Contents

MATRIX:

```
const int p=1e9+7;
struct matrix{
    int d[3][3];
};
matrix tich(matrix a,matrix b){
    matrix ans;
    for(int i=1;i<3;i++){
        for(int j=1;j<3;j++){
            int s=0;
            for(int k=1;k<3;k++){

s=(s%p+a.d[i][k]*b.d[k][j]%p)%p;
            }
            ans.d[i][j]=s;
        }
    }
    return ans;
}
matrix pow(matrix a,int p){
    if(p==1)return a;
    matrix ans=pow(a,p/2);
    ans=tich(ans,ans);
    if(p%2==1)ans=tich(ans,a);
    return ans;
}
```


```
const int p=1e9+7;
struct matrix{
    int d[3][3];
};
```


//fibonanci

F(0) = 1

F(1) = 1

...

F(i) = F(i-1) + F(i-2), i >= 2

$$\begin{bmatrix} F_n \\ F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F_{n-1} \\ F_{n-2} \end{bmatrix}$$

F(n) = F(n-1) + F(n-2)

F(n-1) = F(n-1)

Công thức tổng quát:

$$\begin{bmatrix} F_n \\ F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F_{n-1} \\ F_{n-2} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^2 \begin{bmatrix} F_{n-2} \\ F_{n-3} \end{bmatrix} = \cdots = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1} \begin{bmatrix} F_1 \\ F_0 \end{bmatrix}$$

//luỵthuamatran

$$B(k) = A + A^2 + \ldots + A^k$$

Nếu B chẵn

$$\begin{aligned} B(k) &= A + \ldots + A^{k/2} + A^{k/2+1} + \ldots + A^k \\ &= A + \ldots + A^{k/2} + A^{k/2}(A + \ldots + A^{k/2}) \\ &= (I + A^{k/2})(A + \ldots + A^{k/2}) \end{aligned}$$

Nếu B lẻ

$$\begin{aligned} B(k) &= A + \ldots + A^{k/2} + A^{k/2+1} + \ldots + A^{k-1} + A^k \\ &= A + \ldots + A^{k/2} + A^{k/2}(A + \ldots + A^{k/2}) + A^k \\ &= (I + A^{k/2})(A + \ldots + A^{k/2}) + A^k \end{aligned}$$

DP BITMASK

```
const int N=2000000;
int n;
int a[25][25];
int F[N];
void solve(){
    int last=(1<<n)-1;
    memset(F,0,sizeof(F));
    for(int s=0;s<=last;s++){
        int cnt=0;
        for(int i=1;i<=n;i++)if(s>>(i-
1)&1)cnt++;
        for(int i=1;i<=n;i++){
            if(s>>(i-1)&1){
                int p=s xor (1<<(i-1));

F[s]=max(F[s],F[p]+a[cnt][i]);
            }
        }
    }
    cout<<F[last]<<endl;
}
```

//hanhtrinhdulich

```
void solve(){
    reset();
    queue<ii>q;
    q.push({0,0});
    check[0][0]=1;
    dp[0][0]=0;
    while(!q.empty()){
        ii u=q.front(); q.pop();
        for(int i=1;i<=n;i++){
            int k=1<<(i-1);
            if((u.fi&k)==0){
                x=u.fi^k;

dp[x][i]=min(dp[x][i],dp[u.fi][u.se]+a[u.se
][i]);
                if(check[x][i]==0){
                    check[x][i]=1;
                    q.push({x,i});
                }
            }
        }
    }
    int ans
=*min_element(dp[last]+1,dp[last]+n+1);
    cout<<ans<<endl;
}
```

```
matrix luythua(matrix a,int k){
    if(k==1)return a;
    matrix a1=dv(a,k/2);
    matrix a2=luythua(a,k/2);
    matrix ans=tich(a1,a2);
    if(k%2==1){
        matrix b=pow(a,k);
        ans=tong(ans,b);
    }
    return ans;
}
```

//Tribonanci

Áp dụng bài 2:

Công thức ma trận:

$$\begin{bmatrix} T_n \\ T_{n-1} \\ T_{n-2} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} T_{n-1} \\ T_{n-2} \\ T_{n-3} \end{bmatrix} = \ldots = A^{n-3} \begin{bmatrix} T_3 \\ T_2 \\ T_1 \end{bmatrix}$$

Tính tổng: $F(N) = T(1) + T(2) + \ldots + T(N)$

$$T(4) = A_{1,1} \times T_3 + A_{1,2} \times T_2 + A_{1,3} \times T_1$$
$$T(5) = A^2_{1,1} \times T_3 + A^2_{1,2} \times T_2 + A^2_{1,3} \times T_1$$
$$T(N) = A^{n-3}_{1,1} \times T_3 + A^{n-3}_{1,2} \times T_2 + A^{n-3}_{1,3} \times T_1$$

Tổng quát:

$$X = A + A^2 + \ldots + A^{n-3}$$
$$F(N) = T(1) + T(2) + T(3) + X_{1,1} \times T_3 + X_{1,2} \times T_2 + X_{1,3} \times T_1$$

DSU:

```
const int N=1e5+5;
int parent[N] ,rank[N];
int find(int i){
    if(parent[i]==-1)return i;
    parent[i]=find(parent[i]);
    return parent[i];
}
void Union(int x,int y){
    int p_x=find(x);
    int p_y=find(y);
    if(p_x!=p_y)parent[p_x]=p_y;
}
```

Dùng hạng để giảm độ sâu của cây (rank)

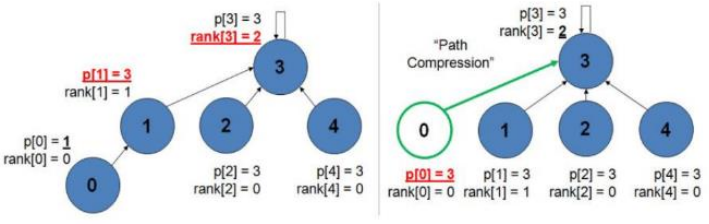


Figure 2.7: unionSet(0, 3) → findSet(0)

Một số bài quy hoạch động:

//Magical

```
int n;
int a[N];
int dp[N][200], last[N][200];
/*Main*/
void init() {
    memset(dp,0,sizeof(dp));
    memset(last,0,sizeof(last));
}
void solution()
{
    cin >> n;
    for (int i = 1; i <= n; i++)
        cin >> a[i];
    for (int i = 1; i <= n; i++)
    {
        for (int s = 1; s <= 100; s++)
        {
            if (s == a[i])
                last[i + 1][s] = i;
            else
                last[i + 1][s] =
last[i][s];
        }
    }
    for (int i = 1; i <= n; i++)
    {
        for (int s = 1; s <= 200; s++)
        {
            dp[i][s] = dp[i - 1][s];
            if (s <= a[i])
                continue;
            if (last[i][s - a[i]])
            {
                int id=last[i][s-a[i]];
                dp[i][s] = max(dp[i][s],
dp[id - 1][s] + 2);
            }
        }
    }
    int ans=0;
    for(int i=1;i<=200;i++){
        ans=max(ans,dp[n][i]);
    }
    cout<<ans<<endl;
}
```

//hinh vuong lon nhat

```
int n,m;
int a[n+1][m+1],dp[n+1][m+1];
int solve(){
```

//Union vs rank

```
void Union(int u,int v){
    u=find(u);
    v=find(v);
    if(u==v)return ;
    if(rank[u]==rank[v])rank[u]++;
    if(rank[u]>rank[v])parent[v]=u;
    else parent[u]=v;
}
```

Big Number:

```
const int base = 1000000000; const int
base_digits = 9;
struct bigint {
    vector<int> a; int sign;

    bigint() :
        sign(1) {
    }

    bigint(long long v) {
        *this = v;
    }

    bigint(const string &s) {
        read(s);
    }

    void operator=(const bigint &v) {
        sign = v.sign;
        a = v.a;
    }

    void operator=(long long v) {
        sign = 1;
        if (v < 0)
            sign = -1, v = -v;
        for (; v > 0; v = v / base)
            a.push_back(v % base);
    }

    bigint operator+(const bigint &v) const
{
        if (sign == v.sign) {
            bigint res = v;

            for (int i = 0, carry = 0; i <
(int) max(a.size(), v.a.size()) || carry;
++i) {
                if (i == (int) res.a.size())
                    res.a.push_back(0);
    int ans=0;
    memset(dp,0,sizeof(dp));
    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            cin>>a[i][j],dp[i][j]=a[i][j];
        }
    }
    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            if(a[i][j]==0)continue;
            if(a[i][j]==a[i-1][j-1]&&
a[i][j]==a[i-1][j]&&a[i][j]==a[i][j-1]){
                dp[i][j]=min(dp[i-1][j-
1],min(dp[i-1][j],dp[i][j-1]))+1;
            }
            ans=max(ans,dp[i][j]);
        }
    }
    return ans;
}
```

//C(n,k)

```
int nCr[1003][1003] = {0};
void preprocess(){
    int k;
    for(int i=0;i<1003;i++){
        nCr[i][0] = nCr[i][i] = 1;
        k = i >> 1;
        for(int j=1;j<=k;j++){
            nCr[i][j] = nCr[i][i-j]
= (nCr[i - 1][j] + nCr[i - 1][j - 1]) %
MOD;
        }
    }
}
```

// xau con chung dai nhat

```
int dp[1001][1001];
int solve(string a,string b){
    int n=a.size(),m=b.size();
    for(int i=0;i<=n;i++)dp[i][0]=0;
    for(int i=0;i<=m;i++)dp[0][i]=0;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            if(a[i-1]==b[j-1])
                dp[i][j]=dp[i-1][j-1]+1;
            else
                dp[i][j]=max(dp[i-1][j],
dp[i][j-1]);
        }
    }
    return dp[n][m];
}
```

//Truy vet xau con chung

```
string xcc(string a,string b){
    string res="";
```

```
                res.a[i] += carry + (i <
(int) a.size() ? a[i] : 0);
                carry = res.a[i] >= base;
                if (carry)
                    res.a[i] -= base;
            }
            return res;
        }
        return *this - (-v);
    }

    bigint operator-(const bigint &v) const
{
        if (sign == v.sign) {
            if (abs() >= v.abs()) {
                bigint res = *this;
                for (int i = 0, carry = 0; i
< (int) v.a.size() || carry; ++i) {
                    res.a[i] -= carry + (i <
(int) v.a.size() ? v.a[i] : 0);
                    carry = res.a[i] < 0;
                    if (carry)
                        res.a[i] += base;
                }
                res.trim();
                return res;
            }
            return -(v - *this);
        }
        return *this + (-v);
    }

    void operator*=(int v) {
        if (v < 0)
            sign = -sign, v = -v;
        for (int i = 0, carry = 0; i < (int)
a.size() || carry; ++i) {
            if (i == (int) a.size())
                a.push_back(0);
            long long cur = a[i] * (long
long) v + carry;
            carry = (int) (cur / base);
            a[i] = (int) (cur % base);
            //asm("divl %%ecx" :
"=a"(carry), "=d"(a[i]) : "A"(cur),
"c"(base));
        }
        trim();
    }

    bigint operator*(int v) const {
        bigint res = *this;
        res *= v;
        return res;
    }
```

```
    int n=a.size(),m=b.size();
    bool check[n];
    memset(check,0,sizeof(check));
    int i=n, j = m;
    while (i > 0 && j > 0){
        if (a[i - 1] == b[j - 1]){
            check[i - 1] = 1;
            i--;  j--;
        }
        else{
            if (dp[i][j] == dp[i - 1][j])i-
-;
            else j--;
        }
    }
    //
    for (int i = 0; i < n; i++){
        if (check[i] == 1)res+=a[i];
    }
    return res;
}
```

//Tìm số N nhỏ nhất thỏa mãn: A là tổng các chữ số của N, B là tổng bình phương các chữ số của N.

```
int dp[101][10001];
int mind(int a,int b){
    if(a<0||b<0||a>100||b>10001)return -1;
    if(a==0&&b==0)return 0;
    if(dp[a][b]!=-1)return dp[a][b];
    int ans=101;
    for(int i=9;i>=1;i--){
        int k=mind(a-i,b-i*i);
        if(k!=-1)ans=min(ans,k+1);
    }
    return dp[a][b]=ans;
}
void print(int a,int b){
    memset(dp,-1,sizeof(dp));
    dp[0][0]=0;
    int k=mind(a,b);
    if(k==-1||k>100)cout<<-1;
    else {
        while(a>0&&b>0){
            for(int i=1;i<=9;i++){
                if(a>=i&&b>=i*i&&
dp[a][b]==dp[a-i][b-i*i]+1){
                    cout<<i;
                    a-=i;
                    b-=i*i;
                    break;
                }
            }
        }
    }
}
```

//cai tui

```
    friend pair<bigint, bigint> divmod(const
bigint &a1, const bigint &b1) {
        int norm = base / (b1.a.back() + 1);
        bigint a = a1.abs() * norm;
        bigint b = b1.abs() * norm;
        bigint q, r;
        q.a.resize(a.a.size());

        for (int i = a.a.size() - 1; i >= 0;
i--) {
            r *= base;
            r += a.a[i];
            int s1 = r.a.size() <=
b.a.size() ? 0 : r.a[b.a.size()];
            int s2 = r.a.size() <=
b.a.size() - 1 ? 0 : r.a[b.a.size() - 1];
            int d = ((long long) base * s1 +
s2) / b.a.back();
            r -= b * d;
            while (r < 0)
                r += b, --d;
            q.a[i] = d;
        }

        q.sign = a1.sign * b1.sign;
        r.sign = a1.sign;
        q.trim();
        r.trim();
        return make_pair(q, r / norm);
    }

    bigint operator/(const bigint &v) const
{
        return divmod(*this, v).first;
    }

    bigint operator%(const bigint &v) const
{
        return divmod(*this, v).second;
    }

    void operator/=(int v) {
        if (v < 0)
            sign = -sign, v = -v;
        for (int i = (int) a.size() - 1, rem
= 0; i >= 0; --i) {
            long long cur = a[i] + rem *
(long long) base;
            a[i] = (int) (cur / v);
            rem = (int) (cur % v);
        }
        trim();
    }

    bigint operator/(int v) const {
        bigint res = *this;
        res /= v;
```

```
const int N=1e5+5;
int dp[N];
int w[101],v[101];
int n,W;
int solve(){
    for(int i=0;i<N;i++)dp[i]=1e18;
    dp[0]=0;
    for(int i=0;i<n;i++){
        for(int j=N-1;j>=0;j--){
            if(dp[j]+w[i]<=W){

dp[j+v[i]]=min(dp[j+v[i]],dp[j]+w[i]);
            }
        }
    }
    for(int i=N-1;i>=0;i--){
        if(dp[i]!=1e18)return i;
    }
    return 0;
}
```

//Loại bỏ phần tử đầu tiên hoặc cuối cùng của dãy . Người chơi đó sẽ kiếm được điểm, với là phần tử bị loại bỏ.

//Cho x và y lần lượt là số điểm của Taro và Jiro sau khi trò chơi kết thúc. Taro muốn x-y lớn nhất có thể, trong Jiro lại muốn làm x-y bé nhất có thể.

```
int dp[1000][1000];
int solve(v a){
    for(int i=0;i<n;i++)dp[i][i]=a[i];
    for(int i=n-1;i>=0;i--){
        for(int j=i+1;j<n;j++){
            dp[i][j]=max( a[i]-dp[i+1][j] ,
a[j]-dp[i][j-1] );
        }
    }
    return dp[0][n-1];
}
```

//tam giác vuông cân

Cho vùng tọa độ Oxy bị giới hạn bởi gốc tọa độ (0, 0) và điểm trên cùng bên phải (X, Y). Nhiệm vụ của bạn là hãy xác định xem có bao nhiêu tam giác vuông cân.

Input:

Dòng đầu tiên là số lượng bộ test T (T <= 100).

Mỗi test gồm hai số nguyên X và Y.

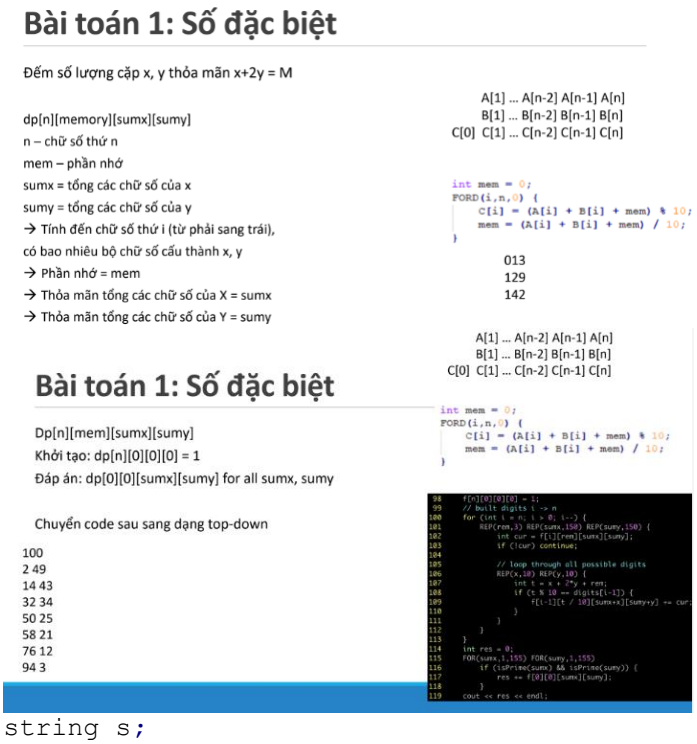
```
int x,y,ans;
int getx(int l,int r){
    l=max(l,0ll);
    r=min(r,x);
    return r-l+1;
}
int gety(int l,int r){
```

```
        return res;
    }

    int operator%(int v) const {
        if (v < 0)
            v = -v;
        int m = 0;
        for (int i = a.size() - 1; i >= 0; --i)
            m = (a[i] + m * (long long) base) % v;
        return m * sign;
    }

    void operator+=(const bigint &v) {
        *this = *this + v;
    }
    void operator-=(const bigint &v) {
        *this = *this - v;
    }
    void operator*=(const bigint &v) {
        *this = *this * v;
    }
    void operator/=(const bigint &v) {
        *this = *this / v;
    }

    bool operator<(const bigint &v) const {
        if (sign != v.sign)
            return sign < v.sign;
        if (a.size() != v.a.size())
            return a.size() * sign < v.a.size() * v.sign;
        for (int i = a.size() - 1; i >= 0; i--)
            if (a[i] != v.a[i])
                return a[i] * sign < v.a[i] * sign;
        return false;
    }

    bool operator>(const bigint &v) const {
        return v < *this;
    }
    bool operator<=(const bigint &v) const {
        return !(v < *this);
    }
    bool operator>=(const bigint &v) const {
        return !(*this < v);
    }
    bool operator==(const bigint &v) const {
        return !(*this < v) && !(v < *this);
    }
    bool operator!=(const bigint &v) const {
        return *this < v || v < *this;
    }
```

```
    l=max(l,0ll);
    r=min(r,y);
    return r-l+1;
}
void solve(int i,int j){
    ans+=getx(i-j,i-j+y)*gety(i+j-x,i+j)-1;
    ans+=getx(i+j-y,i+j)*gety(-i+j,-i+j+x)-1;
}
void solution(){
    cin>>x>>y;
    ans=0;
    for(int i=0;i<=x;i++){
        for(int j=0;j<=y;j++){
            solve(i,j);
        }
    }
    cout<<ans/2<<endl;
}
```

QHD chữ số:

//Một số được gọi là đặc biệt nếu như tổng các chữ số của nó là một số nguyên tố. Cho số tự nhiên N, hãy đếm số cặp (x, y) nguyên dương thỏa mãn x, y là số đặc biệt và x + 2y = N




```
string s;
int n;
int dp[20][4][150][150]={0};
```

```
    void trim() {
        while (!a.empty() && !a.back())
            a.pop_back();
        if (a.empty())
            sign = 1;
    }

    bool isZero() const {
        return a.empty() || (a.size() == 1
&& !a[0]);
    }

    bigint operator-() const {
        bigint res = *this;
        res.sign = -sign;
        return res;
    }

    bigint abs() const {
        bigint res = *this;
        res.sign *= res.sign;
        return res;
    }

    long long longValue() const {
        long long res = 0;
        for (int i = a.size() - 1; i >= 0;
i--)
            res = res * base + a[i];
        return res * sign;
    }

    friend bigint gcd(const bigint &a, const
bigint &b) {
        return b.isZero() ? a : gcd(b, a %
b);
    }
    friend bigint lcm(const bigint &a, const
bigint &b) {
        return a / gcd(a, b) * b;
    }

    void read(const string &s) {
        sign = 1;
        a.clear();
        int pos = 0;
        while (pos < (int) s.size() &&
(s[pos] == '-' || s[pos] == '+')) {
            if (s[pos] == '-')
                sign = -sign;
            ++pos;
        }
        for (int i = s.size() - 1; i >= pos;
i -= base_digits) {
            int x = 0;
            for (int j = max(pos, i -
base_digits + 1); j <= i; j++)
```

```
int digits[20]={0};
bool isprime(int x){
    if(x<2)return 0;
    for(int i=2;i<=sqrt(x);i++){
        if(x%i==0)return 0;
    }
    return 1;
}
/*main*/
int solve(int i,int rem,int sumx,int sumy){
    int ans=0;
    if(i==-1){

if(rem==0&&isprime(sumx)&&isprime(sumy))ret
urn 1;
        return 0;
    }
    if(dp[i][rem][sumx][sumy]!=-1)return
dp[i][rem][sumx][sumy];
    int tmp=0;
    for(int x=0;x<=9;x++)
    for(int y=0;y<=9;y++){
        int t=x+2*y+rem;
        if(t%10==digits[i]){
            tmp+=solve(i-
1,t/10,sumx+x,sumy+y);
        }
    }
    return dp[i][rem][sumx][sumy]=tmp;
}
void solution(){
    memset(dp,-1,sizeof(dp));
    cin>>s; n=s.size();
    for(int i=0;i<n;i++)digits[i]=s[i]-'0';
    cout<<solve(n-1,0,0,0)<<endl;
}
```

//Cho số nguyên N. Nhiệm vụ của bạn là hãy xác định xem các số trong phạm vi từ 0 tới N có bao nhiêu số mà biểu diễn nhị phân của nó có đúng K chữ số 0.

Bài toán 2:

- QHD chữ số với các bài toán đếm số lượng các số <= X và thỏa mãn tính chất cho trước, ta làm qhd từ trái sang phải như sau:
- Biểu diễn X sang dạng chữ số bits[1] bits[2] ... bits[n]
- Gọi f(i, isLower) = số lượng cách:
  - Chọn ra i chữ số đầu tiên sao cho số đang xây dựng T (i bit đầu của T <= i bit đầu của X)
  - Nếu số đang xây dựng T < X → isLower = 1
  - Nếu i bit đầu T bằng X → isLower = 0
  - Khởi tạo: dp[0][0] = 1

```
                x = x * 10 + s[j] - '0';
            a.push_back(x);
        }
        trim();
    }

    friend istream& operator>>(istream
&stream, bigint &v) {
        string s;
        stream >> s;
        v.read(s);
        return stream;
    }

    friend ostream& operator<<(ostream
&stream, const bigint &v) {
        if (v.sign == -1)
            stream << '-';
        stream << (v.a.empty() ? 0 :
v.a.back());
        for (int i = (int) v.a.size() - 2; i
>= 0; --i)
            stream << setw(base_digits) <<
setfill('0') << v.a[i];
        return stream;
    }

    static vector<int> convert_base(const
vector<int> &a, int old_digits, int
new_digits) {
        vector<long long> p(max(old_digits,
new_digits) + 1);
        p[0] = 1;
        for (int i = 1; i < (int) p.size();
i++)
            p[i] = p[i - 1] * 10;
        vector<int> res;
        long long cur = 0;
        int cur_digits = 0;
        for (int i = 0; i < (int) a.size();
i++) {
            cur += a[i] * p[cur_digits];
            cur_digits += old_digits;
            while (cur_digits >= new_digits)
{
                res.push_back(int(cur %
p[new_digits]));
                cur /= p[new_digits];
                cur_digits -= new_digits;
            }
        }
        res.push_back((int) cur);
        while (!res.empty() && !res.back())
            res.pop_back();
        return res;
    }
```

Bài toán 2:

- Biểu diễn X sang dạng chữ số bits[1] bits[2] ... bits[n]
- dp[0][0] = 1 (số 0)
- FOR(i,0,n-1) FOR(isLower, 0, 1) FOR(digit,0,9) // them chu so digit vào sau số hiện tại
  - So sánh digit với bit[i+1]
  - Xác định newIsLower
  - dp[i+1, newIsLower] += dp[i, isLower]

Đếm số lượng các số <= X với X = 254

i	0	1	2	3
bits[]	0	**2**	**5**	**4**
isLower = 0	1	2 (0**, 1**)	25 (0**, 1**, 20*, 21*,...,24*)	Before + 250, 251, 252, 253 = 254
isLower = 1	0	1 (2**)	1 (25*)	1 (254)

Bài toán 2: Số nhị phân có K bit 0

- Bài toán này dính trường hợp ngoại lệ với số 0, (00, 000, 0000... không được tính)
- Dp[i, isLower, k, positive]
  - k: số lượng chữ số 0
  - positive: số hiện tại đang = 0 hay > 0
  - Khởi tạo dp[0][0][0][0] = 1
- FOR(digit,0,1) // them chu so digit vào sau số hiện tại
  - Cập nhật newIsLower
  - newPositive = positive or (digit == 1)
  - newK = k + (digit == 0)
  - dp[i+1, newIsLower, newK, newPositive] += dp[i, isLower, k, positive]
- Đáp án ans = = dp[N, 0, K, 1] + dp[N, 1, K, 1]
- Nếu K == 1 thì ans = ans + 1 (tính riêng cho số 0)

```
int k;
v convert(int n){
    v bit;
    while(n>0){
        bit.push_back(n%2);
        n/=2;
    }
    reverse(bit.begin(),bit.end());
    return bit;
}
int n;
/*main*/
void solve1(){
    int ans=0;
    if(k==1)ans++;
    for(int i=1;i<=n;i++){
        v bit=convert(i);
        int cnt=0;
        for(int i=0;i<bit.size();i++){
            if(bit[i]==0)cnt++;
        }
            if(cnt==k)ans++;
    }
    cout<<ans<<endl;
}

void solve(){
    v bit=convert(n);
    // if(n==0)bit.push_back(0);
    if(n==0&&k==1){
        cout<<1<<endl;
        return ;
    }
```

```
    typedef vector<long long> vll;

    static vll karatsubaMultiply(const vll
&a, const vll &b) {
        int n = a.size();
        vll res(n + n);
        if (n <= 32) {
            for (int i = 0; i < n; i++)
                for (int j = 0; j < n; j++)
                    res[i + j] += a[i] *
b[j];
            return res;
        }

        int k = n >> 1;
        vll a1(a.begin(), a.begin() + k);
        vll a2(a.begin() + k, a.end());
        vll b1(b.begin(), b.begin() + k);
        vll b2(b.begin() + k, b.end());

        vll a1b1 = karatsubaMultiply(a1,
b1);
        vll a2b2 = karatsubaMultiply(a2,
b2);

        for (int i = 0; i < k; i++)
            a2[i] += a1[i];
        for (int i = 0; i < k; i++)
            b2[i] += b1[i];

        vll r = karatsubaMultiply(a2, b2);
        for (int i = 0; i < (int)
a1b1.size(); i++)
            r[i] -= a1b1[i];
        for (int i = 0; i < (int)
a2b2.size(); i++)
            r[i] -= a2b2[i];

        for (int i = 0; i < (int) r.size();
i++)
            res[i + k] += r[i];
        for (int i = 0; i < (int)
a1b1.size(); i++)
            res[i] += a1b1[i];
        for (int i = 0; i < (int)
a2b2.size(); i++)
            res[i + n] += a2b2[i];
        return res;
    }

    bigint operator*(const bigint &v) const
{
        vector<int> a6 = convert_base(this-
>a, base_digits, 6);
        vector<int> b6 = convert_base(v.a,
base_digits, 6);
        vll a(a6.begin(), a6.end());
```

```
    int len=bit.size();
    // cout<<len<<endl;
     bit.insert(bit.begin(),-1);
    ll dp[len+1][2][k+2][2];
    memset(dp,0,sizeof(dp));
    dp[0][0][0][0]=1;
    for(int i=0;i<len;i++){
        for(int
islower=0;islower<2;islower++){
            for(int kk=0;kk<=k;kk++){
                for(int
pos=0;pos<=1;pos++){
                    for(int
digit=0;digit<=1;digit++){

if(islower==0&&digit>bit[i+1])continue;
                        int newk;

if(digit==0&&pos>0)newk=kk+1;
                        else newk=kk;
                        int newpos=0;

if(digit==1||pos==1){
                            newpos=1;
                        }
                        int newislower=0;

if(islower==1)newislower=1;
                        else
if(digit<bit[i+1])newislower=1;

dp[i+1][newislower][newk][newpos]+=dp[i][is
lower][kk][pos];
                        // cout<<i<<"
"<<kk<<" "<<newk<<endl;
                    }
                }
            }
        }
    }

cout<<dp[len][0][k][1]+dp[len][1][k][1]<<en
dl;
}
```

//Số rõ ràng

```
        vll b(b6.begin(), b6.end());
        while (a.size() < b.size())
            a.push_back(0);
        while (b.size() < a.size())
            b.push_back(0);
        while (a.size() & (a.size() - 1))
            a.push_back(0), b.push_back(0);
        vll c = karatsubaMultiply(a, b);
        bigint res;
        res.sign = sign * v.sign;
        for (int i = 0, carry = 0; i < (int)
c.size(); i++) {
            long long cur = c[i] + carry;
            res.a.push_back((int) (cur %
1000000));
            carry = (int) (cur / 1000000);
        }
        res.a = convert_base(res.a, 6,
base_digits);
        res.trim();
        return res;
    }
};
```

SỐ HỌC :

//Sang era

```
vector<bool> a(1e6+5,1);
v e;
void era(){
    for(int i=2;i<=1000000;i++){
        if(a[i]){
            e.push_back(i);
            for(int
j=i*i;j<=1000000;j+=i)a[j]=0;
        }
    }
}
```

//tong uoc cua n

Với $n = p_1^{k_1} \times p_2^{k_2} \times \ldots \times p_r^{k_r}$

Số lượng ước của một số bằng

$$d = (k_1 + 1)(k_2 + 1)\ldots(k_r + 1)$$

Tổng tất cả các ước của n bằng

$$S = \prod_{i=1}^{r} \frac{p_i^{(k_i+1)} - 1}{p_i - 1} = \frac{p_1^{(k_1+1)} - 1}{p_1 - 1} \times \frac{p_2^{(k_2+1)} - 1}{p_2 - 1} \times \ldots \times \frac{p_r^{(k_r+1)} - 1}{p_r - 1}$$

// dinhli Euler

Bờm mới tìm được một tài liệu định nghĩa số rõ ràng như sau: Với số nguyên dương n, ta tạo số mới bằng cách lấy tổng bình phương các chữ số của nó, với số mới này ta lại lập lại công việc trên. Nếu trong quá trình đó, ta nhận được số mới là 1, thì số n ban đầu được gọi là số rõ ràng. Ví dụ, với $n = 19$, ta có:

$$19 \to 82 \ (= 1^2 + 9^2) \to 68 \to 100 \to 1$$

Như vậy, 19 là số rõ ràng.

Không phải mọi số đều rõ ràng. Ví dụ, với $n = 12$, ta có:

$$12 \to 5 \to 25 \to 29 \to 85 \to 89 \to 145 \to 42 \to 20 \to 4 \to 16 \to 37 \to 58 \to 89 \to 145$$

Bờm rất thích thú với định nghĩa số rõ ràng này và thách đố phú ông: Cho một số nguyên dương n, tìm số $S(n)$ là số rõ ràng liền sau số n, tức là $S(n)$ là số rõ ràng nhỏ nhất lớn hơn n. Tuy nhiên, câu hỏi đó quá dễ với phú ông và phú ông đã đố lại Bờm: Cho hai số nguyên dương n và m $(1 \le n, m \le 10^{15})$, hãy tìm số $S^m(n) = S(S(\ldots S(n)))$ là số rõ ràng liền sau thứ m của n.

Bạn hãy giúp Bờm giải câu đố này nhé!

```
int n, k;
bool isclear[N], was[N];
int dp[20][2][N];
v a;
int sumsqr(int x)
{
    int ans = 0;
    while (x > 0)
    {
        ans += (x % 10) * (x % 10);
        x /= 10;
    }
    return ans;
}
/*Main*/
void init()
{
    isclear[1] = 1;
    for (int i = 2; i <= N; i++)
    {
        memset(was, 0, sizeof(was));
        for (int j = i; !was[j]; j =
sumsqr(j))
        {
            was[j] = 1;
            if (j < i)
            {
                isclear[i] = isclear[j];
                break;
            }
        }
    }
}
int solve(int i, bool bigger, int sum)
{
    if (i < 0)
        return bigger && isclear[sum];
    ll &ans = dp[i][bigger][sum];
    if (ans != -1)
        return ans;
    ans = 0;
    for (int x = (bigger ? 0 : a[i]); x <=
9; ++x)
    {
```

- Định lý Euler:
  Khi a và m nguyên tố cùng nhau thì: $a^{\phi(m)} \equiv 1 \pmod{m}$,
- Với m là số nguyên tố, phi(m) = m-1

$$1 \equiv b^{m-1} \pmod{m}$$
$$a/b \equiv a \times b^{m-2} \pmod{m}$$

- Ứng dụng: Tính nCk (n <= 10^6)

STACK-QUEUE:

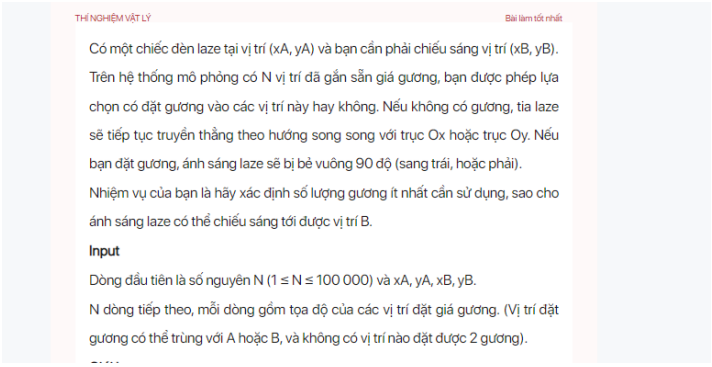
```
// trung to - hau to
int degree(char c){
    if(c=='^')return 5;
    if(c=='*'||c=='/')return 4;
    if(c=='+'||c=='-')return 3;
    return 2;
}
bool check(char c){
    return
((c>='a'&&c<='z')||(c>='A'&&c<='Z'));
}
string convert(string s){
    string res="";
    stack<char> st;
    for(int i=0;i<s.size();i++){
        if(check(s[i]))res+=s[i];
        else if(s[i]=='(')st.push(s[i]);
        else if(s[i]==')'){

while(!st.empty()&&st.top()!='('){
                res+=st.top();
                st.pop();
            }
            st.pop();
        }
        else if(s[i]=='+'||s[i]=='-
'||s[i]=='*'||s[i]=='/'||s[i]=='^'){

while(!st.empty()&&degree(st.top())>=degree(
s[i])){
                res+=st.top();
                st.pop();
            }
            st.push(s[i]);
        }
    }
    while(!st.empty()){
        if(st.top()!='(')res+=st.top();
        ans += solve(i - 1, bigger | (x >
a[i]), sum + x * x);
    }
    return ans;
}
void solution()
{
    cin >> n >> k;
    int _n = n;
    a.clear();
    while (_n > 0)
    {
        a.push_back(_n % 10);
        _n /= 10;
    }
    a.resize(18);
    memset(dp, -1, sizeof(dp));
    int ans = 0, sum = 0;
    bool bigger = 0;
    for (int i = 17; i >= 0; i--)
    {
        for (int x = (bigger ? 0 : a[i]); x
<= 9; x++)
        {
            int now = solve(i - 1, bigger |
(x > a[i]), sum + x * x);
            if (k > now)
                k -= now;
            else
            {
                ans = ans * 10 + x;
                bigger |= x > a[i];
                sum += x * x;
                break;
            }
        }
    }
    cout << ans << endl;
}
```

FENWICK TREE:

```
const int N=1e5+5;
int bit[N];
void update(int i,int k){
    for(;i<N;i+=i&-i)bit[i]+=k;
}
int get(int i){
    int ans=0;
    for(;i;i-=i&-i)ans+=bit[i];
    return ans;
}
```

```
        st.pop();
    }
    return res;
}
```

//thí nghiệm vật lí



THÍ NGHIỆM VẬT LÝ — Bài làm tốt nhất

Có một chiếc đèn laze tại vị trí (xA, yA) và bạn cần phải chiếu sáng vị trí (xB, yB). Trên hệ thống mô phỏng có N vị trí đã gắn sẵn giá gương, bạn được phép lựa chọn có đặt gương vào các vị trí này hay không. Nếu không có gương, tia laze sẽ tiếp tục truyền thẳng theo hướng song song với trục Ox hoặc trục Oy. Nếu bạn đặt gương, ánh sáng laze sẽ bị bẻ vuông 90 độ (sang trái, hoặc phải).

Nhiệm vụ của bạn là hãy xác định số lượng gương ít nhất cần sử dụng, sao cho ánh sáng laze có thể chiếu sáng tới được vị trí B.

Input

Dòng đầu tiên là số nguyên N (1 ≤ N ≤ 100 000) và xA, yA, xB, yB.

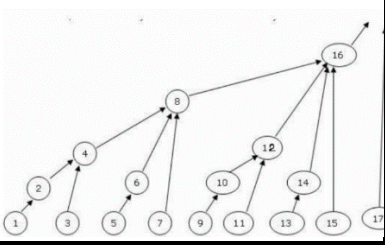
N dòng tiếp theo, mỗi dòng gồm tọa độ của các vị trí đặt giá gương. (Vị trí đặt gương có thể trùng với A hoặc B, và không có vị trí nào đặt được 2 gương).

```
const int N=1e5+5;
const int oo=1e9;
int n;
vector<ii> a(N);
v check(N,oo);
unordered_map<int,v>d[2];
void process(){
    queue<pair<int,bool>>q;
    q.push({0,1});
    q.push({0,0});
    check[0]=0;
    while(!q.empty()){
        ii u=q.front();  q.pop();
        int dir=u.se? 0:1;
        int id=u.se ? a[u.fi].fi:a[u.fi].se;
        for(int x:d[dir][id]){
            if(check[x]==oo){
                q.push({x,!u.se});
                check[x]=check[u.fi]+1;
            }
        }
    }
    cout<<(check[1]==oo ? -1:check[1]-1);
}
//
main(){
    ios_base::sync_with_stdio(0);cin.tie(0);
    cin>>n;
    for(int i=0;i<n+2;i++){
        cin>>a[i].fi>>a[i].se;
        d[0][a[i].fi].push_back(i);
        d[1][a[i].se].push_back(i);
    }
    process();
    return 0;
}
```

//Hexagame

- Ví dụ tính tổng a[1]+a[2]+...+a[n]
- Xây dựng cây BIT
  - tree[1] = a[1]
  - tree[2] = a[1] + a[2]
  - tree[3] = a[3]
  - tree[4] = a[1] + a[2] + a[3] + a[4]
  - tree[5] = a[5]
  - tree[6] = a[5] + a[6]
  - tree[7] = a[7]
  - tree[8] = a[1] + a[2] + a[3] + a[4] + a[5] + a[6] + a[7] + a[8]

a[1]+a[2]+...+a[15] = tree[8] + tree[12] + tree[14] + tree[15]



//bo ba so thu tu nguoc.đếm số bộ ba a[i]>a[j]>ak] với i < j < k.

```
int solve(int a[]){
    int right[n+1];
    int left[n+1];
    for(int i=n-1;i>=0;i--){
        right[i]=get(a[i]-1);
        update(a[i]);
    }
    for(int i=0;i<=n;i++)bit[i]=0;
    for(int i=0;i<n;i++){
        left[i]=i-get(a[i]);
        update(a[i]);
    }
    int res=0;

    for(int i=0;i<n;i++){
        res+=left[i]*right[i];
    }
    return res;
}
```

Cây IT:

// Truy van sum

```
int n,m;
int a[N],Tree[4*N],Lazy[4*N];
void build(int node,int l,int r){
    if(l==r)Tree[node]=a[l];
    else {
        build(2*node,l,(l+r)/2);
        build(2*node+1,(l+r)/2+1,r);

Tree[node]=Tree[2*node]+Tree[2*node+1];
    }
}
void update(int node,int l,int r,int
left,int right,int val){
    if(Lazy[node]!=0){
        Tree[node]+=(r-l+1)*Lazy[node];
        if(l!=r){
            Lazy[node*2]+=Lazy[node];
            Lazy[node*2+1]+=Lazy[node];
        }
```

HEXGAME là một trò chơi xếp hình gồm 10 miếng ghép hình lục giác đều, trên mỗi miếng ghép được điền một số nguyên, có 8 miếng được điền số từ 1 đến 8 và có hai miếng điền số 0. Các miếng liên kết với nhau tạo thành lưới tổ ong. Ban đầu các miếng ghép ở vị trí như hình vẽ. Tại mỗi bước, chọn một miếng ghép có đúng 6 miếng ghép kề cạnh làm tâm, rồi xoay một nấc 6 miếng ghép kề cạnh đó theo chiều kim đồng hồ. Như vậy chỉ có hai cách chọn tâm, đó là chọn tâm bên trái và chọn tâm bên phải.

Yêu cầu: Cho một trạng thái của trò chơi (nhận được sau một dãy biến đổi từ trạng thái ban đầu), hãy tính số phép biến đổi ít nhất để đưa về trạng thái ban đầu.

```
struct note{
    string u;
    int d;
};
string arr="";
string qleft(string q){
    string ans=q;
    ans[1]=q[0];
    ans[5]=q[1];
    ans[8]=q[5];
    ans[7]=q[8];
    ans[3]=q[7];
    ans[0]=q[3];
    return ans;
}
string qleft1(string q){
    string ans=q;
    ans[0]=q[1];
    ans[1]=q[5];
    ans[5]=q[8];
    ans[8]=q[7];
    ans[7]=q[3];
    ans[3]=q[0];
    return ans;
}
string qright(string q){
    string ans=q;
    ans[2]=q[1];
    ans[6]=q[2];
    ans[9]=q[6];
    ans[8]=q[9];
    ans[4]=q[8];
    ans[1]=q[4];
    return ans;
}
string qright1(string q){
    string ans=q;
    ans[1]=q[2];
    ans[2]=q[6];
    ans[6]=q[9];
    ans[9]=q[8];
    ans[8]=q[4];
    ans[4]=q[1];
    return ans;
```

```
        Lazy[node]=0;
    }
    if(r<left||l>right)return ;
    if(left<=l&&r<=right){
        Tree[node]+=(r-l+1)*val;
        if(l!=r){
            Lazy[node*2]+=val;
            Lazy[node*2+1]+=val;
        }
        return ;
    }

update(node*2,l,(l+r)/2,left,right,val);

update(node*2+1,(l+r)/2+1,r,left,right,val)
;
    Tree[node]=Tree[node*2]+Tree[node*2+1];
}
int get(int node,int l,int r,int left,int
right){
    if(l>right||r<left)return 0;
    if(Lazy[node]!=0){
        Tree[node]+=(r-l+1)*Lazy[node];
        if(l!=r){
            Lazy[node*2]+=Lazy[node];
            Lazy[node*2+1]+=Lazy[node];
        }
        Lazy[node]=0;
    }
    if(l>=left&&r<=right)return Tree[node];
    return
get(node*2,l,(l+r)/2,left,right)+get(node*2
+1,(l+r)/2+1,r,left,right);
}
```

Trie:

```
const int size=27;
struct Trie{
    Trie *children [size];
    bool isendofword;
};
Trie *getnode(){
    Trie *a=new Trie;
    a->isendofword=0;
    for(int i=0;i<size;i++){
        a->children[i]=NULL;
    }
    return a;
}
void insert(Trie *root,string s){
    Trie *a=root;
    for(int i=0;i<s.size();i++){
        int id=s[i]-'a';
        if(!a->children[id]){
            a->children[id]=getnode();
```

```
}
unordered_map<string,int>check1;
unordered_map<string,int>check2;
int main(){
    ios_base::sync_with_stdio(0);cin.tie(0);
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int ans=INT_MAX;
    int x;
    for(int i=0;i<10;i++){
        cin>>x;
        arr=arr+to_string(x);
    }
    string s="1238004765";
    queue<note>q;
    q.push({arr,0});
    check1[arr]=1;
    while(!q.empty()){
        note c=q.front();
        q.pop();
        if(c.d==13)continue;
        string adj1=qleft(c.u);
        string adj2=qright(c.u);
        if(check1[adj1]==0){
            q.push({adj1,c.d+1});
            check1[adj1]=c.d+1;
        }
        if(check1[adj2]==0){
            q.push({adj2,c.d+1});
            check1[adj2]=c.d+1;
        }
    }
    q.push({s,0});
    check2[s]=1;
    while(!q.empty()){
        note c=q.front();
        q.pop();
        if(check1[c.u]!=0){
            ans=min(ans,check1[c.u]+c.d);
        }
        if(c.d==13)continue;
        string adj1=qleft1(c.u);
        string adj2=qright1(c.u);
        if(check2[adj1]==0){
            q.push({adj1,c.d+1});
            check2[adj1]=c.d+1;
        }
        if(check2[adj2]==0){
            q.push({adj2,c.d+1});
            check2[adj2]=c.d+1;
        }
    }
    cout<<ans;
}
```

```
        }
        a=a->children[id];
    }
    a->isendofword=1;
}
bool search(Trie *root,string s){
    Trie *a=root;
    for(int i=0;i<s.size();i++){
        int id=s[i]-'a';
        if(!a->children[id])return 0;
        a=a->children[id];
    }
    return (a->isendofword);
}
```

//Một từ cần được tách thành các đoạn con sao cho mỗi đoạn con thuộc một tập các từ cho trước.Viết chương trình xác định số cách tách một từ cho trước.

```
    Trie *root = getnode();
    cin>>s>>n;
    int len=s.size();
    for(int i=1;i<=n;i++){
        cin>>x;
        insert(root,x);
    }
    dp[len]=1;
    for(int i=len-1;i>=0;i--){
        Trie *p=root;
        for(int j=i;j<len;j++){
            int t=s[j]-'a';
            if(p->children[t]==NULL)break;
            p=p->children[t];
            if(p-
>isendofword)dp[i]+=dp[j+1];
        }
        dp[i]%=mod;
    }
    cout<<dp[0]<<endl;
```

//Tin mật

Bessie định dẫn đàn bò đi trốn. Để đảm bảo bí mật, đàn bò liên lạc với nhau bằng cách tin nhắn nhị phân.

Từng là một nhân viên phản gián thông minh, John đã thu được M $(1 \leq M \leq 50000)$ tin nhắn mật, tuy nhiên với tin nhắn i John chỉ thu được b_i $(1 \leq b_i \leq 10000)$ bit đầu tiên.

John đã biên soạn ra 1 danh sách N $(1 \leq N \leq 50000)$ các từ mã hóa mà đàn bò có khả năng đang sử dụng. Thật không may, John chỉ biết được c_j $(1 \leq c_j \leq 10000)$ bit đầu tiên của từ mã hóa thứ j.

Với mỗi từ mã hóa j, John muốn biết số lượng tin nhắn mà John thu được có khả năng là từ mã hóa j này. Tức là với từ mã hóa j, có bao nhiêu tin nhắn thu được có phần đầu giống với từ mã hóa j này. Việc của bạn là phải tính số lượng này.

Tổng số lượng các bit trong dữ liệu đầu vào (tổng các b_i và c_j) không quá 500000.

```
struct Trie{
    Trie *children [2];
    int count,f;
    bool isendofword;
```

Dijkstra:

```
int n,m,start;
vector<ii> a[1005];
int d[1005];
bool check[1005];
struct cmp{
    bool operator()(ii x,ii y){
        return x.fi>=y.fi;
    }
};
void dijkstra(int u){
    for(int i=1;i<=n;i++)d[i]=1e9;
    d[u]=0;
    priority_queue<ii,vector<ii>,cmp>q;
    q.push({0,u});
    while(!q.empty()){
        int k=q.top().se; q.pop();
        for(auto x:a[k]){
            int s=x.se;
            int d_k_s=x.fi;
            if(d[k]+d_k_s<d[s]){
                d[s]=d[k]+d_k_s;
                q.push({d[s],s});
            }
        }
    }
    for(int i=1;i<=n;i++)cout<<d[i]<<" ";
    cout<<endl;
}
```

Floyd:

```
int a[105][105];
int n,m,q;
void floyd(){
    for(int k=1;k<=n;k++){
        for(int i=1;i<=n;i++){
            for(int j=1;j<=n;j++){
                if(a[i][j]>a[i][k]+a[k][j]){
                    a[i][j]=a[i][k]+a[k][j];
                }
            }
        }
    }
}
```

```
};
Trie *getnode(){
    Trie *a=new Trie;
    a->isendofword=0;
    a->count=0;
    a->f=0;
    for(int i=0;i<2;i++){
        a->children[i]=NULL;
    }
    return a;
}
void insert(Trie *root,string s){
    Trie *a=root;
    for(int i=0;i<s.size();i++){
        int id=s[i]-'0';
        if(!a->children[id]){
            a->children[id]=getnode();
        }
        a=a->children[id];
    }
    a->count++;
    a->isendofword=1;
}
void solve(Trie *root,string s){
    Trie *a=root;
    int ans=0,i=0;
    for(i=0;i<s.size();i++){
        int id=s[i]-'0';
        if(!a->children[id])break;
        a=a->children[id];
        ans+=a->count;
    }
    if(i==s.size())ans+=a->f;
    cout<<ans<<endl;
}
int DFS(Trie *root){
    if(root==NULL)return 0;
    root->f=DFS(root-
>children[0])+DFS(root->children[1]);
    return root->f+root->count;
}
string init(){
    string s="";
    int len;char x;
    cin>>len;
    for(int i=0;i<len;i++){
        cin>>x;s+=x;
    }
    return s;
}
//
main(){

ios_base::sync_with_stdio(0);cin.tie(0);
    // freopen("input.txt", "r", stdin);
    // freopen("output.txt", "w", stdout);
    Trie *root=getnode();
```

Hash:

//Cho xâu A và xâu B chỉ gồm các chữ cái thường.Hãy tìm tất cả các vị trí mà A xuất hiện trong B.

```
const int N=1000111;
const int p=1000000003;
string T,P;
int Pow[N],Hash[N];
int n,m;
void set_pow(){
    Pow[0]=1;
    for(int i=1;i<=m;i++)
        Pow[i]=(Pow[i-1]*26)%p;
}
void set_hash(){
    for(int i=1;i<=m;i++)
        Hash[i]=(Hash[i-1]*26+ T[i]-'a')%p;
}
int get_hash(int i,int j){
    return (Hash[j]-Hash[i-1]*Pow[j-
i+1]+p*p)%p;
}
void solution(){
    cin>>T>>P;
    m=T.size(),n=P.size();
    T=" "+T;  P=" "+P;
    set_pow();    set_hash();
    int hash_p=0;
    for(int i=1;i<=n;i++)
        hash_p=(hash_p*26+ P[i]-'a')%p;
    for(int i=1;i<=m-n+1;i++){
        if(hash_p==get_hash(i,i+n-
1))cout<<i<<" ";
    }
}
```

// xau con doi xung dai nhat

```
void solution(){
    set_left(); set_right();
    for(int i=1;i<=n;i++){
        // le
        int left=0,right=min(i,n-i);
        while(left<=right){
            int mid=(left+right)/2;
            if(check(i-mid+1,i+mid)){
                ans=max(ans,mid*2);
                left=mid+1;
            }
            else right=mid-1;
        }
        // chan
        left=0,right=min(i-1,n-i);
        while(left<=right){
            int mid=(left+right)/2;
    int n,m;
    cin>>n>>m;
    for(int i=1;i<=n;i++){
        string s=init();
        insert(root,s);
    }
    DFS(root);
    while(m--){
        string s=init();
        Trie *a=root;
        solve(a,s);
    }
    return 0;
}
```

//chuỗi từ

Chuỗi từ có độ dài n là một dãy các từ $w_1, w_2, ..., w_n$ sao cho với mọi $1 \leq i < n$, từ w_i là tiền tố của từ w_{i+1}.

Nhắc lại từ u có độ dài k là tiền tố của từ v có độ dài l nếu $l > k$ và các ký tự đầu tiên của v trùng với từ u.

Cho tập hợp các từ $S = (s_1, s_2, \ldots, s_m)$. Tìm chuỗi từ dài nhất có thể xây dựng được bằng cách dùng các từ trong tập hợp S (có thể không sử dụng hết các từ).

```
int n;
string s[N];
int dp[N];
const int size=27;
struct Trie{
    Trie *children [size];
    bool isendofword;
};
Trie *getnode(){
    Trie *a=new Trie;
    a->isendofword=0;
    for(int i=0;i<size;i++){
        a->children[i]=NULL;
    }
    return a;
}
void insert(Trie *root,string s){
    Trie *a=root;
    for(int i=0;i<s.size();i++){
        int id=s[i]-'a';
        if(!a->children[id]){
            a->children[id]=getnode();
        }
        a=a->children[id];
    }
    a->isendofword=1;
}
int search(Trie *root,string s){
    int ans=0;
    Trie *a=root;
    for(int i=0;i<s.size();i++){
        int id=s[i]-'a';
        if(!a->children[id])return ans;
        a=a->children[id];
```

```
            if(check(i-mid,i+mid)){
                ans=max(ans,mid*2+1);
                left=mid+1;
            }
            else right=mid-1;
        }
    }
    cout<<ans<<endl;
}
```

//xau con xuat hien k lan

```
int get(int i,int j){
    return (Hash[j]-Hash[i]*Pow[j-i]+p*p)%p;
}
void solution(){
    ans=0;ok=0;
    int left=1,right=n;
    while(left<=right){
        int mid=(left+right)/2;
        for(int i=mid;i<=n;i++){
            d[i]=get(i-mid,i);
        }
        sort(d+mid,d+n+1);
        f[mid]=1;ok=0;
        for(int i=mid+1;i<=n;i++){
            if(d[i]==d[i-1])f[i]=f[i-1]+1;
            else f[i]=1;
            if(f[i]>=k){
                ok=1;
                break;
            }
        }
        if(ok){
            left=mid+1;
            ans=mid;
        }
        else right=mid-1;
    }
    cout<<ans<<endl;
}
```

KMP:

LPS (longest proper prefix is also suffix of P)

LPS[i] = độ dài tiền tố chuẩn dài nhất của xâu prefix P[1...i-1] và cũng là hậu tố của xâu P[1...i]

(**Lưu ý:** chỉ số của xâu bắt đầu từ 1)

LPS[1] = 0 (luôn luôn)

AAAA LPS[] = [0 1 2 3]

ABCDE LPS[] = [0 0 0 0 0]

AAABAAA LPS[] = [0 1 2 0 1 2 3]

i = 2, prefix = P[1..1] = A, postfix = AA, LPS[2] = 1
i = 3, prefix = P[1..2] = AA, postfix = AAA, LPS[3] = 2
i = 4, prefix = P[1..3] = AAA, postfix = AAAB, LPS[4] = 0
i = 5, prefix = P[1..4] = AAAB, postfix = AAABA, LPS[5] = 1
i = 6, prefix = P[1..5] = AAABA, postfix = AAABAA, LPS[6] = 2
I = 7, prefix = P[1..6] = AAABAA, postfix = AAABAAA, LPS[7] = 3

```
        if(i!=s.size()-1&&a->isendofword)ans++;
    }
    return ans;
}
bool cmp(string x,string y){
    return (x.size()<y.size());
}
//
main(){
    Trie *root=getnode();
    cin>>n;
    for(int i=1;i<=n;i++)cin>>s[i];
    sort(s+1,s+n+1,cmp);
    for(int i=1;i<=n;i++){
        dp[i]=search(root,s[i])+1;
        insert(root,s[i]);
    }
    cout<<*max_element(dp+1,dp+n+1);
    return 0;
}
```

ConvexHull:

//Ver 1

```
#define EPS 1e-6
const double PI = acos(-1.0);

double DEG_to_RAD(double d) { return d * PI / 180.0; }
double RAD_to_DEG(double r) { return r * 180.0 / PI; }

inline int cmp(double a, double b) {
    return (a < b - EPS) ? -1 : ((a > b + EPS) ? 1 : 0);
}

struct Point {
    double x, y;
    Point() { x = y = 0.0; }
    Point(double _x, double _y) : x(_x), y(_y) {}

    Point operator + (const Point& a) const { return Point(x+a.x, y+a.y); }
    Point operator - (const Point& a) const { return Point(x-a.x, y-a.y); }
    Point operator * (double k) const { return Point(x*k, y*k); }
    Point operator / (double k) const { return Point(x/k, y/k); }

    double operator * (const Point& a) const { return x*a.x + y*a.y; } // dot product
```

LPS[i] = độ dài tiền tố chuẩn dài nhất của xâu P[1...i-1] sao cho nó cũng là hậu tố của xâu P[1...i]

(Lưu ý: chỉ số của xâu bắt đầu từ 1)

AAAA LPS[] = [0 1 2 3]
ABCDE LPS[] = [0 0 0 0 0]
AAABAAA LPS[] = [0 1 2 0 1 2 3]
AAAABAACD LPS[] = [0 1 2 3 0 1 2 0 0]
ABABAD LPS[] = [0 0 1 2 3 0]
ABABAA LPS[] = [0 0 1 2 3 1]

LPS[i] = j sao cho P[1..j] == P[i-j+1, ..., i] với j <= i-1
→ với xâu P[1→ i-1] đã khớp được j kí tự đầu của P
Khi move từ vị trí i→ i+1, nếu không khớp kí tự i sẽ nhảy theo mảng LPS

```
void kmpPreprocess()
{
    int j = 1;
    lps[1] = 0;
    FOR(i,2,m) {
        int j = lps[i-1];
        while(j > 0 && P[j+1] != P[i])
            j = lps[j];
        if(P[j+1] == P[i])
            lps[i] = j+1;
        else
            lps[i] = 0;
    }
}
```

LPS[i] = độ dài tiền tố chuẩn dài nhất của xâu P[1...i-1] sao cho nó cũng là hậu tố của xâu P[1...i]

(Lưu ý: chỉ số của xâu bắt đầu từ 1)

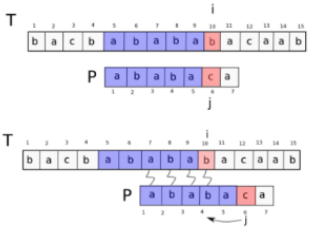
AAAA LPS[] = [0 1 2 3]
ABCDE LPS[] = [0 0 0 0 0]
AAABAAA LPS[] = [0 1 2 0 1 2 3]
AAAABAACD LPS[] = [0 1 2 3 0 1 2 0 0]
ABABAD LPS[] = [0 0 1 2 3 0]

i = 6, j = LPS[5] = 3
P[4] != P[6] → j = LPS[3] = 1
P[2] != P[6] → j = LPS[1] = 0
→ nhảy LPS 2 lần

```
void kmpPreprocess()
{
    int j = 1;
    lps[1] = 0;
    FOR(i,2,m) {
        int j = lps[i-1];
        while(j > 0 && P[j+1] != P[i])
            j = lps[j];
        if(P[j+1] == P[i])
            lps[i] = j+1;
        else
            lps[i] = 0;
    }
}
```

LPS[i] = độ dài tiền tố chuẩn dài nhất của xâu P[1...i-1] sao cho nó cũng là hậu tố của xâu P[1...i]

Tại vị trí i-1, đã khớp j kí tự của P.
Tới vị trí i, nếu khớp j+1 kí tự → tiếp tục (i++, j++)
Ngược lại, nhảy j = LPS[j] (=j2 < j)
→ có được j2 kí tự trong T khớp với j2 kí tự đầu của I



i = 10, j = 5, P[6] != T[10]
Lùi j về vị trí j = LPS[j] = LPS[5] = 3
→ Đã khớp 3 kí tự, duyệt tiếp với i = 10 và j = 3
j = 3, P[4] == T[10] → j++, i++
j = 4, P[5] == T[11] → j++, i++
j = 5, P[6] == T[12] → j++, i++
j = 6, P[7] == T[13] → j++, i++
j = 7 → in ra đáp án đã khớp → nhảy j = LPS[j]

T = ab**ababa**cade
P = ababac
LPS = [0 0 1 2 3 0]
i = 1, j = 0, P[1] == T[1] → j = 1
i = 2, j = 1, P[2] == T[2] → j = 2
i = 3, j = 2, P[3] == T[3] → j = 3
i = 4, j = 3, P[4] == T[4] → j = 4
i = 5, j = 4, P[5] == T[5] → j = 5
i = 6, j = 5 → j = LPS[j] = 3→ j = j+1 = 4
i = 7, j = 4 → P[5] == T[7] → j = j+1 = 5
i = 8, j = 5 → P[6] == T[8] → j = j+1 = 6
→ In ra khớp tại vị trí 3 → 8
j = LPS[j] = 0
i = 9, j = 0 → P[1] = T[9] → j = j+1 = 1
i = 10, j = 1 → j = LPS[j] = LPS[1] = 0 → so sánh fail
i = 11, j = 0 → so sánh fail → j = 0

```
void kmpSearch() {
    vector<int> ans;
    int j = 0;
    for (int i = 1; i <= n; i++) {
        while(P[j+1] != T[i] && j > 0)
            j = lps[j];

        if (P[j+1] == T[i])
            j++;

        if (j == m) {
            cout << "Matched " << i-j+1 << endl;
            j = lps[j];
        }
    }
}
```

//Xâu con substr KMP

```
int n,m;
string T,P;
int lps[N];
/*Main*/
void set_LPS(){
    for(int i=2;i<=m;i++){
        int j=lps[i-1];
        while(j>0&&P[j+1]!=P[i])j=lps[j];
        if(P[j+1]==P[i])lps[i]=j+1;
        else lps[i]=0;
    }
}
void KMP_search(){
    int j=0;
    for(int i=1;i<=n;i++){
        while(P[j+1]!=T[i]&&j>0) j=lps[j];
```

```
    double operator % (const Point& a)
const { return x*a.y - y*a.x; } // cross
product

    int cmp(Point q) const { if (int t =
::cmp(x,q.x)) return t; return
::cmp(y,q.y); }

    #define Comp(x) bool operator x (Point
q) const { return cmp(q) x 0; }
    Comp(>) Comp(<) Comp(==) Comp(>=)
Comp(<=) Comp(!=)
    #undef Comp

    Point conj() { return Point(x, -y); }
    double norm() { return x*x + y*y; }

    // Note: There are 2 ways for
implementing len():
    // 1. sqrt(norm()) --> fast, but
inaccurate (produce some values that are of
order X^2)
    // 2. hypot(x, y) --> slow, but much
more accurate
    double len() { return sqrt(norm()); }

    Point rotate(double alpha) {
        double cosa = cos(alpha), sina =
sin(alpha);
        return Point(x * cosa - y * sina, x
* sina + y * cosa);
    }
};
double area2(Point a, Point b, Point c) {
    return a%b + b%c + c%a;
}
#ifdef REMOVE_REDUNDANT
bool between(const Point &a, const Point
&b, const Point &c) {
    return (fabs(area2(a,b,c)) < EPS &&
(a.x-b.x)*(c.x-b.x) <= 0 && (a.y-b.y)*(c.y-
b.y) <= 0);
}
#endif

void ConvexHull(vector<Point> &pts) {
    sort(pts.begin(), pts.end());
    pts.erase(unique(pts.begin(),
pts.end()), pts.end());
    vector<Point> up, dn;
    for (int i = 0; i < (int) pts.size();
i++) {
        // Note: If need maximum points on
convex hull, need to change >= and <= to >
and <.
```

```
        if(P[j+1]==T[i])j++;
        if(j==m){
            cout<<i-j+1<<" ";
            j=lps[j];
        }
    }
}
void init(){
    memset(lps,0,sizeof(lps));
}
void solution(){
    cin>>T>>P;
    n=T.size(),m=P.size();
    T=" "+T;P=" "+P;
    set_LPS();
    KMP_search();
    cout<<endl;
}
```

Cây khung nhỏ nhất:

```
//cây khung nhỏ nhất
int n, m;
vector<int> parent(300001);
vector<int> ranked(300001);

struct Edge
{
    int u, v;
    long long w;
    Edge(int u, int v, long long w)
    {
        this->u = u;
        this->v = v;
        this->w = w;
    }
};

bool cmp(Edge x, Edge y)
{
    return x.w < y.w;
}

void init()
{
    for (int i = 0; i <= 300000; i++)
    {
        parent[i] = i;
        ranked[i] = 0;
    }
}

int findSet(int x)
```

```
        while (up.size() > 1 &&
area2(up[up.size()-2], up.back(), pts[i])
>= 0) up.pop_back();
        while (dn.size() > 1 &&
area2(dn[dn.size()-2], dn.back(), pts[i])
<= 0) dn.pop_back();
        up.push_back(pts[i]);
        dn.push_back(pts[i]);
    }
    pts = dn;
    for (int i = (int) up.size() - 2; i >=
1; i--) pts.push_back(up[i]);

#ifdef REMOVE_REDUNDANT
    if (pts.size() <= 2) return;
    dn.clear();
    dn.push_back(pts[0]);
    dn.push_back(pts[1]);
    for (int i = 2; i < (int) pts.size();
i++) {
        if (between(dn[dn.size()-2],
dn[dn.size()-1], pts[i])) dn.pop_back();
        dn.push_back(pts[i]);
    }
    if (dn.size() >= 3 &&
between(dn.back(), dn[0], dn[1])) {
        dn[0] = dn.back();
        dn.pop_back();
    }
    pts = dn;
#endif
```

```
{
    if (parent[x] == x)
        return x;
    parent[x] = findSet(parent[x]);
    return parent[x];
}

bool Union(int u, int v)
{
    u = findSet(u);
    v = findSet(v);
    if (u == v)
        return false;
    parent[v] = u;
    return true;
}

vector<Edge> edges;
main()
{
    int t;
    cin >> t;
    while (t--)
    {
        init();
        edges.clear();
        cin >> n >> m;
        long long res = 0;
        while (m--)
        {
            int u, v;
            long long w;
            cin >> u >> v >> w;
            edges.push_back(Edge(u, v, w));
        }
        sort(edges.begin(), edges.end(),
cmp);
        for (auto i : edges)
        {
            if (!Union(i.u, i.v))
                continue;
            res += i.w;
        }
        cout << res << endl;
    }
}
```

//Nối điểm

Cho N điểm trên mặt phẳng Oxy. Để vẽ được đoạn thẳng nối A và B sẽ tốn chi phí tương đương với khoảng cách từ A tới B.

Nhiệm vụ của bạn là nối các điểm với nhau, sao cho N điểm đã cho tạo thành 1 thành phần liên thông duy nhất và chi phí để thực hiện là nhỏ nhất có thể.

Input:

Dòng đầu tiên là số lượng bộ test T (T <= 20).

Mỗi test bắt đầu bởi số nguyên N (1 <= N <= 100).

N dòng tiếp theo, mỗi dòng gồm 2 số thực x[i], y[i] là tọa độ của điểm thứ i (|x[i]|, |y[i]| <= 100).

Output:

Với mỗi test, in ra chi phí nhỏ nhất tìm được với độ chính xác 6 chữ số thập phân sau dấu phảy.

```
int n;
ii a[N];
vector<iii>List;
int parent[N];
db dd(ii x,ii y){
    ii ans={y.fi-x.fi,y.se-x.se};
    return
sqrt(ans.fi*ans.fi+ans.se*ans.se);
}
int find(int i){
    if(parent[i]==-1)return i;
    parent[i]=find(parent[i]);
    return parent[i];
}
bool Union(int u, int v)
{
    u = find(u);
    v = find(v);
    if (u == v)
        return false;
    parent[v] = u;
    return true;
}
bool cmp(iii a,iii b){
    return a.se<b.se;
}
void kruskal(){
    db ans=0;
    sort(List.begin(),List.end(),cmp);
    for(auto x:List){
        if(!Union(x.fi.fi,x.fi.se))continue;
        ans+=x.se;
    }
    printf("%.6lf\n",ans);
}
void solution(){
    memset(parent,-1,sizeof(parent));
    List.clear();
    cin>>n;
    for(int i=1;i<=n;i++){
        cin>>a[i].fi>>a[i].se;
    }
    for(int i=1;i<=n;i++){
        for(int j=1;j<i;j++){
```

```
List.push_back({{i,j},dd(a[i],a[j])});
        }
    }
    kruskal();
}
```