Bitmask DP (No of Permutation of distinct number)

```
/*
=======[ Theme ]=======
LightOj: Painful Bases.

you are given a base, an integer K
and
a valid number in the base which
contains distinct digits.
output number of permutations of the
given number
which are divisible by K. K is given
in decimal.
*/
long long dp[66000][20];
char a[20];
int n,k,l;

int modul(char ch,int carry) {
    int x;
    if(ch>='0' &&ch<='9') x=ch-'0';
    else  x=ch-'A'+10;
    x+=carry*n;
    x=x%k;
    return x;
}

long long call(int pos,int mod) {
    if(pos==(1<<l)-1) {
        if(mod==0) return 1;
        else return 0;
    }
    long long &t=dp[pos][mod];
    if(t!=-1) return t;
    long long sum=0;
    for(int i=0;i<l;i++) {
        if(!(pos&1<<i)) {
            int x=modul(a[i],mod);
            sum+=call(pos|1<<i,x);
        }
    }
    t=sum;
    return sum;
}

main() {
    int tc,t=1;
    scanf("%d",&tc);
    while(tc--) {
        scanf("%d%d",&n,&k);
        getchar();
        gets(a);
        memset(dp,-1,sizeof(dp));
        l=strlen(a);
        long long sum=call(0,0);
        printf("Case %d: %lld\n"
                            ,t++,sum);
    }
    return 0;
}
```

Bitmask DP (No of Permutation of repeat numbers)

```
/*
=======[ Theme ]=======
LightOj: Anagram Division

Given a string s and a positive integer d,
determine how many permutations of s are
divisible by d.
Numbers can be repeated.
*/
int dp[1050][1050];
char arr[20];
int n,N;

int go(int mask,int mod) {
    if(mask==(1<<N)-1) {
        if(mod==0)
            return 1;
        else
            return 0;
    }
    int &t=dp[mask][mod];
    if(t!=-1)
        return t;
    int sum=0;
    int i,k;
    for(i=0; i<N; i++)
    {
        if(!(mask&1<<i))
            if(i==0||arr[i]!=arr[i-1]
                    ||(mask&1<<(i-1)))
            {
                sum+=go(mask|1<<i,
                  (mod*10+arr[i]-'0')%n);
            }
    }
    t=sum;
    return sum;
}

main()
{
    int t=1,tc;
    scanf("%d",&tc);
    while(tc--)
    {
        scanf("%s %d",&arr,&n);
        N=strlen(arr);
        sort(arr,arr+N);
        memset(dp,-1,sizeof(dp));
        int sum=go(0,0);
        printf("Case %d: %d\n",t++,sum);
    }
    return 0;
}
```

DP (no of distinc LCS)

```
/*
=======[ Theme ]=======
LightOj: LCS Revisited

find the number of distinct LCS of s and
t. modulo 1000007.
*/
int dp[1010][1010];
int dp2[1010][1010];
char a[1010],b[1010];
int  mid,ans,sum;
int N,K,S;
int LCS(int i,int j){
    if(i<0||j<0) return 0;
    int &t=dp2[i][j];
    if(t!=-1) return t;
    int ret=0;
    if(a[i]==b[j]) ret=LCS(i-1,j-1)+1;
    else {
        ret=LCS(i,j-1);
        ret=max(ret,LCS(i-1,j));
    }
    t=ret;
    return t;
}

int go(int i,int j) {
    if(i<0||j<0) return 0;
    int &t=dp[i][j];
    if(t!=-1) return t;
    int ret=0;
    if(a[i]==b[j]) {
        if(LCS(i,j)==1)
            return 1;
        else
            ret=go(i-1,j-1);
    }
    else {
        int n=LCS(i-1,j);
        int m=LCS(i,j-1);
        if(n==m) {
            ret=(ret+go(i-1,j)%mod)%mod;
            ret=(ret+go(i,j-1)%mod)%mod;
            if(LCS(i,j)==LCS(i-1,j-1))
                ret=((ret-go(i-1,j-1))
                        %mod+mod)%mod;
        }
        else if(n>m)
            ret=go(i-1,j);
        else if(n<m)
            ret=go(i,j-1);
    }
    t=ret;
    return t;
}

n = strlen(a);
l = strlen(b);
Answer = go(n-1,l-1);
```

DP (No. of ways to the pallindrome)

```
/*
=======[ Theme ]=======
LightOj: The Specials Menu

number of ways he could remove
letters
from a particular word so that it
would become a palindrome.
*/

long long dp[70][70];
int I,J,K,L,N;
char a[70];
long long go(int i,int j)
{
    if(i>j)
    {
        return 0;
    }
    long long &t=dp[i][j];
    if(t!=-1)
        return t;
    long long ret=0;
    ret=go(i+1,j);
    ret+=go(i,j-1);
    ret-=go(i+1,j-1);
    if(a[i]==a[j])
        ret+=go(i+1,j-1)+1;
    t=ret;
    return t;
}

main()
{
    int t=1,tc;
    scanf("%d",&tc);
    getchar();
    while(tc--)
    {
        memset(dp,-1,sizeof(dp));
        int n,k,l,i;
        gets(a);
        l=strlen(a);
        long long sum=go(0,l-1);
        printf("Case %d: %lld\n"
                          ,t++,sum);
    }
    return 0;
}
```

DP-Coin Change (Limited Coin)

```
/*
=======[ Theme ]=======
n types of coins of value A1, A2 ... An.
C1, C2, ... Cn denote the number of
coins of value A1, A2 ... An.
find the number of ways you can make K
using the coins.

*/
int n,k;
int tc[51];
int coin[51];
long long dp[51][1001];

long long check(int i, int amount)
{
    int j;
    long long y=0;
    if(i>=n)
    {
        if(amount==k)
            return 1;
        else
            return 0;
    }
    long long &t=dp[i][amount];
    if(t!=-1)
        return t;

    y=check(i+1,amount)%mod;
    for(j=1;j<=tc[i];j++)
    {
        if(amount+coin[i]*j<=k)
           y+=check(i+1,
                  amount+coin[i]*j)%mod;
    }
    return t=y%mod;
}

main()
{
    int i,j,t;
    long long m;
    scanf("%d",&t);
    for(i=1;i<=t;i++)
    {
        memset(dp,-1,sizeof(dp));
        scanf("%d %d",&n,&k);
        for(j=0;j<n;j++)
            scanf("%d",&coin[j]);
        for(j=0;j<n;j++)
            scanf("%d",&tc[j]);
        m=(check(0,0))%mod;
        printf("Case %d: %lld\n",i,m);
    }
    return 0;
}
```

LCS of K strings

```
/*
Lcs of K strings.
each string initially conatains
element from 1 to n.
1<=K<=5.
Solution: Golam Mazid Vai.

*/

#define eps 1e-9
#define Pi 2*acos(0)
#define inf 1<<30
#define mod 1000000007
#define Sz 100005

int dp[1005][1005];
int A[10][10005],B[10][1005];
int f(int v,int p,int n,int k)
{
    if(p>n)
        return 0;
    if(dp[v][p]!=-1)
        return dp[v][p];
    int c=0,d,i;
    c=f(v,p+1,n,k);
    d=B[1][p];
    for(i=2,i<=k;i++)
    {
        if(A[i][d]>A[i][v])
            continue;
        break;
    }
    if(i>k)
        c=Max(c,f(d,p+1,n,k)+1);
    return dp[v][p]=c;
}
int main()
{
    LL n,k,i,j,a;
    while(cin>>n>>k)
    {
        mem(dp,-1);
        for(i=1,i<=k;i++)
        {
            for(j=1,j<=n;j++)
            {
                cin>>a;
                A[i][a]=j;
                B[i][j]=a;
            }
        }
        printf("%d\n",f(0,1,n,k));
    }
    return 0;
}
```

BIT (All possible Increasing Subsequences)

```
/*
An increasing subsequence from a
sequence A1, A2 ... An is defined by
Ai1, Ai2 ... Aik,
where the following properties hold
1.      i1 < i2 < i3 < ... < ik and
2.      Ai1 < Ai2 < Ai3 < ... < Aik
Now you are given a sequence, you have
to find the number of all possible
increasing subsequences.

*/

ll MaxVal;

ll tree[NN];
pair<ll,ll> a[NN];
ll b[NN];

ll update(ll idx,ll val)
{
    while(idx<=MaxVal)
    {
        tree[idx]=(tree[idx]+val)%mod;
        idx+=idx & (-idx);
    }
    return 0;
}

ll query(ll idx)
{
    ll sum=0;
    while(idx>0)
    {
        sum=(sum+tree[idx])%mod;
        idx-=idx & (-idx);
    }
    return sum;
}
```

```
main()
{
    int t=1,tc;
    cin>>tc;

    ll i,j,k,l,m,n;
    while(tc--)
    {
        cin>>n;

        for(i=1;i<=n;i++)
        {
            cin>>a[i].ft;
            a[i].second=i;
        }
        sort(a+1,a+n+1);
        memset(tree,0,sizeof(ll)*(n+2));

        b[0]=0;
        b[a[1].sd]=1;
        for(i=2;i<=n;i++)
        {
            if(a[i].ft==a[i-1].ft)
                b[a[i].sd]=b[a[i-1].sd];
            else
               b[a[i].sd]=b[a[i-1].sd]+1;
        }

        ll ret=0;
        MaxVal=b[a[n].sd];
        for(i=1;i<=n;i++)
        {
            ret=(query(b[i]-1)+1)%mod;
            update(b[i],ret);
        }
        printf("Case %d: %lld\n"
                      ,t++,query(MaxVal));
    }
    return 0;
}
```

the_redback

Code Library

Maruf Tuhin
CUET CSE 11
maruf.2hin@gmail.com

By Maruf_Tuhin