Problem 1

样例解释：两种可能性，1 2 3 4，3 4 1 2

- 1 2，3 4
- 这两个东西要分别作为连续子段出现在最后的队伍排列里面

根据排在前面的人是谁的记忆，得到一系列的子段

把子段拿去任意排列

- 编号为i的在哪个子段里
- 其它的子段可以排在这个子段前面（影响i的位次）或者后面（不影响i的位次），选择互相独立
- i的位次就是，在前面的子段的长度之和+i在所在的子段里面的位置
- 01背包

Problem 2

Floyd做一遍，把真正的最短路求出来

基础设计模式：把所有影响未来的东西记下来：

- 在这里，就是把三个员工的位置全部记下来
- $f[i][x_1][x_2][x_3]$，表示处理完第 i 个请求，三个员工分别在这三个位置的最小花费
- $f[i][x_1][x_2][x_3] \longrightarrow f[i+1][l_{i+1}][x_2][x_3], f[i][x_1][l_{i+1}][x_3], f[i][x_1][x_2][l_{i+1}]$
- 任意状态转移到处理完第 i 个请求，都需要至少有一个点在 l_i，这一维可以压缩掉
- $f[i][x][y]$，表示处理完第 i 个请求，其它两个员工分别在 x, y 的最小花费
- $f[i][x][y] \longrightarrow f[i+1][x][y], f[i+1][l_i][y], f[i+1][x][l_i]$，推过去的时候，要把最短路径的长度加上去

Problem 3.1

$S_1(n,k)$ 表示 n 个数排列成 k 个可旋转的环的方案数

$S_1(n,1)$，就是环排列，数量就是 $(n-1)!$，$S_1(n,n)=1$

- 当前的第 n 个数，$n-1$ 个数排成环的方案上，可以新建一个环，或者插入到环上的一个元素后面
- $S_1(n,k)=S_1(n-1,k-1)+(n-1)\times S_1(n-1,k)$

$S_2(n,k)$ 表示 n 个数划分成 k 个非空集合的方案数

$S_2(n,n)=1$，$S_2(n,1)=1$，$S_2(n,2)=2^{n-1}-1$

- 考虑1所在的集合，剩下的 $n-1$ 的数可以选择跟1相同集合或者不同集合，但是因为有至少两个非空集，所以至少要有一个数选择跟1不同集合，排除掉一种方案
- $S_2(n,k)$，对于第 n 个数来说，可以单独成为一个非空集，或者加入一个已有的非空集（k种方案）
- $S_2(n,k)=S_2(n-1,k-1)+k\times S_2(n-1,k)$

$R(n,k)$ 表示长度为 n 的排列有 k 个逆序对的个数

- $R(n,0)=R(n,n(n-1)/2)=1$，$R(n,1)=n-1$
- 假如说有一个 1 到 $n-1$ 的排列，可以把 n 插入到其中的一个位置得到一个新的排列，而且无论原来的 $1\sim n-1$ 的排列是怎么样的，增加的逆序对的个数只跟 n 插入的位置有关
- $R(n,k)=R(n-1,k)+R(n-1,k-1)+\cdots+R(n-1,k-n+1)$，分别对应 n 插在最后，倒数第二位……，第一位之后，剩下的 $n-1$ 个数在剩余的位置进行排列的方案数
- 状态数 $O(n^3)$，递推的方式是取一段连续的区间和，可以使用前缀和优化做到 $O(1)$

$E(n,k)$，表示长度为 n 的排列里有 k 个相邻递增的排列的个数

- 插入 n 的时候，看是否增加相邻递增对
- 增加的话，那么就是从 $E(n-1,k-1)$ 推来，任何一个符合 $E(n-1,k-1)$ 性质的排列，n 都不能插在相邻递增的中间，以及排列的开头，剩余可用的插的位置就是 $(n-(k-1)+1)=(n-k)$
- 不增加的话，就是从 $E(n-1,k)$ 推来，任何一个符合性质的排列，n 必须插在相邻递增的中间或者开头，可插的位置就是 $k+1$
- $E(n,k)=(n-k)E(n-1,k-1)+(k+1)E(n-1,k)$

Problem 3.2

Problem 3.1里面有关于位置下标的约束只有相对大小，这里给的是绝对的位置下标

- 所以使用插入的方式进行递推不太方便

排列的题目，使用一种能够构造出排列的方式进行DP

- n插到n-1的排列里（排除）
- ```
  // 等概率随机生成1-n排列的方式
  for(int i=1;i<=n;++i)
      p[i]=i;
  for(int i=1;i<=n;++i)
      swap(p[i],p[rand()%i+1]);
  ```
- 从n-1个数的排列构造n个数的排列，确定一下第n个数相对于前面n-1个数的相对大小
  - 1 4 3 2：1 -> 1 2 (-> 1 2 1.5) -> 1 3 2 (-> 1 3 2 1.5) -> 1 4 3 2

$f[i][k]$，表示说填好了位置下标 $1\sim i$，并且第 i 个数在前面 i 个数中从小到大排第 k

对于第 i 个数来说，如果排名确定为 k，并且……

- $p[i-1]<p[i]$，那么前 $i-1$ 个数中，只有排名小于 k 的数才会比这个前 i 个数的第 k 名小，这时候就有 $f[i][k]=f[i-1][1]+f[i-1][2]+\cdots+f[i-1][k-1]$
- 1 2 3 4 5 .... i-1 -> 1 2 2.5 3 4 5 ... i-1 -> 1 2 3 4 5 ... i
- $p[i-1]>p[i]$，那么前 $i-1$ 个数中，只有排名大于等于 k 的数才会比这个前 i 个数的第 k 名大，这时候就有 $f[i][k]=f[i-1][k]+f[i-1][k+1]+\cdots+f[i-1][i-1]$
  ```

Problem 5.1

```
1 1 1
0 1 0
9，(1,2)种草1种，(1,2)不种草8种
```

假如上面若干行都填了，那么对于下面的，没有决定种不种的格子，只会关心已经填过的最后一行

$f[i][S]$，表示填完了前 i 行，最后一行种草的状态是 S 的时候的方案数

- 最后一行 m 个格子，每个格子种/不种，第 $i-1$ 位记录 i 位有没有种，那么可以表示成二进制数
- S 就是用一个二进制数表示
- 对于当前的 $f[i][S]$，首先要判断 S 内部是否合法（只能在能种草的格子上种，并且没有相邻的两列同时种），从若干个 $f[i-1][S']$ 求和，S' 和 S 不能冲突
- $f[i][S]=\sum_{S'} f[i-1][S']$

分析一下复杂度

- 没有相邻格子为1的 S 的数量：2 3 5 8 13 21 34 55 89 144 233 377.....（这个是一个简单的递推）
- 也就是，不管怎么样，符合内部合法性的 S 最多就只有377个
- 预处理一下合法状态，然后只考虑合法状态，最暴力的转移复杂度就是 $n\times F_n^2$

处理其它的约束（位运算的技巧）

- 是否种在了不能种草的格子上：位运算压一下原始种草矩阵每行的信息，and一下的事
- S 与 S' 是否冲突：$S\&S'$ 是否不为0

Problem 5.2

跟5.1类似，$f[i][k][S]$，多记录一下个数

判断冲突的时候：$((S)|(S<<1)|(S>>1))\&S'$ 的关系

使用的国王个数就是位里面1的个数，$bitcount[S]$，数据范围比较小，可以预处理，或者使用之前的位运算技巧

Extra：可以考虑逐格转移

- 跟插头DP相关的内容，丢在提高组讲完之后讲

Problem 6.1

对于一个整数的数码进行讨论，逐位进行考虑，按位构造整数

介绍一个技巧：区间 [l,r] 内有多少个数XX -> [1,r]的数量减去[1,l-1]的数量

- 注意有的sxbk的出题人可能令 $l=0$
- 两个不规则的边界只会剩下一个，只需要考虑一个不太规整的上界
- [1,r] 内有多少个数，相邻数码差值均>=2

简单的情况：恰好 k 位数，没有上界，要求相邻数码差值>=2

- $f[i][d]$，从高位到低位填好了 i 个数码，当前填的最低位的数码是 d 的时候的数量
- 初始状态 $f[1][1\sim 9]=1$
- 转移的话：$f[i][d]=\sum f[i-1][0\sim 9]-f[i-1][d-1]-f[i-1][d]-f[i-1][d+1]$
- 最终要的结果，就是$\sum f[k][0\sim 9]$
- 记一下，这个简单的情况，$g[k]=\sum f[k][0\sim 9]$
- 对于一个中间的 $1\le i<k$，实际上也是恰好 i 位数的结果，总之可以把这个 g 求出来

假如有一个上界 r，假设 r 是一个 k 位数，那么不超过 r 的结果分为两个部分

- 不超过 $k-1$ 位数，这部分的整数 <r，可以应用简单的情况，$g[1]+g[2]+\cdots+g[k-1]$
- 恰好 k 位数，上界为 r，要求相邻数码差值>=2
- $h[i][d][0/1]$，从高位到低位填好了 i 个数码，当前填的最低位的数码是 d，并且当前的前 i 位与 r 的前 i 位是否相等
  - r = 123456789
  - now= 123……，这个时候 now 的前三位与 r 相同，所以now的第四位就不能超过r的第四位
  - now= 100……，这个时候 now 比前三位就已经小于r，所以now后面填写位的时候就没有上界r的约束（因为已经被满足）
- $h[i][d][0]$，也就是不相等，下一位可以 0-9 随便填，转移到 $h[i+1][0\sim 9][0]$
- $h[i][d][1]$，也就是相等，下一位只能填0~r[i+1]，转移到 $h[i+1][0\sim r[i]-1][0]$ 以及 $h[i+1][r[i]][1]$
- 结合这里的相邻数码差值>=2的约束，转移的时候要排除掉d-1,d,d+1

求所有整数数位和的和

- [1,r]-[1,l-1]
- 使用DP求出[1,r],[1,l-1]的数位和，形式上一样，所以举[1,r]为例子
- 在这里，有前导0并不影响数位和——想一想为什么在前面的例子中要排除掉前导0的情况（前面的例子中前导0会如何影响结果）
- 因此假如r是k位，可以把<k位的数，补上前导0到k位，一起考虑
- $h[i][d][0/1]$，从高位到低位填好了 i 个数码，当前填的最低位的数码是 d，当前所有填法的数位和
- 初始条件：$h[1][d=0,1,2,3,\cdots,r[1]-1][0]=d$，以及 $h[1][r[1]][1]=r[1]$
- $h[i][d][0]$，也就是不相等，下一位可以 0-9 随便填，转移到 $h[i+1][0\sim 9][0]$，$h[i+1][d'][0]+=h[i][d][0]+d'\times$个数，问题就在于这个个数——当第 i 位为 d 时，前面的位有多少种填法，使得前 i 位<r的前i位 —— 这个东西就是 r 的前 i-1 位形成的十进制数，根据 d 与 r[i] 的大小关系可能+1

Problem 6.2

$k=7$ 的时候，37符合要求，3给删掉，7是7的倍数

高位的数可以随便删，低位相对比较稳定，如果存在低 i 位已经满足条件，那么高位就可以随便填

$f[i][j][0/1]$，表示从低到高填写 i 位，当前的填写出来的部分模 k 为 j，是否已经有某些低位满足是 k 的倍数的条件

$f[i][j][0/1]->f[i+1][(j+d\times 10^i)\%k][0/1],[d=0,1,2,3,4,5,6,7,8,9]$

第三维的结果根据之前的第三维以及转移后的第二维的值来判断

统计答案的时候：因为还要判断一下有没前导0，所以说可以在状态上再加一维表示在当前的最高位上填写的值是不是0，统计符合条件的 $f[n][\dots][1]$ 部分