目录

[一、 图论 2](#_Toc509048574)

[1、 BFS版二分图匹配 2](#_Toc509048575)

[2、 Prufer编码与Cayley公式 4](#_Toc509048576)

[二、 数学 5](#_Toc509048577)

[1、 double矩阵加乘、快速幂 5](#_Toc509048578)

[2、 康托展开及逆运算 6](#_Toc509048579)

[3、 整数拆分 8](#_Toc509048580)

[4、 背包 10](#_Toc509048581)

[5、 NTT（新） 12](#_Toc509048582)

[三、 字符串 14](#_Toc509048583)

[1、 hash和hashset 14](#_Toc509048584)

[四、 杂 16](#_Toc509048585)

[1、 真·快速读入 16](#_Toc509048586)

[2、 头文件 16](#_Toc509048587)

# 图论

## BFS版二分图匹配

#include<cstdio>

#include<algorithm>

#include<iostream>

#include<cstring>

using namespace std;

typedef long long LL;

const int MAX=1e5+5;//按理说取左侧点数即可

const int MAXM=6e6+5;//总边数

int head[MAX];//数组形式vector

int nxt[MAXM],val[MAXM];

int match[MAX<<1],check[MAX<<1],prv[MAX<<1];

int que[MAXM];//模拟队列

int st,en,cnt,numL;//numL需要初始化为左侧点数！

inline void add\_edge(int u,int v)//加边 只需要加左到右的即可

{

val[cnt]=v;nxt[cnt]=head[u];head[u]=cnt++;

}

//下标 0——numL-1

int Hungarian()

{

int ans=0;

memset(match,-1,sizeof(match));

memset(check,-1,sizeof(check));

for(int i=0;i<numL;i++)

{

if(match[i]==-1)

{

st=en=0;

que[en++]=i;

prv[i]=-1;

bool sta=false;

while(st!=en&&!sta)

{

int u=que[st];

for(int t=head[u];t!=-1&&!sta;t=nxt[t])

{

int v=val[t];

if(check[v]!=i)

{

check[v]=i;

que[en++]=match[v];

if(match[v]>=0)

prv[match[v]]=u;

else

{

sta=true;

int d=u,e=v;

while(d!=-1)

{

int q=match[d];

match[d]=e;

match[e]=d;

d=prv[d];

e=q;

}

}

}

}

++st;

}

if(match[i]!=-1)++ans;

}

}

return ans;

}

int n,a[MAX],hp[MAX],p[MAX],a2[MAX],hp2[MAX];

int main()

{

while(~scanf("%d",&n))

{

for(int i=0;i<n;i++){read(a[i]);read(hp[i]);read(p[i]);}

// scanf("%d%d%d",&a[i],&hp[i],&p[i]);

for(int i=0;i<n;i++){read(a2[i]);read(hp2[i]);}

// scanf("%d%d",&a2[i],&hp2[i]);

memset(head,-1,sizeof(head));

numL=n;

for(int i=0;i<n;i++)

{

if(!p[i])

{

for(int j=0;j<n;j++)

{

if(a[i]>=hp2[j]&&hp[i]>a2[j])add\_edge(i,j+n);

}

}

else if(p[i]==1)

{

for(int j=0;j<n;j++)

if(a[i]>=hp2[j])add\_edge(i,j+n);

}

else if(p[i]==2)

{

for(int j=0;j<n;j++)

if(hp[i]>a2[j])add\_edge(i,j+n);

}

else

{

for(int j=0;j<n;j++)

add\_edge(i,j+n);

}

}

// cout<<Hungarian()<<endl;

// cout<<Hungarian()<<endl;

if(Hungarian()==n)

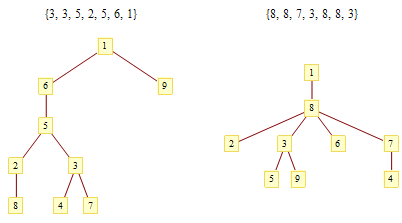
printf("YES\n");

else printf("NO\n");

}

}

## Prufer编码与Cayley公式



Cayley公式是说，一个完全图K\_n有n^(n-2)棵生成树，换句话说n个节点的带标号的无根树有n^(n-2)个。今天我学到了Cayley公式的一个非常简单的证明，证明依赖于Prüfer编码，它是对带标号无根树的一种编码方式。

给定一棵带标号的无根树，找出编号最小的叶子节点，写下与它相邻的节点的编号，然后删掉这个叶子节点。反复执行这个操作直到只剩两个节点为止。由于节点数n>2的树总存在叶子节点，因此一棵n个节点的无根树唯一地对应了一个长度为n-2的数列，数列中的每个数都在1到n的范围内。下面我们只需要说明，任何一个长为n-2、取值范围在1到n之间的数列都唯一地对应了一棵n个节点的无根树，这样我们的带标号无根树就和Prüfer编码之间形成一一对应的关系，Cayley公式便不证自明了。

注意到，如果一个节点A不是叶子节点，那么它至少有两条边；但在上述过程结束后，整个图只剩下一条边，因此节点A的至少一个相邻节点被去掉过，节点A的编号将会在这棵树对应的Prüfer编码中出现。反过来，在Prüfer编码中出现过的数字显然不可能是这棵树（初始时）的叶子。于是我们看到，没有在Prüfer编码中出现过的数字恰好就是这棵树（初始时）的叶子节点。找出没有出现过的数字中最小的那一个（比如④），它就是与Prüfer编码中第一个数所标识的节点（比如③）相邻的叶子。接下来，我们递归地考虑后面n-3位编码（别忘了编码总长是n-2）：找出除④以外不在后n-3位编码中的最小的数（左图的例子中是⑦），将它连接到整个编码的第2个数所对应的节点上（例子中还是③）。再接下来，找出除④和⑦以外后n-4位编码中最小的不被包含的数，做同样的处理……依次把③⑧②⑤⑥与编码中第3、4、5、6、7位所表示的节点相连。最后，我们还有①和⑨没处理过，直接把它们俩连接起来就行了。由于没处理过的节点数总比剩下的编码长度大2，因此我们总能找到一个最小的没在剩余编码中出现的数，算法总能进行下去。这样，任何一个Prüfer编码都唯一地对应了一棵无根树，有多少个n-2位的Prüfer编码就有多少个带标号的无根树。

一个有趣的推广是，n个节点的度依次为D1, D2, …, Dn的无根树共有(n-2)! / [ (D1-1)!(D2-1)!..(Dn-1)! ]个，因为此时Prüfer编码中的数字i恰好出现Di-1次。

# 数学

## double矩阵加乘、快速幂

const int m\_num=5;//矩阵的大小

/\*

矩阵 e[i][j]表示第i行第j列的矩阵元素的值

\*/

struct matrix

{

double e[m\_num][m\_num];

int row,col;

matrix(){}

matrix(int \_r,int \_c):row(\_r),col(\_c){memset(e,0,sizeof(e));}

matrix operator \* (const matrix &tem)const

{

matrix ret=matrix(row,tem.col);//新形成的矩阵规模为 左行右列

for(int i=1;i<=ret.row;i++)

for(int j=1;j<=ret.col;j++)

for(int k=1;k<=col;k++)

ret.e[i][j]+=1LL\*e[i][k]\*tem.e[k][j];

return ret;

}

matrix operator + (const matrix &tem)const

{

matrix ret=matrix(row,col);

for(int i=1;i<=row;i++)

for(int j=1;j<=col;j++)ret.e[i][j]+=e[i][j]+tem.e[i][j];

return ret;

}

void getE()//化为单位阵

{

for(int i=1;i<=row;i++)

for(int j=1;j<=col;j++)e[i][j]=(i==j?1:0);

}

};

matrix m\_qpow(matrix tem,int x)//矩阵快速幂

{

matrix ret=matrix(tem.row,tem.row);

ret.getE();

while(x)

{

if(x&1)ret=ret\*tem;

x>>=1;tem=tem\*tem;

}

return ret;

}

## 康托展开及逆运算

X=a[n]\*(n-1)!+a[n-1]\*(n-2)!+...+a[i]\*(i-1)!+...+a[1]\*0! ，其中a[i]为当前未出现的元素中是排在第几个（从0开始）。这就是康托展开。

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

const int N = 100 + 10;

char s[N];

ll fact[N];

void fact\_table()//阶乘表，这里仅打表到20

{

fact[0] = 1;

for(int i = 1; i <= 20; i++) fact[i] = i \* fact[i-1];

}

ll Cantor\_expansion(char \*s)

{

ll res = 0;

int len = strlen(s);

for(int i = 0; s[i]; i++)

{

int rnk = 0;

for(int j = i+1; s[j]; j++)

if(s[i] > s[j]) rnk++;

res += rnk \* fact[len-i-1];

}

return res;

}

void Cantor\_inverse\_expansion(ll n, int m)

{//n是在全排列中的名次，注意是从0开始计数的，若从1开始计数则要减去1。m是元素个数

vector<int> num;

int arr[N], k = 0;

for(int i = 1; i <= m; i++) num.push\_back(i);

for(int i = m; i >= 1; i--)

{

ll r = n % fact[i-1];

ll t = n / fact[i-1];

n = r;

sort(num.begin(), num.end());

arr[k++] = num[t];

num.erase(num.begin() + t);

}

for(int i = 0; i < k; i++) printf("%d", arr[i]);

printf("\n");

}

int main()

{

fact\_table();

while(~ scanf("%s", s))

{

int len = strlen(s);

ll res = Cantor\_expansion(s);//求出此排列的名次

printf("%lld\n", res);

Cantor\_inverse\_expansion(res, len);//根据排列的名次复原此排列

}

return 0;

}

## 整数拆分

/\*hdu4651

题意：

普通的整数拆分

限制：

1 <= n <= 1e5

思路：

五边形数定理

\*/

#include<iostream>

#include<cstdio>

using namespace std;

#define LL \_\_int64

const int N=100005;

const int MOD=1000000007;

LL dp[N],fi[N];

LL five(LL x){ return (3\*x\*x-x)/2; }

//五边形数

void wbxs(){

dp[0]=1;

int t=1000; //其实可以等于sqrt(N)

for(int i=-t;i<=t;++i)

fi[i+t]=five(i); //Q

for(int i=1;i<=100000;++i){

int flag=1;

for(int j=1;;++j){

LL a=fi[j+t],b=fi[-j+t];

if(a>i && b>i) break;

if(a<=i) dp[i]=(dp[i]+dp[i-a]\*flag+MOD)%MOD; //p

if(b<=i) dp[i]=(dp[i]+dp[i-b]\*flag+MOD)%MOD;

flag\*=-1;

}

}

}

int main(){

wbxs();

int T,n;

scanf("%d",&T);

while(T--){

scanf("%d",&n);

printf("%I64d\n",dp[n]);

}

return 0;

}

/\*hdu4658

题意：

整数n进行拆分，拆分过程中一个数重复次数要小于k

限制：

0 <= n <= 1e5

思路：

母函数，五边形数定理

可以得出该题的母函数为：

G[x]=( 1 + x + x^2 +...+ x^(k-1) ) \* ( 1 + x^2 + x^4 +...+ x^((k-1)\*2) ) \*...

=(1-x^k)/(1-x) \* (1-x^(k\*2))/(1-x^2) \* ...

=( (1-x^k) \* (1-(x^k)^2) \*... ) / ( (1-x) \* (1-x^2) \*... )

=Q(x^k)/Q(x) (根据五边形数定理得)

=Q(x^k)\*P(x)

=( 1 - x^k - (x^k)^2 + (x^k)^5 + (x^k)^7 -... ) \* ( 1 + x + 2\*x^2 + 3\*x^3 + 5\*x^4 + 7\*x^5 + ... )

接下来就是按照幂的对应关系算一下就行了。

\*/

#include<iostream>

#include<cstdio>

#include<cmath>

using namespace std;

#define LL \_\_int64

const int N=100005;

const int MOD=1000000007;

int p[N],Q[N];

int five(int x){ return (3\*x\*x-x)/2; }

int PY;

void wbxs(){

p[0]=1;

PY=sqrt(N)+1;

for(int i=-PY;i<=PY;++i)

Q[i+PY]=five(i);

for(int i=1;i<=100000;++i){

int flag=1;

for(int j=1;;++j){

int a=Q[j+PY],b=Q[-j+PY];

if(a>i && b>i) break;

if(a<=i) p[i]=(p[i]+p[i-a]\*flag)%MOD;

if(b<=i) p[i]=(p[i]+p[i-b]\*flag)%MOD;

flag\*=-1;

}

if(p[i]<0) p[i]=(p[i]+MOD)%MOD;

}

}

int n,k;

void gao(){

int ans=p[n];

int flag=-1;

for(int i=1;;++i){

int a=Q[i+PY],b=Q[-i+PY];

if(a\*k>n && b\*k>n) break;

//cout<<a<<' '<<b<<endl;

if(a\*k<=n) ans=(ans+p[n-a\*k]\*flag)%MOD;

if(b\*k<=n) ans=(ans+p[n-b\*k]\*flag)%MOD;

flag\*=-1;

}

ans=(ans+MOD)%MOD;

printf("%d\n",ans);

}

int main(){

wbxs();

int T;

scanf("%d",&T);

while(T--){

scanf("%d%d",&n,&k);

gao();

}

return 0;

}

## 背包

#include<stdio.h>

#include<iostream>

#include<string.h>

#include<algorithm>

using namespace std;

const int MAXN=110;

int dp[MAXN];

int value[MAXN];//每袋的价格

int weight[MAXN];//每袋的重量

int bag[MAXN];//袋数

int nValue,nKind;

//0-1背包，代价为cost,获得的价值为weight

void ZeroOnePack(int cost,int weight)

{

for(int i=nValue;i>=cost;i--)

dp[i]=max(dp[i],dp[i-cost]+weight);

}

//完全背包，代价为cost,获得的价值为weight 每种无穷多

void CompletePack(int cost,int weight)

{

for(int i=cost;i<=nValue;i++)

dp[i]=max(dp[i],dp[i-cost]+weight);

}

//多重背包

void MultiplePack(int cost,int weight,int amount)

{

if(cost\*amount>=nValue) CompletePack(cost,weight);

else

{

int k=1;

while(k<amount)

{

ZeroOnePack(k\*cost,k\*weight);

amount-=k;

k<<=1;

}

ZeroOnePack(amount\*cost,amount\*weight);//这个不要忘记了，经常掉了

}

}

int main()

{

int T;

scanf("%d",&T);

while(T--)

{

//这个dp的初始化一定不要忘记，可以不装满则初始化为0，

//否则dp[0]=0,其余的为-INF

memset(dp,0,sizeof(dp));

scanf("%d%d",&nValue,&nKind);

for(int i=0;i<nKind;i++)

scanf("%d%d%d",&value[i],&weight[i],&bag[i]);

for(int i=0;i<nKind;i++)

MultiplePack(value[i],weight[i],bag[i]);

printf("%d\n",dp[nValue]);

}

return 0;

}

## NTT（新）

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

const int MAX=2e6+5;

/\*

模版部分开始：

\*/

const ll MOD=998244353;

const ll g=3;//模MOD的原根

inline ll mul(ll x,ll y,ll mod=MOD){return x\*y%mod;}

inline ll poww(ll a,ll b,ll mod=MOD)

{

a%=mod;

ll ret=1;

for(;b;b>>=1LL,a=mul(a,a,mod))

if(b&1)ret=mul(ret,a,mod);

return ret;

}

void ntt(ll \*A,int len,int inv=1)//A下标0开始 inv=1 DFT inv=-1 IDFT 想改成vector只需要ll \*A 改成vector<ll>&A

{

int k;

for(int i=1,j=len/2;i<len-1;i++)

{

if(i<j)swap(A[i],A[j]);

k=len/2;

while(j>=k)

{

j-=k;k/=2;

}

if(j<k)j+=k;

}

for(int i=2;i<=len;i<<=1)

{

ll wn=poww(g,(MOD-1)/i);

if(inv==-1)wn=poww(wn,MOD-2);

for(int j=0;j<len;j+=i)

{

ll w=1;

for(k=j;k<(j+i/2);k++)

{

ll a=A[k],b=mul(w,A[k+i/2]);

A[k]=(a+b)%MOD;

A[k+i/2]=(a-b+MOD)%MOD;

w=mul(w,wn);

}

}

}

if(inv==-1){

ll vn=poww(len,MOD-2);

for(int i=0;i<len;i++)A[i]=mul(A[i],vn);

}

}

/\*

模版部分结束

以下为使用示例 忽略对x的处理即为 (1+x)^n的系数异或和

\*/

char a[MAX];

int k;

ll ans,A[MAX],n;

int main()

{

scanf("%s%d",a,&k);

int len=strlen(a);

for(int i=0;i<len;i++)

n=(n\*10+a[i]-'0')%(MOD-1);

A[0]=A[1]=1;

ntt(A,k,1);

for(int i=0;i<k;i++)A[i]=poww(A[i],n);

ntt(A,k,-1);

for(int i=0;i<k;i++)

ans^=A[i];

printf("%lld\n",ans);

return 0;

}

# 字符串

## hash和hashset

/\*

M为要模的值

hashset 检验是否有相同（两个hash检验）（与set功能同 但有更好的复杂度？ 采用的方式与手写vector大致相同）

\*/

struct hashset{

const static int seed=1e7+7;//模数 不能太大

const static int maxn=2e6+7;//数组大小(总共结点个数）

struct node{

int x,y;

int next;

node(){};

node(int \_x,int \_y,int n):x(\_x),y(\_y),next(n){};

}T[maxn];//更好地空间局部性?(雾)

int head[seed],size;

void clear(){//初始化

memset(head,-1,sizeof(head));

size=0;

}

void insert(int x,int y){//插入

int& h=head[x%seed];

for (int i=h;~i;i=T[i].next)

if (T[i].x==x&&T[i].y==y) return;

T[size]=node(x,y,h);h=size++;

}

bool count(int x,int y){//是否存在

for (int i=head[x%seed];~i;i=T[i].next)

if (T[i].x==x&&T[i].y==y) return 1;

return 0;

}

}have;

/\*

hash操作函数

\*/

struct hash{

int px[maxn],val[maxn],p;//px存p的i次方 val存字符串的前缀hash值 p为hash使用的质数

void setp(int P,int n=2000000){//设置hash采用的p并进行预处理

int i;px[0]=1;p=P;

for(int i=1;i<=n;i++) px[i]=(ll)px[i-1]\*p%M;

}

void set(char a[],int n){//对长度为n的字符串hash

int i;val[0]=0;

for(int i=1;i<=n;i++) val[i]=((ll)val[i-1]\*p+a[i-1])%M;

}

int get(int l,int r){//获得[l,r]区间的hash值 注意这里l从1开始 即原字符串的起始位置0在此为[1,……]

int ret=val[r]-(ll)val[l-1]\*px[r-l+1]%M;

(ret<0)&&(ret+=M);return ret;

}

};

/\*

使用举例

\*/

hash HA,HB;

char a[maxn],b[maxn];

int i,n,m;

int main(){

scanf("%s",a);

int n=strlen(a);

REP(i,n) a[i+n]=a[i];

HA.setp(27);HB.setp(29);

HA.set(a,n+n);HB.set(a,n+n);

have.clear();

FOR(i,1,n) have.insert(HA.get(i,i+n-1),HB.get(i,i+n-1));

scanf("%d",&m);

REP(i,m){

int i,ans=0;

scanf("%s",b);

int m=strlen(b);

HA.set(b,m);HB.set(b,m);

FOR(i,1,m-n+1) if (have.count(HA.get(i,i+n-1),HB.get(i,i+n-1))) ans++;

printf("%d\n",ans);

}

return 0;

}

# 杂

## 真·快速读入

namespace fastIO {

#define BUF\_SIZE 100000

//fread -> read

bool IOerror = 0;

inline char nc() {

static char buf[BUF\_SIZE], \*p1 = buf + BUF\_SIZE, \*pend = buf + BUF\_SIZE;

if(p1 == pend) {

p1 = buf;

pend = buf + fread(buf, 1, BUF\_SIZE, stdin);

if(pend == p1) {

IOerror = 1;

return -1;

}

}

return \*p1++;

}

inline bool blank(char ch) {

return ch == ' ' || ch == '\n' || ch == '\r' || ch == '\t';

}

inline void read(int &x) {

char ch;

while(blank(ch = nc()));

if(IOerror)

return;

for(x = ch - '0'; (ch = nc()) >= '0' && ch <= '9'; x = x \* 10 + ch - '0');

}

#undef BUF\_SIZE

};

## 头文件

#include <cstdio>

#include <iostream>

#include <algorithm>

#include <vector>

#include <set>

#include <list>

#include <map>

#include <string>

#include <cstring>

#include <stack>

#include <queue>

#include <cmath>

#include <ctime>

#include <bitset>

#include <utility>

#include <complex>

#include <assert.h>

#include <limits.h>

//#include <bits/stdc++.h>

using namespace std;

#define rank rankk

#define mp make\_pair

#define pb push\_back

#define xo(a,b) ((b)&1?(a):0)

#define tm tmp

//#define LL ll

typedef unsigned long long ull;

typedef pair<int,int> pii;

typedef long long ll;

typedef pair<ll,int> pli;

typedef pair<int,ll>pil;

typedef pair<ll,ll> pll;

//const double INF=1e20;

const int INF=0x3f3f3f3f;

//const int INF= 0x7fffffff;

//const ll INF=0x3f3f3f3f3f3f3f3fll;

const ll INFF=0x3f3f3f3f3f3f3fll;

//const ll INFF=1e14+5;

const int MAX=1e2+5;

//const ll MAXN=2e8;

//const int MAX\_N=MAX;

//const ll MOD=1e9+7;

const int MOD=1e9+7;

//const long double pi=acos(-1.0);

//const double eps=0.00000001;

int gcd(int a,int b){return b?gcd(b,a%b):a;}

template<typename T>inline T abs(T a) {return a>0?a:-a;}

template<class T> inline

void read(T& num) {

bool start=false,neg=false;

char c;

num=0;

while((c=getchar())!=EOF) {

if(c=='-') start=neg=true;

else if(c>='0' && c<='9') {

start=true;

num=num\*10+c-'0';

} else if(start) break;

}

if(neg) num=-num;

}

inline ll powMM(ll a,ll b,ll M){

ll ret=1;

a%=M;

// b%=M;

while (b){

if (b&1) ret=ret\*a%M;

b>>=1;

a=a\*a%M;

}

return ret;

}

//const long double eps=-1.0;

//clock\_t t1 = clock();

//fprintf(stderr, "%ld ms\n", clock() - t1);