S ---【云智计划】--- 4月3日 --- 模拟测#6【温岭中学】

T1 -- 取模

解法：

- **考察知识点**：组合数学，模运算

  20 分做法，暴力生成每个序列，进行计算，时间复杂度 $O(m^n n^2)$。相信没有人不会吧。60 分留给可能的 $O(n^2)$ 做法，反正我没想到。

  来讲讲正解。不妨先想一想对于一个给定序列 $a_1, a_2, \ldots, a_n$ 应该如何用低于 $O(n^2)$ 的复杂度计算。

  很容易想到根据剩余系分类。对于 $a_i \bmod 2 \equiv 0$ 这类无论和谁加，都可以拆成 $\frac{a_i}{2}$ 出来算。而对于 $a_i \bmod 2 \equiv 1$ 这类，如果是同一类和它相加，就会多一个 1 的贡献。于是想到可以写出这样的计算式

  $$\sum_{i=1}^{n} \left\lfloor \frac{a_i}{2} \right\rfloor (n-1) + \binom{|S_1|}{2}$$

  其中 $|S_1|$ 为给定序列中 $a_i \bmod 2 = 1$ 的 a_i 个数。

  想想怎么推广到对于求所有可能序列和。简单的组合数学即可，答案式为

  $$\left(\sum_{a_i \in [1,m]} \left\lfloor \frac{a_i}{2} \right\rfloor \right) (n-1) \times n m^{n-1} + \sum_{i=2}^{n} \binom{i}{2} t^i (m-t)^{n-i}$$

  其中 t 为 $[1, m]$ 中奇数个数。

Code

```cpp
#include<cstdio>
//#define int ll
typedef long long ll;
const int mod=998244353;
int fac[1000005],inv[1000005];
int pow(int x,int p) {
    int res=1;
    for(;p;p>>=1) {
        if(p&1) res=res*x%mod;
        x=x*x%mod;
    }
    return res;
}
signed main() {
    int n,m; scanf("%d%d",&n,&m);
    int odd=m/2; fac[0]=inv[0]=1;
    int inv2=pow(2,mod-2),x=(1+m/2)*(m/2)%mod;
    for(int i=1;i<=n;++i) {
        fac[i]=fac[i-1]*i%mod;
        inv[i]=inv[i-1]*pow(i,mod-2)%mod;
    }
    if(m%2==0) {x-=m/2;}
    else {++odd;}
    int res=0; //printf("%d\n",res);
```

```
    for(int i=2;i<=n;++i) {
        int c2=fac[n]*inv[i]%mod*inv[n-i]%mod;
        res+=i*(i-1)%mod*pow(odd,i)%mod*pow(m-odd,n-i)%mod*c2%mod;
        res%=mod;
    }
    res=res*inv2%mod;
    res+=n*(pow(m,n-1)*(n-1)%mod*x%mod)%mod;
    res%=mod;
    printf("%lld\n",res);
    //printf("%lld\n",inv2*(pow(m,n-1)*(n-1)%mod*x%mod+))
    return 0;
}
```

T2 -- 魔法

解法：

- **考察知识点**：优化建图，`bitset` 优化，区间最值，单调栈

  自然有一个很 `naive` 的做法，直接对于题意建图然后跑一个 $O(n(n+m))$ 的 bfs，但是边数 $m=n^2$ 会 TLE。别问我区间长度和为什么是 n^2 级别，你想想排列 `1 2 3 4 5 6 ...` 的区间长度和是多少（笑

  不妨先不考虑魔力值相同的情况。在魔力值互不相同的情况下，建出来的图自然是一些 DAG。注意到我们只需要考虑可达性，去掉一些不必要的边自然是不影响可达性的。应该保留哪些边呢？这里直接给出结论，对于每个 i，只需要保留 $i<j<R$ 当中 a_j 最大的 j 和 $L<j<i$ 中 a_j 最大的 j。

  存在魔力值相同的情况呢？由于相同魔力值可以互相到达，因此可以将魔力值相同的看作一个点，我们对于魔力值建图而不是编号建图。可以使用 bitset 加速上述过程，时间复杂度为 $O\left(n\log n+\frac{n^2}{\omega}\right)$。

Code

```cpp
/* stuff you should look for
    * int overflow, array bounds, uppercase/lowercase
    * special cases (n=1?)
    * do sth. instead of nothing and stay organized
    * WRITE STUFF DOWN
    * DON'T GET STUCK ON ONE APPROACH
*/
#include<cstdio>
#include<bitset>
#include<queue>
#include<vector>
#include<algorithm>
int cnt=0;
struct node {
    int x,y;
}e[40005];
std::bitset<20005> vis[20005];
std::vector<int> pos[20005];
int a[20005],st[20005],L[20005],R[20005],col[20005],stt[16][20005];
int lg[20005],du[20005],h[20005],to[40005],ver[40005];
inline int read() {
    register int x=0,f=1;register char s=getchar();
    while(s>'9'||s<'0') {if(s=='-') f=-1;s=getchar();}
    while(s>='0'&&s<='9') {x=x*10+s-'0';s=getchar();}
    return x*f;
}
inline bool cmp(const node &lhs,const node &rhs) {return lhs.x!=rhs.x?
lhs.x<rhs.x:lhs.y<rhs.y;}
inline void add(int x,int y) {++du[y]; to[++cnt]=y; ver[cnt]=h[x];
h[x]=cnt;}
inline int chkmax(const int &x,const int &y) {return a[x]>a[y]? x:y;}
```

```
inline int ask(int l,int r) {
    if(l>r) return -1;
    int bit=lg[r-l+1];
    return chkmax(stt[bit][l],stt[bit][r-(1<<bit)+1]);
}
inline void topo(int n) {
    std::queue<int> q;
    for(int i=1;i<=n;++i) {
        vis[i].set(i);
        if(!du[i]) q.push(i);
    }
    while(q.size()) {
        int x=q.front(); q.pop();
        for(int i=h[x];i;i=ver[i]) {
            int y=to[i]; vis[y]|=vis[x];
            if(!(--du[y])) q.push(y);
        }
    }
}
int main() {
//  freopen("magic8.in","r",stdin);?
//  freopen("magic8.out","w",stdout);
    int n=read();
    lg[0]=-1; for(int i=1;i<=n;++i) lg[i]=lg[i>>1]+1;
    for(int i=1;i<=n;++i) pos[a[i]=read()].push_back(i),stt[0][i]=i;
    int top=0; st[top=1]=0;
    for(int i=1;i<=n;++i) {
        while(top>1&&a[st[top]]<a[i]) --top;
        L[i]=st[top]; st[++top]=i;
    }
    st[top=1]=n+1;
    for(int i=n;i>=1;--i) {
        while(top>1&&a[st[top]]<a[i]) --top;
        R[i]=st[top]; st[++top]=i;
    }
    for(int bit=1;bit<=15&&(1<<bit)<=n;++bit) {
        for(int i=1;i+(1<<bit)-1<=n;++i) {
            stt[bit][i]=chkmax(stt[bit-1][i],stt[bit-1][i+(1<<(bit-1))]);
        }
    }
    int tot=0;
    for(int i=1;i<=n;++i) {
        int tt=ask(L[i]+1,i-1); if(~tt) e[++tot]=(node){a[tt],a[i]};
        tt=ask(i+1,R[i]-1); if(~tt) e[++tot]=(node)
{a[tt],a[i]};//add(tt,i);
//      printf("%d %d %d\n",tt,L[i],R[i]);
//      if(~tt) add(tt,i); printf("%d->%d\n",i,tt);
//      tt=ask(i+1,R[i]-1); if(~tt) add(tt,i); printf("%d->%d\n",i,tt);
    }
    std::sort(e+1,e+1+tot,cmp);
    for(int i=1;i<=tot;) {
        add(e[i].x,e[i].y);
        int cur=i+1;
        while(cur<=tot&&e[cur].x==e[i].x&&e[cur].y==e[i].y) ++cur;
        i=cur;
```

```
    }
    topo(n);
    int Q=read();
    while(Q--) {
        int p=read(),q=read();
        printf("%s\n",vis[a[p]][a[q]]? "YES":"NO");
    }
//  for(int i=1;i<=n;++i) {
//      for(int j=1;j<=n;++j) {
//          printf("%d ",(int)vis[i][j]);
//      }
//      printf("\n");
//  }
//  for(int i=1;i<=n;++i) {
//      if(pos[i].size()) {
//          col[i]=pos[i][0];
//          for(int y:pos[i]) printf("%d ",y); printf("\n");
//          for(int j=1;j<pos[i].size();++j) {
//              vis[col[i]]|=vis[pos[i][j]];
//          }
//      }
//  }
//  for(int i=1;i<=n;++i) {
//      for(int j=1;j<=n;++j) {
//          printf("%d ",(int)vis[i][j]);
//      }
//      printf("\n");
//  }
//  int Q=read();
//  while(Q--) {
//      int p=read(),q=read();
//      printf("%s\n",vis[col[a[p]]][q]? "YES":"NO");
//  }
    return 0;
}

```

T3 -- 数列删除

解法:

至少删除 m 个数，意思就是最多保留 $n-m$ 个数。

删除的总和最小，意思就是保留的总和最大。

非降子序列问题可以用经典的动态规划来解决。

用 $f[i][j]$ 表示，当前选的最后一个数是 $a[i]$，一共选了 j 个数，选的数总和最大是多少。

转移就是枚举上一个数 $a[k]$，满足 $k<i\&\&a[k]\le a[i]$，$f[i][j]$ 可以用 $f[k][j-1]+a[i]$ 转移。

Code

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

using namespace std;

int f[1010][1010];
int a[1010];
int main() {
    int n, m, i, j, k, sum = 0, ans = 0;
    cin >> n >> m;
    for ( i = 1 ; i <= n ; i++)
        cin >> a[i], sum += a[i];

    for ( i = 1; i <= n; i++ ) {
        f[i][1] = a[i];
        for (j = 2; j <= n-m; j++)
            for (k = 1; k < i; k++)
                if (a[k] <= a[i])
                    f[i][j] = max(f[i][j], f[k][j-1]+a[i]);

        for (j = 1; j <= n-m; j++)
            ans=max(ans, f[i][j]);
    }

    cout<<sum-ans;


    return 0;
}
```

T4 -- 游戏升级

解法：

对应题目数据范围：

$10pts$：输出 0 即可。

$20pts$：暴力即可。

$20pts$：$x > A_1$ 时小明无法升级，暴力枚举小于等于 A_1 的 x，大于 A_1 的整体处理。

$20pts$：即 $\lfloor \frac{A_1}{x} \rfloor = \lfloor \frac{A_2}{x} \rfloor$。乱设的部分分，其实是提醒你考虑 $\lfloor \frac{A_1}{x} \rfloor$ 的取值种类。

对于 100% 的数据：只需要发现 $\lfloor \frac{A_i}{x} \rfloor$ 的取值种类只有 $O(\sqrt{A_i})$ 种。然后这题就没了，可以用类似整除分块的写法求出有多少组这样的取值。复杂度 $O(T\sqrt{A})$。

难题

测试点 $1,2$

暴力枚举初始的 x,y，然后不断进行 $x=x+y,y=x+y$ 判断是否存在某一时刻 $x=X$。

测试点 $3,4$

只枚举初始 x 的取值，设 $y=k$，然后带入 $x=x+y,y=x+y$ 的过程中，每个时刻 x 的值是 $ak+b$ 的形式，其中 a,b 是定值，然后就是判断 $ak+b=X$ 是否有 $k\in[1,M]$ 的解了。

测试点 $5,6$

输出上面的 a,b 可以发现 $a=f_i,b=f_{i+1}$，其中 i 为奇数。

然后就是求 $xf_i+yf_{i+1}=N(x\in[0,N],y\in[0,M])$ 的解的个数，因为 $gcd(f_i,f_{i+1})=1$，可以用扩欧求一个特解 x_0,y_0。满足条件的 $x\equiv x_0\ mod\ f_{i+1},y\equiv y_0\ mod\ f_i$，因此可以找到 x 最大且满足条件的解 (x_1,y_1)，和 x 最小且满足条件的解 (x_2,y_2)，于是可以算出当前情况下，解的个数为 $\frac{x_1-x_2}{f_{i+1}}$。复杂度为 $O(nlog^2X)$。

测试点 $7-10$

输出每一组 x_0,y_0，会发现 $x_0=-f_{i-1},y_0=f_{i-2}$。可用归纳法证明：

假设 $-f_if_{i-1}+f_{i+1}f_{i-2}=1$ 成立，那么

$$\begin{aligned}&-f_{i+1}f_i+f_{i+2}f_{i-1}\\=&-f_{i+1}(f_{i-1}+f_{i-2})+(f_{i+1}+f_i)f_{i-1}\\=&-f_{i+1}f_{i-2}+f_if_{i-1}=-1\end{aligned}$$

所以 $-f_{i+2}f_{i+1}+f_{i+3}f_i=1$（注意上面枚举的 i 为奇数），然后就可以优化掉一个 log。

关于细节

若没开 $\_int128$，或没判断 $x=0$，或只枚举了 70 个斐波那契数，会 WA。

Code

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

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    int T;
    cin >> T;
    while (T--) {
        int a1, b1, a2, b2, n;
        cin >> a1 >> b1 >> a2 >> b2 >> n;
        int ans = 0;
        for (int l = 1, r, i; l <= a1 + 1; l = r + 1) {
            i = a1 / l;
            r = i ? a1 / i : 1e9;
```

```
            int j = i + b1 - b2;
            if (j < 0 || j > a2)
                continue;
            int l2 = a2 / (j + 1) + 1, r2 = j ? a2 / j : 1e9;
            // cerr<<"> "<<i<<" "<<j<<"  "<<l<<" "<<r<<"  "<<l2<<" "
<<r2<<endl;
            l2 = max(l, l2);
            r2 = min(r, r2);
            r2 = min(n, r2);
            if (l2 <= r2)
                ans += r2 - l2 + 1;
        }
        cout << ans << '\n';
    }
}
```

T5 -- 排位

解法：

- **考察知识点**：单调队列优化 DP

  这个问题描述一看就十分的复杂，考虑简化。

  - 将段位与星数抽象为积分。则每个段位对应的是一段区间。
    - 这里会有一个小问题，i 段 m 星和 $i+1$ 段 0 星在积分看来是相同的，简单讨论一下，可以得到这两者取决于最后一局的输赢。
  - 设 1 的连续段长度为 L_i，则能够达到的最大积分为

$$B+1+\min\left(\sum_i\left\lfloor\frac{L_i}{d}\right\rfloor-(n-B),0\right)$$

  - 问题转化为确定一些 `?`，求

$$\max\left\{\sum_i\left\lfloor\frac{L_i}{d}\right\rfloor\right\}$$

  - 设 $f[i,j,0/1]$ 为第 $1\sim i$ 赢了 j 局，第 i 局是 $0/1$

$$\begin{aligned}f[i,j,1]&=\max_{K\le k<i}\left\{f[k,j-(i-k),0]+\left\lfloor\frac{i-k}{d}\right\rfloor\right\}\\f[i,j,0]&=\max_{K\le k<i}\{f[k,j,1]\}\end{aligned}$$

  - 整除不破坏单调性，因此直接对 $f[i,j,0]-\frac{i}{d}$ 维护单调队列 Q_{i-j} 即可。
  - 时间复杂度为 $O(nm)$。

Code

```cpp
/* stuff you should look for
    * int overflow, array bounds, uppercase/lowercase
    * special cases (n=1?)
    * do sth. instead of nothing and stay organized
    * WRITE STUFF DOWN
    * DON'T GET STUCK ON ONE APPROACH
*/
#include<cstdio>
#include<cstring>
#include<set>
const int m=5;
std::set<int> S[2];
struct node {
    int fi; double se;
}q1[5005][5005],q2[5005][5005];
int hd1[5005],hd2[5005],tl1[5005],tl2[5005];
int ss[5005];
int f[5005][5005][2];
char s[5005];
```

```
inline int read() {
    register int x=0,f=1;register char s=getchar();
    while(s>'9'||s<'0') {if(s=='-') f=-1;s=getchar();}
    while(s>='0'&&s<='9') {x=x*10+s-'0';s=getchar();}
    return x*f;
}
inline int min(const int &x,const int &y) {return x<y? x:y;}
inline int max(const int &x,const int &y) {return x>y? x:y;}
int main() {
//  freopen("star2.in","r",stdin);
//  freopen("star10.out","w",stdout);
//  int m=read(),d=read(),B=read(),C=read();
    int d=read(),B=read();
    scanf("%s",s+1); int n=strlen(s+1);
    S[0].insert(0); S[1].insert(0);
    for(int i=1;i<=n;++i) {
        if(s[i]=='0') S[0].insert(i);
        else if(s[i]=='1') S[1].insert(i),++ss[i];
        ss[i]+=ss[i-1];
    }
//  memset(f,-0x3f,sizeof(f));
//  f[0][0][0]=0;
    for(int i=0;i<=n;++i) hd1[i]=hd2[i]=1,tl1[i]=tl2[i]=0;
    q1[0][++tl1[0]]=(node){0,0}; q2[0][++tl2[0]]=(node){0,0};
//  for(int i=1;i<=n;++i)
    for(int i=1;i<=n;++i) {
        int lst0=*(--S[0].lower_bound(i)),lst1=*(--S[1].lower_bound(i));
        for(int j=0;j<ss[i];++j) f[i][j][0]=f[i][j][1]=-0x3f3f3f3f;
        for(int j=ss[i];j<=min(i,B);++j) {
            if(s[i]!='0') {
                int lim0=max(lst0,i-j);
                while(hd1[i-j]<=tl1[i-j]&&q1[i-j][hd1[i-j]].fi<lim0)
++hd1[i-j];
                int k=q1[i-j][hd1[i-j]].fi;
                if(hd1[i-j]<=tl1[i-j]) {
                    f[i][j][1]=f[k][j-i+k][0]+(i-k)/d;
                }
                else {
                    f[i][j][1]=-0x3f3f3f3f;
                }
//              printf("[%d,%d,0] %d %d %d %d\n",k,j-i+k,lim0,hd1[i-
j],tl1[i-j],f[k][j-i+k][0]);
//              for(int k=i-1;k>=lst0&&j-i+k>=0;--k) {//k>=i-j
//                  f[i][j][1]=max(f[i][j][1],f[k][j-i+k][0]+(i-k)/d);
//                  printf("[%d,%d] %d\n",k,j-i+k,f[k][j-i+k][0]);
//              }
            }
            if(s[i]!='1') {
                int lim1=lst1;
                while(hd2[j]<=tl2[j]&&q2[j][hd2[j]].fi<lim1) ++hd2[j];
                int k=q2[j][hd2[j]].fi;
                if(hd2[j]<=tl2[j]) {
                    f[i][j][0]=f[k][j][1];
                }
                else {
```

```
                    f[i][j][0]=-0x3f3f3f3f;
                }
//              for(int k=i-1;k>=lst1&&j-i+k>=0;--k) {//k>=i-j
//                  f[i][j][0]=max(f[i][j][0],f[k][j][1]);
//              }
//              printf("[%d,%d,1] %d %d %d\n",k,j,lim1,hd2[j],tl2[j],f[k][j][1]);
            }
            if(s[i]=='0') f[i][j][1]=-0x3f3f3f3f;
            else if(s[i]=='1') f[i][j][0]=-0x3f3f3f3f;
//          printf("(%d %d)%d %d\n",i,j,f[i][j][0],f[i][j][1]);
        }
        for(int j=ss[i];j<=min(i,B);++j) {
            if(s[i]!='0') {
                while(hd2[j]<=tl2[j]&&q2[j][tl2[j]].se<=f[i][j][1]) --tl2[j];
                q2[j][++tl2[j]]=(node){i,1.00*f[i][j][1]};
            }
            if(s[i]!='1') {
                while(hd1[i-j]<=tl1[i-j]&&q1[i-j][tl1[i-j]].se<=f[i][j][0]-i*1.00/d) --tl1[i-j];
                q1[i-j][++tl1[i-j]]=(node){i,f[i][j][0]-i*1.00/d};
            }
        }
    }
    int mx=max(f[n][B][0],f[n][B][1]);
    int count=B+1+mx-(n-B);
    if(count<=0) printf("1\n");
    else {
        if(count%m==0) {
    //      printf("%d\n",(mx-1)/m+1);
            if(count==B+1+(f[n][B][0]-(n-B))) printf("%d\n",count/m+1);
            else if(count==B+1+(f[n][B][1]-(n-B))) printf("%d\n",count/m);
        }
        else {
            printf("%d\n",(count-1)/m+1);
        }
    }
//  printf("%d\n",mx);
//  if(mx<(C-1)*m) {
//      printf("NO\n");
//  }
//  else if(mx==(C-1)*m) {
//      if(f[n][B][0]==mx) printf("YES\n");
//      else if(f[n][B][1]==mx) printf("NO\n");
//  }
//  else {
//      printf("YES\n");
//  }
    return 0;
}
```

很荣幸，能为您提供便捷的解题思路，如哪里信息有bug，欢迎指正！

赛后补题很关键，及时优化代码，为自己知识宝库多积累一份珍贵资源

梦熊信竞平台，预祝您可以在2024年赛事中取得最佳战绩！