << (trace[i] < 0 ? -i : trace[i]) << ' ';
}

int main() {
    ios::sync_with_stdio(false);
    cin >> n;
    for (int i = 1; i <= n; ++i) cin >> a[i];
    initialize();
    solve();
    return 0;
}

```

# BTVN

Trình bày bài 3 đầy đủ 4 bước + code bằng NNLT Python

Link nộp bài : 19520448@gm.uit.edu.vn

# NGUỒN

- ▶ <https://vnoi.info/>
- ▶ <http://ntucoder.net>

A black and white photograph of a person standing in a large, curved tunnel under construction. The tunnel walls are made of concrete with diagonal reinforcement bars. Large pipes run along the ceiling and walls. The person is wearing a hard hat and dark clothing, standing near the bottom left.

THANKS FOR WATCHING

27