NY共克归艰（6）参考题解

A-简单的题

首先，如果反复选择a执行该操作，不管中途有没有选择b，都可以把执行过的操作里的x加起来合并成一次操作。 所以如果要使a和b都变成零，最多只需要执行两次操作。 那么a和b可以表示为 $a = x + 2y, b = 2x + y$，令此方程组有解。

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

using namespace std;
int main() {
  int T, a, b, c;
  scanf("%d", &T);
  while (T--) {
    scanf("%d%d", &a, &b);
    if ((a + b) % 3) {
      puts("no");
      continue;
    }
    c = (a + b) / 3;
    if (a >= c && b >= c)
      puts("yes");
    else
      puts("no");
  }
  return 0;
}
```

B-数列游戏

利用贪心思想，每次处理最大的偶数，直接计次数即可。 等价于对于每个偶数不断除二，统计一共出现了多少个不同的偶数。

```
#include <bits/stdc++.h>
#define MAXN 7000000
using namespace std;
int a[MAXN];
int main() {
  int n, i, x, N, T;
  scanf("%d", &T);
  while (T--) {
```

```
    N = 0;
    scanf("%d", &n);
    for (i = 1; i <= n; i++) {
      scanf("%d", &x);
      while (x % 2 == 0) {
        a[++N] = x;
        x >>= 1;
      }
    }
    sort(a + 1, a + 1 + N);
    int ans = 0;
    for (i = 1; i <= N; i++)
      if (a[i] != a[i - 1])
        ans++;
    printf("%d\n", ans);
  }
  return 0;
}
```

C-安全系数

本题为一道组合数学的题目

n个房间 m种数字，总状态数即m^n,若不相同，即相邻房间的两个人数字不同，所以第一个人有m种选择，则后面的每一个人都有$m-1$种选择，所以不冲突的状态总数为$m*(m-1)^{n-1}$;所以冲突的状态数就为$m^n - m*(m-1)^{n-1}$。

解决了这个问题后，快速幂解决就好了。

```
#include <bits/stdc++.h>
#define mod 100003
#define ll long long
using namespace std;
ll m, n;
ll qpow(ll a, ll b) {
    ll c = 1, d = a % mod;
    while (b > 0) {
        if (b & 1)
            c = (c % mod * d % mod) % mod;
        b >>= 1;
        d = (d % mod * d % mod) % mod;
    }
    return c;
}
int main() {
    scanf("%lld%lld", &m, &n);
    long long ans = qpow(m, n);
```

```
    ans = ans - m * qpow(m - 1, n - 1) % mod;
    if (ans < 0)
        ans += mod;
    printf("%lld", ans);
    return 0;
}
```

D-整数划分

解法一:

根据题意，我们直接考虑用dfs去解决，对于搜索，我们每一次需要存入当前算了多少个数以及上一个数是什么，以及和为多少，然后我们按照从小到大去递归每一种情况即可。

解法二：

动态规划，划分数问题。

$dp[i][j]$表示把i分为j个数的方案数。

考虑j个数中是否存在1, 第一种情况从i中取出j, 均分到j份, 保证每份都不为零, 即$dp[i-j][j]$

第二种情况，单独取一个1作为一个数$dp[i-1][j-1]$

所以$dp[i][j]=dp[i-j][j]+dp[i-1][j-1]$

各位可以阅读学习一下各种划分数的情况: 划分数总结

```
#include <bits/stdc++.h>
using namespace std;
int n, k, ans;
void dfs(int sum, int tim, int lat)  {//剩下的，切的次数，上一次选的数
    if (tim > k) {
        ans++;
        return;
    }
    if (sum < lat)
        return;
    if (tim == k) {
        ans++;
        return;
    }
    for (int i = lat; i <= sum; i++) dfs(sum - i, tim + 1, i);
}
int main() {
    cin >> n >> k;
    dfs(n, 1, 1);
    cout << ans << endl;
    return 0;
}
```

```
#include <bits/stdc++.h>
using namespace std;
int dp[201][201] = { 1 };
int main() {
    for (int i = 1; i < 201; i++) {
        for (int j = 1; j <= i; j++) {
            dp[i][j] = dp[i - j][j] + dp[i - 1][j - 1];
        }
    }
    int n, m;
    scanf("%d%d", &n, &m);
    printf("%d\n", dp[n][m]);
    return 0;
}
```

E-序列问题

根据题意，给定一个非负序列，求长度大于F的连续子序列的平均数最大 解法:在实数上二分平均数mid，判断a中是否有长度大于F平均数大于等于mid，再进行调整二分区间 设定一个b数组，$b[i] = a[i] - mid$ 当$b[i]$的区间和大于等于0的时候说明区间平均数大于等于mid 用sum数组表示b数组前缀和 再求出长度大于等于F的所有区间中的最大区间和=前缀和-前面的最小前缀和（要保证区间长度大于F） 判读和是否大于等于0

```
#include <bits/stdc++.h>
//#define int long long
using namespace std;

const int N = 100005;
long long n, L;
long long a[N], sum[N], mn[N];
bool check(int mid) {
    for (int i = 1; i <= n; i++) sum[i] = sum[i - 1] + a[i] - mid;
    for (int i = 1; i <= n; i++) mn[i] = min(mn[i - 1], sum[i]);
    for (int i = L; i <= n; i++)
        if (sum[i] - mn[i - L] >= 0)
            return 1;
    return 0;
}
int main() {
    cin >> n >> L;
    long long avg = 0;
    for (int i = 1; i <= n; i++) scanf("%lld", &a[i]), a[i] *= 1000, avg =
max(avg, a[i]);
    int l = 0, r = avg, ans;
    while (l <= r) {
```

```
        int mid = l + r >> 1;
        if (check(mid))
            l = mid + 1, ans = mid;
        else
            r = mid - 1;
    }
    cout << ans << endl;
    return 0;
}
```

F-不下降与不上升

首先明确一点，只要$a_m \leq b_m$即可满足条件。对于不下降数列a，如果$a_i = x$，那么a_{i-1}就有x个正整数可以选择。那么数列a中第i位填x的方案数就等于$\sum$\{数列a中第$i-1$位填y的方案数\}$(1 \leq y \leq x)$。设$a[i][j]$为数列a第i位填数字j的方案数，则有递推关系式 $a[i][j] = \sum_{k=1}^{j} a[i-1][k]$。$b$数组同理。那么先预处理出$a[i][j]$和$b[i][j]$，$ans = \sum_{j=1}^{n} \sum_{k=j}^{n} a[m][j] * b[m][k]$。

```
#include <bits/stdc++.h>
#define mod 1000000007
using namespace std;
int a[11][1005], b[11][1005];
int main() {
    int n, m;
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++) a[1][i] = b[1][i] = 1;
    for (int i = 2; i <= m; i++)
        for (int j = 1; j <= n; j++) {
            for (int k = 1; k <= j; k++) a[i][j] = (a[i][j] + a[i - 1][k]) %
mod;
            for (int k = n; k >= j; k--) b[i][j] = (b[i][j] + b[i - 1][k]) %
mod;
        }
    long long ans = 0;
    for (int i = 1; i <= n; i++)
        for (int j = i; j <= n; j++) ans = (ans + 1LL * a[m][i] * b[m][j] %
mod) % mod;
    printf("%lld", ans);
    return 0;
}
```