动态规划

清华大学 张华清

Luogu 1156 垃圾陷阱

卡门——农夫约翰极其珍视的一条 `Holsteins` 奶牛——已经落了到“垃圾井”中。“垃圾井”是农夫们扔垃圾的地方，它的深度为$D(2 \le D \le 100)$英尺。

卡门想把垃圾堆起来，等到堆得与井同样高时，她就能逃出井外了。另外，卡门可以通过吃一些垃圾来维持自己的生命。

每个垃圾都可以用来吃或堆放，并且堆放垃圾不用花费卡门的时间。

假设卡门预先知道了每个垃圾扔下的时间$t(0 < t \le 1000)$，以及每个垃圾堆放的高度$h(1 \le h \le 25)$和吃进该垃圾能维持生命的时间$f(1 \le f \le 30)$，要求出卡门最早能逃出井外的时间，假设卡门当前体内有足够持续10小时的能量，如果卡门10小时内没有进食，卡门就将饿死。

输入格式

第一行为2个整数，D和 $G(1 \le G \le 100)$，G为被投入井的垃圾的数量。

第二到第$G+1$行每行包括3个整数：$T(0 < T <= 1000)$，表示垃圾被投进井中的时间；$F(1 \le F \le 30)$，表示该垃圾能维持卡门生命的时间；和 $H(1 \le H \le 25)$，该垃圾能垫高的高度。

输出格式

如果卡门可以爬出陷阱，输出一个整表示最早什么时候可以爬出；否则输出卡门最长可以存活多长时间。

Luogu 1156 垃圾陷阱

- 可以视为某种背包问题。
- 普通背包问题，有两个量：空间和收益。
- 一个物品有两种选择：
- 不要某个物品有好处：不费空间。有坏处：没有收益
- 要某个物品有好处：有收益。有坏处：费空间。

Luogu 1156 垃圾陷阱

- 这个问题，有两个量：高度和生命值。
- 一个垃圾也有两种选择：
- 把一个垃圾吃了，有好处：加生命；有坏处：不能加高度
- 把一个垃圾用来垫高，有好处：加高度；有坏处：不能加生命

Luogu 1156 垃圾陷阱

- 这种问题，就是把一个量放在DP状态里，另外一个量放在DP值里，表示某个量为j时，另一个量最好是多少
- 比如背包，我们可以设f[i][j]为前i个物品，剩余容量为j的最大收益；
- 其实也可以设f[i][j]为前i个物品，收益为j时的最小容量。

Luogu 1156 垃圾陷阱

- 这个题，可以设f[i][j]为考虑了前i个垃圾（当然先按时间排序），高度为j时最大生命值是多少
- 也可以设f[i][j]为前i个垃圾（当然先按时间排序），生命值为j时最大高度是多少
- “生命值不能为0”的限制只需要在转移时稍加限制即可。

Luogu 1156 垃圾陷阱

- 这个题，是谁做下标谁当DP值都可以。但有一些题是需要好好考虑的。
- 往往是范围小的那个当下标，范围大的当DP值。

UVA12105 越大越好 Bigger is Better

> 用不超过 n 根火柴摆出一个尽量大的、能被 m 整除的数。
>
> $1 \le n \le 100, 1 \le m \le 3000$ 。

UVA12105 越大越好 Bigger is Better

- 自然的想法是，设f[i][j]为用了i根火柴，能够摆出的，模m为j的最大数字是多少。转移就枚举下一位填什么。
- 问题在于，这个“最大数字”（即DP值）太大了，需要高精度。
- 交换下标和DP值也不行。

UVA12105 越大越好 Bigger is Better

- 最大化数字的一个套路：按位贪心：
- 先让数字位数尽量大；
- 然后从高位到低位依次考虑每一位，最大化这一位的数字。
- 打个比方，我这一位填8也可以，填7也可以，那我不用看更低的那些位数，就可以直接确定这一位填8一定比填7优秀。（这就叫贪心嘛..）

UVA12105 越大越好 Bigger is Better

- 那么我们怎么知道位数最高是几，以及这一位可不可以填9，可不可以填8，可不可以填7…呢?
- 先考虑第一个问题。原来，我们要知道数字具体是几（要么放在DP状态里要么放在DP值里）。但现在，我们只需要知道这个数字的位数。状态量一下子就下来了。那么就好做了。
- 我们预处理一个数组f[i][j]，表示一共填了i位，模m为j所需要的最小火柴数。这是容易做的。
- 然后先考虑最高能填几位。我们就看看f[i][0]是否<=n就好。

$f[i][j] \longrightarrow f[i+1][(10j+k) \bmod m]$

$+c(k)$

UVA12105 越大越好 Bigger is Better

- 然后从高位到低位依次考虑。
- 对于第i位，从9到0依次尝试能不能填。
- 注意，此时更高位(i+1,i+2...)都已经确定了。那我们已经确定了仅考虑更高位时，模m的值。当第i位也确定了之后，我们就可以算出，1...i-1位模m需要是多少，设为qwe.那我们只需要判断，填i-1位，模m为qwe，最小需要的火柴棒数是否小于等于我当前有的火柴棒数。

UVA12105 越大越好 Bigger is Better

- 具体来说：假设填到第i位时，需要1…i位的数字模m为tmpm；假设此时还剩下res根火柴
- 当你试试能不能填数字h时，你就看看

```
if(f[i-1][(tmpm-h*hh[i]%m+m)%m]<=res-c[h]){
```

- 这个条件成不成立就行了。

Luogu 5363 [SDOI2019] 移动金币

Alice和Bob将要进行如下的一场游戏。二人轮流操作，且Alice先行。 当轮到一个玩家的时候，他可以选择一枚金币，并将其向左移动任意多格，且至少移动一格。 金币不能被移出棋盘，也不能越过其它金币。

一个 $1 \times n$ 的棋盘上最初摆放有 m 枚金币。其中每一枚金币占据了一个独立的格子，任意一个格子内最多只有一枚金币。

如果轮到一个玩家的时候他已经无法做出任何有效操作了（显然这个时候m枚金币恰好落在最左侧的m个格子中），则被判定为输家。已经知道Alice和Bob都是极致聪明的人，他们在任何局面下总能做出最优的操作。那么有多少初始状态能保证Alice必胜呢?

(50分) $1 \le n \le 150000$且$1 \le m \le 50$。

$n-m$

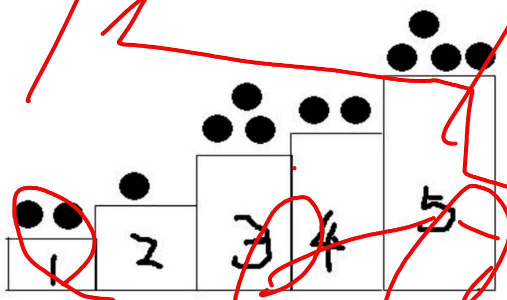
Luogu 5363 [SDOI2019] 移动金币

- 这种题一般都是先拆盒子，即先得到满足什么条件时合法，再计数。
- 移动金币？相当于移动格子。

Luogu 5363 [SDOI2019] 移动金币

- 于是相当于阶梯博弈。

1.**阶梯博弈**：阶梯的序号如图所示，地面表示第0号阶梯。每次都可以将一个阶梯上的石子向其左侧移动任意个石子，没有可以移动的空间时（及所有石子都位于地面时）输。



- 结论：奇数级阶梯的石子数目异或起来不为0，则先手必胜，否则先手必败
- 简要说明（并非今天讲课重点）：当对手动偶数级的石子时，我总能把它移动回偶数级；奇数就没有这个性质——对手把石子从1级移动到0级，我就动不了了
- 下文说的石子即原题中的空格

Luogu 5363 [SDOI2019] 移动金币

- 然后进入DP的部分，开始计数：
- “不为0”不好做，转化成用总方案数减去为0的方案数。（其实也可以做，见下面[THUPC2021]游戏）
- 为啥“为0”好做呢？因为“为0”是“二进制中每一位都要为0才行”，而“不为0是至少有一位不为0即可”

Luogu 5363 [SDOI2019] 移动金币

- DP状态怎么设计?
- 既然要求我们每一位异或为0，也即有偶数个1，那么我们自然按位考虑。（而不是一个台阶一个台阶地考虑）
- 要求石子数为给定的n-m，那么自然应该把石子数记到状态里。
- 设f[i][j]为考虑前i位（从高到低or从低到高都可以），已经放了j个石子的方案数。
- 转移时枚举有k个台阶的石子数第i位为1，其中k必须是偶数。

$$f[i][j] = f[i][j] + f[i-1][j-k*(1<<(i-1))]*c(nm,k);$$

- 其中nm是奇台阶个数

Luogu 5363 [SDOI2019] 移动金币

- 算答案就枚举几个石子是放在了奇数台阶，剩下的石头就在偶数台阶乱放就行了。
- 最后用总方案数减一下就行了。

Luogu 6196 [EER1] 代价

题目描述

给出一个长度为 $n+2$ 的序列 a_i，其中第 1 个数和第 $n+2$ 个数固定为 1。你每次可以选择序列中间的一个数删除（不能是第一个和最后一个），删除位置 p 上的数的代价为 $a_{p-1}\times a_p\times a_{p+1}$。你需要执行这个操作直到无法操作为止。求最小的代价和。

- n<=500
- n<=1e6

Luogu 6196 [EER1] 代价

f[l,r]

- n<=500?
- 区间DP
- 不能简单设f[l][r]为删去[l,r]内所有数的最小花费。因为需要知道左右是谁。
- 设f[l][r]为删去[l,r]内所有数，同时钦定此时a[l-1]和a[r+1]都没有被删掉，的最小花费。
- 转移枚举**最后一个被删掉的是谁**
- $f[l][r] = min_k(f[l][k-1] + a[l-1] * a[k] * a[r+1] + f[k+1][r])$。

Luogu 6196 [EER1] 代价

- n<=1e6?
- 贪心。
- 若序列中间没有1，从两头往中间删。花费为$\sum_k a_k * a_{k+1} + min\ a_k$。
- 若序列中有1，分成若干段分别处理即可。

Luogu 6654 [YsOI2020] 归零

题目描述

闲暇时光，Ysuerpman 选择用计算器打发时间。他输入了一个很长的十进制数 S。具体有多长呢？共 n 位。为了方便解释，设从低到高第 i 位上的数字是 S_i（下标从 1 开始）。

Ysuerpman 每次会选择一个**非零**数字位进行「四舍五入」。具体来说，假设「四舍五入」的是第 i 位：

- 如果 $S_i < 5$，则让 S 减去 $S_i \cdot 10^{i-1}$。
- 如果它 $S_i \geq 5$，则让 S 加上 10^i 再减去 $S_i \cdot 10^{i-1}$。

经过若干次操作后，S 总会变成 0。现在问题来了，请问有多少种使得 S 变成 0 的不同的方案？两个方案不同当且仅当某一次选择的操作位置不同。

对于 100% 的数据，满足 $1 \leq n \leq 64$，S 不含前导零。

Luogu 6654 [YsOI2020] 归零

- 重要观察：当某个数被四舍五入了以后，左右两部分被分成了“相对独立”的两部分。
- 举例：6549352，当4四舍五入成0：6509352，你发现9352这部分最多再变成10352，然后新出现的1永远不可能变成再往前进位了：也就是说右边再不可能对左边产生任何影响（当然左边始终不可能对右边有影响）。
- **这启示我们考虑区间DP。**

Luogu 6654 [YsOI2020] 归零

- 自然的想法是枚举断点（即第一次操作的位置），然后两边变成0的方案数乘一下？（设f[l][r]为[l,r]这一段，全变成0的方案数）
- 显然有问题，还需要考虑两边操作的顺序。
- 于是需要多开一维，记录用多少次变成0，然后两边组合数一下。（设f[l][r][c]为[l,r]这一段，用c次操作变成0的方案数）
- 还要考虑右边的进位。设f[l][r][c][0/1]为[l,r]这一段，l-1是否向l进位，用c次操作变成0的方案数。（注意l为低位）

Luogu 6654 [YsOI2020] 归零

- 转移枚举第一次四舍五入的位置是哪。
- 枚举左边用了几次lc，则右边用了rc=c-lc次。两边方案数乘上C(c-1,lc)
- 特殊情况：向r+1进位。r+1从0->1，只需要一次操作即可。
- 思考：我们为什么认为r+1原来是0?

- 复杂度O(n^5)。

Luogu 6654 [YsOI2020] 归零

- 剪枝：预处理出[l][r]这一段最少需要几次操作，最多需要几次操作。把不合法的c直接剪掉。
- 最少：从低位到高位，一个一个四舍五入。考虑你把高位四舍五入变成0了，如果低位再像高位进了个1，一定不优。
- 最多：从高到低位，一个一个四舍五入，如果进位了，再花费一次操作把进的1变成0。

工厂

【样例 1 输入】

```
4 2
1 3
1 5
4 6
2 7
```

【样例 1 输出】

```
4
```

【样例 1 解释】

最佳方案是前两个工人在 1 号流水线，后两个工人在 2 号流水线，这样每条流水线的工作时长都是 2，总时长为 4。

另一种合法方案是工人 1、2 和 4 在 1 号流水线，工人 3 在 2 号流水线，但这样总时长仅为 3。

- n,p<=200

题意：

大神 wyp 开了家工厂，工厂有 n 个工人和 p 条流水线。

工厂的工人都是睡神，因此第 i 个工人只会在 si 至 ti 时刻才会工作。

每个工人都会被分派到一条流水线上，然而，一条流水线只会在这条线的工人到齐时才能开工，其余时间即使有部分工人到了也只能休息。

根据大神 wyp 的神谕，不能有流水线的工作时间为 0，也不能有工人没被分派到流水线上（即使这样会降低实际工作时间）。

工人们 dp 不过关，所以请你求出能得到的最大的工作时间总和。

保证题目至少存在一种合法的分配方案。

工厂

- 区间类问题的一个常见的解法:
- 如果区间没有包含关系，那往往是好做的。此时左右端点都是单调的。
- 所以就想办法让所有区间互不包含。

工厂

- 如果B区间被A区间包含（即B∈A），那么我们发现，贪心地考虑，A只有两种情况：要么自己一组（此时它对最终答案做出自己长度的贡献），要么和B一组（此时它对答案不做贡献）。因为如果它和一个不属于它的区间分在一组，那一定会使最终答案变小，还不如让它和B分在一组，使最终答案不变。
- （如果有多个区间被A包含，那A随便和哪个分在一组都是一样的（反正都没贡献））
-

工厂

- 于是我们称不包含其他任何集合的集合为P型集合，其他的为Q型集合。
- 这时P型集合显然不存在互相包含关系，可以直接DP。
- 这时可以发现，按照左端点排序（常见操作），那么可以发现，划在一组里的区间一定是连续的。那么可以直接n^3Dp转移，就f[i][j]表示前i个区间，分成j组的方案数，枚举当前这一组从哪里开始即可。（所谓“组”就是流水线啦）

工厂

- 然后你枚举一下P型集合分成了i组。还剩p-i组，用前p-i长的Q型集 合补充（就像刚刚说的，Q型集合要么自己一组，贡献为自己的长度，要么和被它包含的某个集合一组，不做贡献。）

max f[m][i] + a 最大的 p-i 个

2602 [ZJOI2010]数字计数

题目描述

复制Markdown 展开

给定两个正整数 a 和 b，求在 $[a,b]$ 中的所有整数中，每个数码(digit)各出现了多少次。

- 对于 30% 的数据，保证 $a \le b \le 10^6$；
- 对于 100% 的数据，保证 $1 \le a \le b \le 10^{12}$。

2602 [ZJOI2010]数字计数

- 每个数码分别计算
- 做差分。算[1,b]的答案减去[1,a-1]的答案。

2602 [ZJOI2010]数字计数

- 数位DP的一般套路：
- 状态设计：f[i][0/1][0/1]为当第i+1位是/否贴上界时，第i+1位及更高的位是否全为0时（因为往往不考虑前导零，需要特殊处理），较低的前i位的答案。
- *注：大多动态规划状态的额外信息都是对已考虑元素的限制，而数位DP这里的0/1是对尚未考虑的部分加以限制。*
- 往往用记忆化搜索更容易实现

2602 [ZJOI2010]数字计数

```
long long dfs(int x,bool upp,bool flg0,int d){
    //考虑到第[1,x]位（1为最低位），前x+1位贴不贴上界，是不是全是0（考虑前导0嘛w），有几个数字d
    if(x==0)return 0;
    if(f[x][upp][flg0]!=-1)return f[x][upp][flg0];
    long long anss=0;
    for(int i=0;i<=(upp?nn[x]:9);i++){
        if(i==d){//如果现的这位就是要处理的数码d，那就算一下
            if((flg0==false)||d!=0){//前导0的0不能算
                anss+=((upp&&i==nn[x])?nm[x-1]+1:qwq[x-1]);//看看[0,x-1]位有多少个数，每有1个数就会多一个
                ////nm[i]是x拿出较低的1到i位是几，qwq[i]是10的i次方
            }
        }
        anss+=dfs(x-1,(upp&&i==nn[x]),flg0&&i==0,d);
    }
    return f[x][upp][flg0]=anss;
}
```

4317 花神的数论题

题目描述

话说花神这天又来讲课了。课后照例有超级难的神题啦…… 我等蒟蒻又遭殃了。 花神的题目是这样的：设 $\text{sum}(i)$ 表示 i 的二进制表示中 1 的个数。给出一个正整数 N ，花神要问你 $\prod_{i=1}^{N} \text{sum}(i)$ ，也就是 $\text{sum}(1) \sim \text{sum}(N)$ 的乘积。

对于 100% 的数据，$1 \leq N \leq 10^{15}$。

4317 花神的数论题

- 分别求出1...n中，几个数有1个1,2个1,3个1...
- 数位DP求解。

4317 花神的数论题

- 数位DP的一般套路：
- 状态设计：f[i][0/1][0/1]为当第i+1位是/否贴上界时，第i+1位及更高的位是否全为0时（因为往往不考虑前导零，需要特殊处理），较低的前i位的答案。
- *注：大多动态规划状态的额外信息都是对已考虑元素的限制，而数位DP这里的0/1是对尚未考虑的部分加以限制。*
- 往往用记忆化搜索更容易实现

4317 花神的数论题

- 对于本题，设f[i][0/1][j]为当第i+1位是/否贴上界时，较低的前i位有j个1的方案数。

```
long long dfs(int x,bool upp,int nm1){
    if(nm1<0)return 0;
    if(x==0)return nm1==0;
    if(f[x][upp][nm1]!=-1)return f[x][upp][nm1];
    long long anss=0;
    if(upp){
        if(nn[x]==1)anss=dfs(x-1,true,nm1-1)+dfs(x-1,false,nm1);
        else anss=dfs(x-1,true,nm1);
    }
    else anss=dfs(x-1,false,nm1)+dfs(x-1,false,nm1-1);
    return f[x][upp][nm1]=anss;
}
```

4124 [CQOI2016] 手机号码

f

0/1 0/1

题目描述

复制Markdown 展开

人们选择手机号码时都希望号码好记、吉利。比如号码中含有几位相邻的相同数字、不含谐音不吉利的数字等。手机运营商在发行新号码时也会考虑这些因素，从号段中选取含有某些特征的号码单独出售。为了便于前期规划，运营商希望开发一个工具来自动统计号段中满足特征的号码数量。

工具需要检测的号码特征有两个：号码中要出现至少 3 个相邻的相同数字；号码中不能同时出现 8 和 4。号码必须同时包含两个特征才满足条件。满足条件的号码例如：13000988721、23333333333、14444101000。而不满足条件的号码例如：1015400080、10010012022。

手机号码一定是 11 位数，前不含前导的 0。工具接收两个数 L 和 R，自动统计出 $[L,R]$ 区间内所有满足条件的号码数量。L 和 R 也是 11 位的手机号码。

4124 [CQOI2016] 手机号码

- 限制：
  - 至少三个相邻相同数字
  - 不能同时有4和8。

$f[i][j][t][o][N]$

4124 [CQOI2016] 手机号码

- 状态设计?
- 相应地需要多记录一些信息:
  - 第i+1位是啥
  - 第i+2位是啥
  - 是否已经有了三个连续相同数字
  - 是否有4
  - 是否有8

f[i][0/1][2][0...9][0...9]

倒第一 倒第2

[0/1][0/1][0/1]

4 8

4124 [CQOI2016] 手机号码

```
long long dfs(int x,bool upp,bool lx,int l2,int l1,bool flg4,bool flg8){
    //第x位，贴不贴上界，目前有没有连续3个相同的，第x+2位是啥，第x+1位是啥，有没有4，有没有8
    //都是贴不贴上届有没有连续3个相同的有没有4有没有8都是对于前x+1位说的，也就是还没考虑第x位
    //ps:高位为大（也即x+1是x位的高位
    if(flg4&&flg8)return 0;
    if(x==0)return lx;//必须有三个连续的(4，8已经在上一行判过了
    if(f[x][upp][lx][l2][l1][flg4][flg8]!=-1)return f[x][upp][lx][l2][l1][flg4][flg8];
    long long anss=0;
    for(int i=0;i<=(upp?nn[x]:9);i++){
        anss+=dfs(x-1,upp&&i==nn[x],lx|(l2==l1&&l1==i),l1,i,flg4|(i==4),flg8|(i==8));
    }
    return f[x][upp][lx][l2][l1][flg4][flg8]=anss;
}
```

7609 [THUPC2021] 游戏

题目描述

复制Markdown　展开

小 C 和小 W 两个人打算玩一局双人博弈游戏。

现在小 C 手里有 n 颗相同的石子，小 W 打算把它们分为有顺序的 m 堆，其中第 i 堆石子的数量不能超过 a_i，但允许为 0。

随后，由小 C 先手，两人轮流进行操作，每次可以选择一堆非空的石子并拿走其中若干个（至少 1 个），无法操作者输。

作为算法竞赛界的老司机，小 C 和小 W 早已对各种游戏的策略摸得门儿清，于是这次他们打算玩点不一样的：他们想要知道，有多少种分石子的方法能使得小 C 有必胜策略。

输入格式

第一行，两个正整数 n, m（$1 \le n \le 10^{18}$，$1 \le m \le 10$）。

第二行，m 个非负整数 a_i（$0 \le a_i \le 10^{18}$）。

7609 [THUPC2021] 游戏

- 拆博弈论盒子。题意：把n个石子分成m堆，第i堆不超过a[i]且 Xor 非 0 的方案数。
- 以下我们讨论的数位默认为二进制。

7609 [THUPC2021] 游戏

- 问题1：第i堆石子数量不超过a[i]的上界?
- 一般的数位DP只处理一个数，用一个0/1变量记录是否贴上界。
- 这里我们用一个集合（二进制数）S分别记录每堆石子的数量是否贴上界。

0/1

0/1

0/1

0/1

7609 [THUPC2021] 游戏

- 问题2：和需要为n?
- 显然不应该用一个变量记录当前已经用了多少个石子（1e18！）
- 在数位DP的过程中，记录当前位，m个数的和应该是几（即应该有几个1）。
- 该位没填够怎么办？进位！
- 假设第i位应该有j个1，但m个数的第i位只填了k个1，那么第i-1位（较低位）应该有(j-k)*2+[n的第i-1位]个1。
- 这一维的大小(能填的1的个数)不超过2*m。

7609 [THUPC2021] 游戏

- 问题3：异或和需要不为0
- 我们当然可以用类似移动金币的方法，用总方案数减去异或和为0的方案数
- 介绍另一种方法：只需要额外用一个0/1变量，记录前面的位中是否有一位异或和已经为0即可。
- *注：总方案数怎么算？可以再跑一个类似的数位DP，也可以2^m 容斥（隔板法）。*

7609 [THUPC2021] 游戏

- 问题4：如何转移?
- 如果暴力枚举m个数第i位填啥，那么一次转移的复杂度是$O(2^n)$，总复杂度为$O(4^n poly(n))$，不可接受。
- 常见优化思路：类似轮廓线DP，依次考虑每个数第i位填啥。只需要额外记录一个0/1变量表示考虑到的前k个数的异或和是多少即可。

7609 [THUPC2021] 游戏

- 将这些小问题逐个解决后，整道题目也就迎刃而解了！

```
//当前贴上界情况是S，需要后面选j位的方案数（j<=2m）
//变成记录这一位需要填j个1的方案数，当做完一行后， j*=2，表示进位;同时，遇到n的这一位是1，就再++
//当前是否有异或和是0的位：当前这位异或和为多少
    for(int w=60;w>=0;w--){
        for(int i=1;i<=m;i++){
            std::swap(nww,lst);
            memset(f[nww],0,sizeof(f[nww]));
            for(int S=0;S<=(1<<m)-1;S++){
                for(int j=0;j<=2*m;j++){
                    for(int h1=0;h1<=1;h1++){
                        for(int h2=0;h2<=1;h2++){
                            for(int nw=0;nw<=(((S>>(i-1))&1)?a[i][w]:1)&&j-nw>=0;nw++){//枚举填啥
                                int nS=S;
                                if((S&(1<<(i-1)))&&nw<a[i][w])nS^=(1<<(i-1));
                                f[nww][nS][j-nw][h1][h2^nw]=add(f[nww][nS][j-nw][h1][h2^nw],f[lst][S][j][h1][h2]);
                            }
                        }

                    }
                }
            }
        }
```

引入：TSP问题

NPC

- 旅行商问题：给定一个带权图，求最短哈密顿回路。
- 自然可以n!枚举。
- 注意：我们不关心已经走过的点的内部顺序，而只关心走过了谁。
- 于是可以用状压DP：设f[S][x]为已经走过了S里的点，当前在x的最小花费。
- 常见方法：状压优化阶乘枚举。

$n! \to 2^n n^2$

6622 [省选联考 2020 A/B 卷] 信号传递

题目描述

复制Markdown 展开

一条道路上从左至右排列着 m 个信号站，初始时从左至右依次编号为 $1, 2, \ldots, m$，相邻信号站之间相隔 1 单位长度。每个信号站只能往它右侧的任意信号站传输信号（称为普通传递），每单位长度距离需要消耗 1 单位时间。道路的最左侧有一个控制塔，它在最左侧信号站的左侧，与其相隔 1 单位长度。控制塔能与任意信号站进行双向信号传递（称为特殊传递），但每单位长度距离需要消耗 k 个单位时间。对于给定的长度为 n 的信号传递序列 S，传递规则如下：

1. 共 $n-1$ 次信号传递，第 i 次信号传递将把信号从 S_i 号信号站传递给 S_{i+1} 号。
2. 若 S_{i+1} 号信号站在 S_i 号右侧，则将使用普通传递方式，从 S_i 号直接传递给 S_{i+1} 号。
3. 若 S_{i+1} 号信号站在 S_i 号左侧，则将使用特殊传递方式，信号将从 S_i 号传递给控制塔，再由控制塔传递给 S_{i+1} 号。
4. 若 $S_i = S_{i+1}$，则信号无须传递。

阿基作为大工程师，他能够任意多次交换任意两个信号站的位置，即他能够重排信号站的顺序，这样会使得 S 消耗的传递时间改变。现在阿基想知道，在他重排信号站顺序后，S 所消耗的传递时间最小能是多少。

100% 的数据：$2 \le m \le 23$，$2 \le n \le 10^5$，$1 \le k \le 100$，$1 \le S_i \le m$。

6622 [省选联考 2020 A/B 卷] 信号传递

- 显然S是骗人的。只需要设cnt[i][j]为S中有多少次是i和j相邻（i在j前。）
- 暴力：阶乘枚举。
- 自然，我们希望能够用状压DP优化。但问题在于，我们真的不关心已经确定了的信号塔的内部顺序吗？（如，设f[S]为前|S|个信号塔为S的最小花费，但转移时需要知道S内每个塔的具体位置。）

6622 [省选联考 2020 A/B 卷] 信号传递

- 其实是需要知道的！
- 当我们确定前|S|个信号塔中第|S|个是谁后，我们希望求出它和前|S|-1个塔通信的总代价。
- 不管代价形如(p[y]-p[x]还是k(p[x]+p[y]))，我们可以知道p[y]/kp[y]这部分代价，但不知道-p[x]/kp[x]这部分代价。因为只有当你刚刚加入x时，你才知道x在第几个位置（即p[x]）。

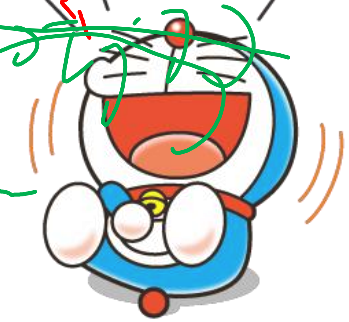
6622 [省选联考 2020 A/B 卷] 信号传递

- 套路：代价提前计算！
- 当我们确定x后，我们不仅考虑它和它前面的塔产生的贡献，也把它和它后面的塔产生的贡献中，x带来的那一部分提前考虑！
  - 因为此时p[x]是已知的
  - 哪些塔排在x之后也是已知的，那么x会贡献多少也是已知的

6622 [省选联考 2020 A/B 卷] 信号传递

- 具体写一下转移柿子：
- $f[S] = min_i\, f[S \setminus i] + cost[S][i]$，其中
- $cost[S][i] = \sum_{j\in S\setminus i} cnt[j][i] * p[i] - cnt[i][j] * k * p[i] + \sum_{j\notin S} cnt[i][j] * (-p[i]) + cnt[j][i] * k * p[i]$

$2^n \cdot n^2$



6622 [省选联考 2020 A/B 卷] 信号传递

- 如果每次暴力计算cost[S][i]，复杂度$O(2^n n^2)$，无法通过本题。
- 我们可以递推cost数组！
- 设x是S除了i的最小元素。
- 从cost[S\x][i]推到cost[S][i]。
- 相当于从把x从i后面变到了i前面，把x和i之间的贡献改一下即可：
- $cost[S][i] = cost[S\backslash x][i] - (cnt[i][x] * (-p[i]) + cnt[x][i] * k * p[i]) + (cnt[x][i] * p[i] + cnt[i][x] * k * p[i])$。

6622 [省选联考 2020 A/B 卷] 信号传递

- 过了？不幸的是空间复杂度也为$O(m * 2^m)$。只需要按照S中1的个数从小到大枚举集合，一边算cost一边DP。使用滚动数组。
- 需要2*23*C(23,11)*4B的空间，可以接受。

$\binom{23}{11}$

1000000 0

1000

120

121

6622 [省选联考 2020 A/B 卷] 信号传递

- 法二：经典结论：二进制数从0变到n，数位变化的次数为O(n)。
- 那么不要通过枚举S最低位的方式递推了。直接从S推到S+1，把所有0/1数位变化的位置分别修改即可。（需要用取lowbit的方式快速找到哪些数位变了。）
- 这样直接不需要S这一维了。

7519 [省选联考 2021 A/B 卷] 滚榜

题目描述

复制Markdown 展开

封榜是 ICPC 系列竞赛中的一个特色机制。ICPC 竞赛是实时反馈提交结果的程序设计竞赛，参赛选手与场外观众可以通过排行榜实时查看每个参赛队伍的过题数与排名。竞赛的最后一小时会进行“封榜”，即排行榜上将隐藏最后一小时内的提交的结果。赛后通过滚榜环节将最后一小时的结果（即每只队伍最后一小时的过题数）公布。

Alice 围观了一场 ICPC 竞赛的滚榜环节。本次竞赛共有 n 支队伍参赛，队伍从 $1 \sim n$ 编号，i 号队伍在封榜前通过的题数为 a_i。排行榜上队伍按照过题数从大到小进行排名，若两支队伍过题数相同，则编号小的队伍排名靠前。

滚榜时主办方以 b_i 不降的顺序依次公布了每支队伍在封榜后的过题数 b_i（最终该队伍总过题数为 $a_i + b_i$），并且每公布一支队伍的结果，排行榜上就会实时更新排名。Alice 并不记得队伍被公布的顺序，也不记得最终排行榜上的排名情况，只记得每次公布后，本次被公布结果的队伍都成为了新排行榜上的第一名，以及所有队伍在封榜后一共通过了 m 道题（即 $\sum_{i=1}^{n} b_i = m$）。

现在 Alice 想请你帮她算算，最终排行榜上队伍的排名情况可能有多少种。

【数据范围】

对于所有测试数据：$1 \le n \le 13$，$1 \le m \le 500$，$0 \le a_i \le 10^4$。

每个测试点的具体限制见下表：

测试点编号	$n \le$	$m \le$
$1 \sim 2$	2	10
$3 \sim 5$	3	10
$6 \sim 8$	8	100
$9 \sim 12$	10	200
$13 \sim 16$	12	300
$17 \sim 20$	13	500

7519 [省选联考 2021 A/B 卷] 滚榜

- 阶乘暴力？贪心判断可行性即可。
- 具体来说，$b_{p[i]} = \max(b_{p[i-1]}, (a_{p[i-1]}+b_{p[i-1]}) - a_{p[i]} + [p[i-1] < p[i]]) = b_{p[i-1]} + \max(0, a_{p[i-1]} - a_{p[i]} + [p[i-1] < p[i]])$

7519 [省选联考 2021 A/B 卷] 滚榜

- 还是希望用状压DP优化阶乘枚举。
- 注意，b_i的分配不计入方案数，我们要统计的只是队伍的合法顺序。
- 我们需要$\sum b_i = m$，为满足这个限制显然$\sum b_i$越小越好，因为最后一支队伍的b可以任意大。这和前面的贪心是一样的。

7519 [省选联考 2021 A/B 卷] 滚榜

- 那么，我们自然会这样设状态：
- 显然需要记录集合S为前|S|支队伍，当他们**最少**用i道题目时，所能形成的顺序方案数。为了转移这还不够，需要额外记录最后一个队伍是谁，以及它的bi是多少。转移枚举下一个队伍是谁，那么下一个队伍的bi可直接求出（和贪心一样）。
- 状态复杂度$2^n * m * n * m$。转移枚举最后一个队伍是谁，复杂度$O(2^n m^2 n^2)$。

7519 [省选联考 2021 A/B 卷] 滚榜

- 如何优化？也是代价提前计算的思路。
- 我们希望把最后一个数的b_i这一维省去。
- 记录它是因为题目要求b_i单调不减。而这等价于b_i的差分（即下一项-上一项）>=0。
- 我们只需要当$b_{p[i]} > b_{p[i-1]}$时，即$a_{p[i-1]} - a_{p[i]} + [p[i-1] < p[i]]$>0，把后面每一个b都提前加上这个差分值，那么我们就不需要记录b_i是多少了！

7519 [省选联考 2021 A/B 卷] 滚榜

```
for(int S=1;S<(1<<n)-1;S++){
    for(int i=1;i<=n;i++)if((S>>(i-1))&1){
        for(int c=0;c<=m;c++){
            for(int x=1;x<=n;x++)if(!((S>>(x-1))&1)){
                int w=max(0,a[i]-a[j]+(i<j))*(n-nm1[S]);
                if(c-w>=0){
                    f[S|(1<<(x-1))][x][c-w]+=f[S][i][c];
                }
            }
        }
    }
}
```

7519 [省选联考 2021 A/B 卷] 滚榜

- 从这题可以学到的两个点：
- 1. 带有某个限制的计数问题，当构成限制的元素本身不计入方案数时，考虑在DP过程中保证最优性（即最容易满足该限制）。这样的统计不重不漏。
- 2. 单调递增->差分后>=0。代价提前计算。

谢谢大家！