第 1 章 atcode058

1.1 arc058

1.1.1 c-Iroha's Obsession

暴力, 注意考虑边界

```
#include<bits/stdc++.h>
using namespace std;
int arr[15];
int N,k;int ans=0;
int check(int x){
    while(x){
        if(arr[x%10]==1){
            return 0;
        }
        x/=10;
    }
    return 1;
}
int main(){
    for(int i=0;i<15;i++){
        arr[i]=0;
    }
    scanf("%d%d",&N,&k);
    for(int i=0;i<k;i++){
        int temp;
        scanf("%d",&temp);
        arr[temp]=1;
    }
    for(int i=N;i<=100010;i++){
        if(check(i)){
            printf("%d\n",i);
            //system("pause");
            return 0;
        }
    }
}
```

1.1.2 d-Iroha and a Grid

题意:n*m 矩阵左下方 A*B 为禁区, 每次可以走下或者左 n,m<=1e5, 问从左上到右下不经过禁区时的方案数?

从左上角到右下角所有方案数为 $\binom{n+m-2}{n-1}$, 一个 n*m 的方格，向下走 n-1 步，向右走 m-1 步，只要向下 (向右) 走了 n-1(m-1) 步后，上下的没有选则。然后对径拆分。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=2e5+20;
const ll mod=1e9+7;
ll f[N],n,m,A,B;
```

```
ll powmod(ll x,ll n){
    ll s=1;
    while(n){
        if(n&1)
            s=(s*x)%mod;
        x=(x*x)%mod;
        n>>=1;
    }
    return s;
}
ll C(ll n,ll m){
    ll a=f[n];
    ll b=(f[m]*f[n-m])%mod;
    return (a*powmod(b,mod-2))%mod;
}
int main(){
    f[0]=1;
    for(ll i=1;i<N;i++)
        f[i]=(f[i-1]*i)%mod;
    while(cin>>n>>m>>A>>B){
        ll res=0;
        for(ll i=B+1;i<=m;i++){
            ll tmp=(C(i-1+n-A-1,n-A-1)*C(m-i+A-1,m-i))%mod;
            res=(res+tmp)%mod;
        }
        cout<<res<<endl;
    }
    return 0;
}
/*
2 3 1 1
2
*/
```

1.1.3 e-Iroha and a Grid

给出你三个数 x,y,z 还有 N，有 N 个位置，位置用来添加数字，要求连续位置上的数的和能够得到 x,y,z 不要求所有位置都用得到，求出所有的满足题意的序列的种类数，对 1e9+7 取模

几个关键点，1:5 表示成 10000，在然加一个 1，用同样的方法看可以看成几个数合成了 5，如 10100 是 2+3=5，这样只需要看最关键的几个点就能判断能否构成合法序列。2，转移方程，不可能在三步转移出四个 1 的情况，所以 dp[3][(111101)] 一定为 0；

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int mod=1e9+7;
int dp[50][1<<17];
int ans=1;
ll S,fow;
int main(){
    int n;scanf("%d",&n);
    int x,y,z;scanf("%d%d%d",&x,&y,&z);
    for(int i=0;i<n;i++){
        ans=1ll*ans*10%mod;
```

```
    }
    S=(1<<x+y+z)-1;
    fow=(1<<x-1)|(1<<x+y-1)|(1<<x+y+z-1);
    dp[0][0]=1;
    for(int i=1;i<=n;i++){
        for(int j=0;j<=S;j++){
            for(int k=1;k<=10;k++){
                int T=(j<<k)|(1<<k-1);
                T&=S;
                if((T&fow)!=fow)
                    dp[i][T]=(dp[i][T]+dp[i-1][j])%mod;
            }
        }
    }
    for(int i=0;i<=S;i++){
        ans=(ans-dp[n][i]+mod)%mod;
    }
    printf("%d\n",ans);
    //system("pause");
    return 0;
}
```

第 2 章 atcode

2.1 arc059

2.1.1 c-Be Together

自定义了一个花费，求最小花费。注意平均数可能会大 1，直接加 1 两个都判断。

```
#include<bits/stdc++.h>
using namespace std;
const int N=1e2+50;
int arr[N];
int main(){
    int n;scanf("%d",&n);
    int sum=0;
    for(int i=0;i<n;i++){
        scanf("%d",&arr[i]);
        sum+=arr[i];
    }
    int temp1=sum/n;
    int temp2=sum/n+1;
    int ans1=0;
    int ans2=0;
    for(int i=0;i<n;i++){
        ans1+=(arr[i]-temp1)*(arr[i]-temp1);
        ans2+=(arr[i]-temp2)*(arr[i]-temp2);
    }
    printf("%d\n",min(ans1,ans2));
    //system("pause");
}
```

2.1.2 d-Be Together

给定一个序列，如果某个子序列中的有一个字母的数量大于子序列长度的一半则为平衡序列。

鸽巢原理，不可能全部相隔两个以上 d

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    std::ios::sync_with_stdio(false);
    cin.tie(0);
    int flag=0;
    string s;
    cin>>s;
    for(int i=0;i<s.size()-1;i++){
        if(s[i]==s[i+1]){
            cout<<i+1<<"␣"<<i+2<<endl;
            flag=1;
            break;
        }
        else if(i<s.size()-2&&s[i]==s[i+2]){
            cout<<i+1<<"␣"<<i+3<<endl;
            flag=1;
```

```
            break;
        }
    }
    if(!flag)
    cout<<-1<<"␣"<<-1<<endl;
    return 0;
}
```

2.1.3 e-Children and Candies

给一个人 x 个糖果，对应的方案乘上 $\sum_{i=a[i]}^{b[i]} i^m$，所以转移是 $O(n^4)$ 不过因为 a_i, b_i 固定，前缀和优化一下就好，最后是 $O(n^3)$，取模有减法，注意。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=410;
const ll mod = 1e9+7 ;
ll a[N],b[N];
ll sum[N][N], l[N][N];
ll dp[N][N];
int main(){
    ll n,m;
    scanf("%lld%lld",&n, &m);
    for(int i=1;i<=n;i++){
        scanf("%lld",&a[i]);
    }
    for(int i=1;i<=n;i++){
        scanf("%lld",&b[i]);
    }
    for(int i=0; i<N; i++){
        l[i][0]=1;
        for(int j=1; j<N; j++){
            l[i][j]=(l[i][j-1]*i)%mod;
        }
    }
    for(int i=1; i<N; i++){
        for(int j=0; j<N; j++){
            sum[i][j]=(sum[i][j]+sum[i-1][j]+l[i][j])%mod;
        }
    }
    dp[0][0]=1;
    for(int i=1;i<=n;i++){
        for(int j=0;j<=m;j++){
            for(int k=0;k<=j;k++){
                dp[i][j]=((dp[i][j]%mod+(1ll*dp[i-1][j-k]*(sum[b[i]][k]-sum[a[i]-1][k]))%mod+mod)%mod+mod)%mod
     ;
            }
        }
    }
    printf("%lld\n",(dp[n][m]+mod)%mod);
    //system("pause");
    return 0;
}
```

2.1.4 f-Unhappy Hacking

设 f[i][j] 表示按了 i 次键盘，然后一共出现了 j 个字母的情况, 用 f[i][j] 去更新别人... 所以就转移两次就行了,f[i+1][j+1]+=2 f[i][j] （因为可以按 0 或者 1，所以要乘 2）,f[i+1][min(j-1,0)]+=f[i][j]（这里是按后退键的情况，因为 0 也是合法状态所以要算进去，取个 min）, 但是这样子算出来的 f[n][len] 是能够用 n 次操作拼出长度为 len 的字符串的方法总数（一共可以拼出 2^len 个串），所以要除一下 2^len

```
#include <bits/stdc++.h>
using namespace std;
#define N 5010
#define ll long long
const int mod = 1e9 + 7;
int n;
char s[N];
int f[N][N];
//f[ i ][ j ] 表示按了i次键盘，出现了j个字母
ll power(ll a, ll b)
{
    ll ans = 1, base = a;
    while (b) {
        if (b & 1)
            ans = (ans * base) % mod;
        base = (base * base) % mod;
        b >>= 1;
    }
    return ans % mod;
}
int main()
{
    scanf("%d%s", &n, s + 1);
    int len = strlen(s + 1);
    f[0][0] = 1;
    for (int i = 0; i <= n; i++) {
        for (int j = 0; j <= i; j++) {
            f[i + 1][j + 1] += 2 * f[i][j];
            f[i + 1][j + 1] %= mod;
            f[i + 1][max(j - 1, 0)] += f[i][j];
            f[i + 1][max(j - 1, 0)] %= mod;
        }
    }
    ll ans = 1ll * power(power(2, len), mod - 2) % mod * f[n][len];
    ans %= mod;
    printf("%lld", ans);
    return 0;
}
```

第 3 章　atcode060

3.1 arc060

3.1.1 c-Tak and Cards

求一串数字中不连续子串使其总和是 a 的倍数的个数

类似于背包 dp，转移方程 $\begin{cases} dp[0][0]=1 \\ dp[i][j]=dp[i][j]+dp[i-1][j-arr[k]] \end{cases}$ i 为 i 个数，j 为 i 个数的和

```
#include<bits/stdc++.h>
typedef long long ll;
using namespace std;
const int N=55;
int arr[N];
ll sum[N];
ll dp[N][N*N];
int main(){
    int n,A;
    sum[0]=0;
    scanf("%d%d",&n,&A);
    for(int i=1;i<=n;i++){
        scanf("%d",&arr[i]);
        sum[i]=sum[i-1]+arr[i];
    }
    dp[0][0]=1;
    for(int k=1;k<=n;k++){
        for(int i=k;i>0;i--){
            for(int j=sum[k];j>=arr[k];j--){
                dp[i][j]+=dp[i-1][j-arr[k]];
            }
        }
    }
    ll ans=0;
    for(int i=1;i<=n;i++){
        ans+=dp[i][i*A];
    }
    printf("%lld\n",ans);
    system("pause");
}
/*
4 8
7 9 8 9
5
*/
```

3.1.2 b-Digit Sum

给一个函数 $f(b,n)$ 可以求出 n 在 b 进制下各位数的和，设 $f(b,n)=s$，现在已知 n,s，求 b 我们可以分成两种情况来讨论, 当 $b \le \sqrt{n}$，直接枚举小于 $\sqrt{b}$ 的所有因

子，如果满足式子 $fun(b,n)=s$ 所以直接输出。如果 $b \geq \sqrt{n}$ 则么，则满足式子

$$x_0 + x_1 * b^1 + x_2 * b^2 + \cdots + x_m * b^m = n$$

$$x_0 + x_1 + x_2 + \cdots + x_m = s$$

可以推出

$$n = p * b + q$$

$$s = p + q$$

解得

$$b = \frac{n-s}{p} + 1$$

再枚举 n-s 的因子，最好两个因子同时判断，速度更快，不过枚举之后要更新答案的时候记得再套一遍 $f(b,n)$ 来检查一下。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
ll fun(ll b,ll n){
    return n<b ? n : fun( b , n / b ) + n % b;
}
int main(){
    ll n,s;
    scanf("%lld%lld",&n,&s);
    if( s > n )
        return puts( "-1" ) , 0 ;
    if( s == n )
        return printf("%lld\n",n+1),0;
    ll m=sqrt(n);
    for(ll i=2; i<=m; i++){
        if(fun(i,n)==s){
            printf("%lld\n",i);
            return 0;
        }
    }
    ll ans=1e11;
    ll ns=n-s;
    for(ll i=1;i*i<=ns;i++){
        if((ns)%i==0){
            ll b=(ns)/i+1;
            if(fun(b,n)==s){
                ans=min(ans,b);
            }
            if(fun(i+1,n)==s){
                ans=min(ans,b);
            }
        }
    }
    printf("%lld\n",ans!=1e11?ans:-1);
    return 0;
}
```

3.1.3 e-Tak and Hotels

题意给出 n 个点，每个点的值表示坐标轴的位置，每天最多移动 L 单位，必须在点停留，问从第几个点到另一个点需要多久

首先用二分求出每个点每一天能够走到哪个点（贪心地想，能走更远肯定走更远，至于如果走过了，肯定在那天的一半的时候就走到目的地了，所以时间花费还是一样的）考虑优化，想了好久忽然想起弹飞绵羊那题，这题除了没有修改操作其实是和弹飞绵羊差不多的所以大力分块！二分预处理不变，然后对原序列分块。再维护一下每个点跳出该块后在哪以及跳出该块要花多少天在查询的时候就可以一个块一个块的跳跳到目的地所在的那一块，剩下的路径暴力就可以

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 10;
int n, L, m;
int a[N], to[N];
int block, num, ans;
int nxt[N], val[N], belong[N];
int find(int x)
{
    int l = x + 1, r = n, ans = 0;
    while (l <= r) {
        int mid = (l + r) >> 1;
        if (a[mid] - a[x] <= L)
            l = mid + 1, ans = mid;
        else
            r = mid - 1;
    }
    return ans;
}
int main()
{
    scanf("%d", &n);
    for (int i = 1; i <= n; i++) {
        scanf("%d", &a[i]);
    }
    sort(a + 1, a + n + 1);
    scanf("%d%d", &L, &m);
    for (int i = 1; i < n; i++) {
        to[i] = find(i);
    }
    to[n] = n + 1;
    // for(int i=1;i<=n;i++){
    //     cout<<to[i]<<" ";
    // }
    block = sqrt(n);
    num = n / block;
    if (n % block)
        num++;
    for (int i = 1; i <= n; i++) {
        belong[i] = (i - 1) / block + 1;
    }
    for (int i = 1; i < n; i++) {
        int t = i, sp = 0;
        while (belong[t] == belong[i]) {
            t = to[t];
```

```
            sp++;
        }
        val[i] = sp;
        nxt[i] = t;
        if (i >= block * (num - 1) + 1) {
            val[i]--;
            nxt[i]--;
        }
    }
    for (int i = 1; i <= m; i++) {
        int x, y;
        ans = 0;
        scanf("%d%d", &x, &y);
        if (x > y)
            swap(x, y);
        while (belong[x] < belong[y]) {
            ans += val[x];
            x = nxt[x];
        }
        while (x < y) {
            x = to[x];
            ans++;
        }
        printf("%d\n", ans);
    }
    //system("pause");
}
```

还有一种倍增的解法 f[x][y] 表示第 x 个点 2^y 天数能到的最远点。

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn = 2e5 + 10;
ll a[maxn];
ll f[maxn][35];
int main()
{
    int n, step, q, i, j, p;
    scanf("%d", &n);
    for (i = 0; i < n; i++)
        scanf("%lld", &a[i]);
    scanf("%d%d", &step, &q);
    for (i = 0; i < n; i++) {
        p = upper_bound(a, a + n, a[i] + step) - a;
        f[i + 1][0] = p;
    }
    for (int j = 1; j <= 30; j++)
        for (int i = 1; i <= n; i++)
            f[i][j] = f[f[i][j - 1]][j - 1];
    int l, r;
    while (q--) {
        scanf("%d%d", &l, &r);
        if (l > r)
            swap(l, r);
        ll ans = 0;
        for (int i = 30; i >= 0; i--) {
            if (f[l][i] < r) {
                ans += (ll)(1 << i);
                l = f[l][i];
```

```
            }
        }
        printf("%lld\n", ans + 1);
    }
    return 0;
}
/*
9
1 3 6 13 15 18 19 29 31
10
4
1 8
7 3
6 7
8 5

4
2
1
2
*/
```