搜索

黎伟诺

7.18.2024

Meet in the middle

算法的主要思想是将整个搜索过程分成两半，分别搜索，最后将两半的结果合并。（折半搜索）

方程的解数

P5691

已知一个 n 元高次方程：

$$\sum_{i=1}^{n} k_i x_i^{p_i} = 0$$

其中：$x_1, x_2, \dots, x_n$ 是未知数，$k_1, k_2, \dots, k_n$ 是系数，$p_1, p_2, \dots p_n$ 是指数。且方程中的所有数均为整数。

假设未知数 $x_i \in [1, m]$ $(i \in [1, n])$，求这个方程的整数解的个数。

$1 \le n \le 6$，$1 \le m \le 150$。

方程的解数

P5691

观察这个式子，想办法把 n 的规模降为原来的一半。

$$\sum_{i=1}^{n} k_i x_i^{p_i} = 0$$

把左边拆开得：

$$\sum_{i=1}^{\lfloor n/2 \rfloor} k_i x_i^{p_i} + \sum_{\lfloor n/2 \rfloor}^{n} k_i x_i^{p_i} = 0$$

移项得：

$$\sum_{i=1}^{\lfloor n/2 \rfloor} k_i x_i^{p_i} = -\sum_{\lfloor n/2 \rfloor}^{n} k_i x_i^{p_i}$$

于是这样就可以折半搜索了。

接下来确定一下是否满足性质：

x_i 的不同取值不会影响别的答案，方程的解数量不会发生改变。

因为是要枚举 x_i 的取值，所以无论怎么搜，答案都是一样的，所以满足状态可逆。

方程的解数

P5691

考虑合并。

可以发现，等号两边的区别在于有一个负号。

于是我们可以搜一半，然后找另一半中满足与它相加等于 0 的数，也就是相反数。

之后合并答案就可以了。

可以哈希合并，也可以排序后二分

双指针的统计方法：首先将两个答案数组排序。

然后找到两个相加满足条件的第一个位置，然后统计左半部分一样的有多少个，右半部分一样的有多少个，然后运用乘法原理合并起来就好了。

Balanced Cow Subsets G

P3067

题意翻译

我们定义一个奶牛集合 S 是平衡的，当且仅当满足以下两个条件：

- S 非空。
- S 可以被**划分**成两个集合 A, B，满足 A 里的奶牛产奶量之和等于 B 里的奶牛产奶量之和。划分的含义是，$A \cup B = S$ 且 $A \cap B = \varnothing$。

现在给定大小为 n 的奶牛集合 S，询问它有多少个子集是平衡的。请注意，奶牛之间是互不相同的，但是它们的产奶量可能出现相同。

输入格式

第一行一个整数 n，表示奶牛的数目。

第 2 至 $n+1$ 行，每行一个数 a_i，表示每头奶牛的产奶量。

输出格式

输出一个数表示方案总数。

样例解释

共存在三种方案。集合 $\{1,2,3\}$ 可以划分为 $\{1,2\}$ 与 $\{3\}$；集合 $\{1,3,4\}$ 可以划分为 $\{1,3\}$ 与 $\{4\}$；集合 $\{1,2,3,4\}$ 可以划分为 $\{1,4\}$ 与 $\{2,3\}$，共 3 种子集。

数据范围及约定

对于全部数据，保证 $1 \le n \le 20$，$1 \le a_i \le 10^8$。

首先，一个有 20 头奶牛，那么考虑对于每一头奶牛来说有 3 种状态，放在一组，放在另一组，不放任何一组，如果暴力枚举时间复杂度为 $O(3^n) > 10^9$，无法接受。
考虑将 n 头奶牛分为两半，每组分别暴力求解，时间复杂度 $O(3^{\frac{n}{2}})$ 可以通过。
假设在前一半中，在第一组中放的数的和为 a，在第二组中放的数为 b。
假设在后一半中，在第一组中放的数的和为 c，在第二组中放的数为 d。
那么 $a + c = b + d$
由于我们要对每一半分开处理，所以考虑将同一半的数放在一起处理，即移项得 $a - b = c - d$。
因此，我们只需要统计在每一半中和为 $a - b$ 的方案有多少种（放进 $unordered\_map$ 里），再进行组合。
由于题目求的是可能的集合数，是需要将对应的 $mask$ 赋值为 1，而不是 $cnt++$。

小小现在需要解决一个简化的电路布线问题，在一个 $n \times m$ 的方格中进行电路布线。其中：

- 并号 `#` 标记的格子已经被占用，不能布线。
- 加号 `+` 标记的格子会连接到电路的其他部分，必须被布线。在给定的电路布线问题中，至少有一个格子必须被布线。
- 点号 `.` 标记的格子小小有权选择是否布线：布线即将该格标记为加号，不布线即保持为点号。

小小的任务是选择尽可能多的格子进行布线 (将 `.` 的格子标记为 `+`)，满足：

1. 布线电路连通。即从任意一个已布线的格子，都能通过上、下、左、右移动到相邻已布线格子的方式，到达任意另一个布线的格子。
2. 布线不存在短路 (回路)，即不存在某个布线的格子能通过 > 2 步的上、下、左、右移动到相邻布线格子的方式回到自身，且经过的格子各不相同。

例如，以下是一个电路布线问题，已有三个格子被标记为必须布线 (加号)：

```
#....#
....+#
.+####
.+...#
```

以下展示了一种合法和两种不合法的布线方案：

```
#+.+.# #.+..# #++..#
+++++# ..+++# .++++#
.+#### .+#### .+####
.++++# .+...# .+...#
合法  不连通  有回路
```

正解折半搜索（口胡谁都会，但是巨难写）
分为上半 18 个点和下半 18 个点（中间是分割线）
1、只有上半或下半，直接暴力 dfs
2、上半和下半都有点，每边 2^18 枚举并删除连成环、除开分割线有不连通的情况
合并之后如何判断是一棵树？需要有连通条件、边数等于点数 -1（两格相邻点会使边数 $+1$）
对分割线的有效状态就是：分割线的 2^6 状态，对应的并查集 $6*5*4*3*2*1=120$、两边的点数减边数（范围是 $[1,3]$，因为分割线最多再产生三条边）
提前枚举两对这样的状态预处理是否合法
然后每个下半的状态去搜上半的 *unordered_map* 统计答案。

乱搞但是能过的爆搜剪枝
暴力做法是一位一位按照坐标顺序，枚举每一个格子即可。
判断图是否连通可以使用并查集或者深度优先搜索判断。关键问题在于如何判断图是否存在回路。
根据回路的性质，在一个环中，可以证明，其点数等于其边数。所以，我们可以算出原图中所有的点数，即一共放了多少个位置，再算出原图中的边数，即一共有多少个点对 $(x_1, y_1), (x_2, y_2)$ 满足他们上面都布了电路而且这两个格子相邻。
这样子能拿 40 分。

乱搞但是能过的爆搜剪枝

接下来考虑如何进行剪枝。这里不难想到两种剪枝方法:

可行性剪枝: 如果原图中已经存在了回路，那么整个方案一定都是不合法的。

最优性剪枝: 哪怕接下来的所有格子都布满电线，电线的数量都没有目前最优方案的高，那么这个方案也可以舍掉。

加上这两个剪枝，足以让我们通过这道题目。

买瓜

P9234

小蓝正在一个瓜摊上买瓜。瓜摊上共有 n 个瓜，每个瓜的重量为 A_i。小蓝刀功了得，他可以把任何瓜劈成完全等重的两份，不过每个瓜只能劈一刀。

小蓝希望买到的瓜的重量的和恰好为 m。

请问小蓝至少要劈多少个瓜才能买到重量恰好为 m 的瓜。如果无论怎样小蓝都无法得到总重恰好为 m 的瓜，请输出 -1。

$1 \leq n \leq 30$。

买瓜

P9234

1. 对于每一个瓜，有三种状态：不买这个瓜，砍一刀，买一半瓜，不砍，买整个瓜。
2. 为了防止浮点精度爆炸，可以把 $m \times 2$ ，买一半瓜时就直接加上 a_i ，买整个瓜就加上 $a_i \times 2$ 。
3. 用数组存储买到重量为 *sum* 的瓜要砍几刀可能会炸，可以用 *map* 存储，而 *unordered_map* 比 *map* 更快。
4. 我们很容易就会想到搜索，但朴素的搜索复杂度为 $O(3^n)$ ，而 $n \leq 30$ ，一定过不了。所以我们就会用到折半搜索。
5. 折半搜索的复杂度为 $O(3^{\frac{n}{2}})$ ，也过不了，所以需要一些优化：
排序，优化搜索顺序。
unordered_map 代替 *map* 。
剪枝，目前重量大于 m 就不搜了，砍瓜次数大于目前最优解就不搜了。

离散对数

给定 $a, b, m \leq 10^9$，如何求解 $a^x \equiv b \pmod{m}$。
保证 (a, m) 互质。

离散对数

令 $x = A\left\lceil\sqrt{m}\right\rceil - B$，其中 $0 \le A, B \le \left\lceil\sqrt{m}\right\rceil$，则有 $a^{A\left\lceil\sqrt{m}\right\rceil - B} \equiv b \pmod{m}$，稍加变换，则有 $a^{A\left\lceil\sqrt{m}\right\rceil} \equiv ba^{B} \pmod{m}$.
我们已知的是 a, b，所以我们可以先算出等式右边的 ba^B 的所有取值，枚举 B，用 hash/map 存下来，然后逐一计算 $a^{A\left\lceil\sqrt{m}\right\rceil}$，枚举 A，寻找是否有与之相等的 ba^B，从而我们可以得到所有的 x，$x = A\left\lceil\sqrt{m}\right\rceil - B$.
注意到 A, B 均小于 $\left\lceil\sqrt{m}\right\rceil$，所以时间复杂度为 $\Theta\left(\sqrt{m}\right)$，用 map 则多一个 $\log$.

Xor-Paths

CF1006F

给出一个 $n \times m$ 的网格，每个格子上有权值 $a[i][j]$，现在 Alice 要从 $(1,1)$ 走到 (n,m)，每次只能向右或向下走，沿路计算异或和，求异或和等于 k 的路径数。

$1 \leq n, m \leq 20, 0 \leq k \leq 10^{18}$

考虑双向搜索。
以对角线为界限，第一个 dfs 记录从左上角到对角线的方案异或和存在一个 map 里面
第二个 dfs 从右下角出发，每当走到对角线时就在 map 里找 $k \oplus xorsum_{rightdown}$ 累计答案，这时时间复杂度就能优化到 2^{20}

填符号

一个正整数数列 $a_1, a_2, \ldots, a_n$，要在中间每一个空位（共 $n-1$ 个）填上加号或者乘号。
问算式结果最终 $\equiv K \pmod{10^9+7}$ 的方案数是多少。
$1 \le n \le 36$

填符号

case1：最中间填加号，两边分别 $2^n \times n$ 枚举然后用 unorderedmap 去找 $left = K - right$ 即可
case2：最中间填乘号
我们假如爆搜左半边的话，复杂度 $2^n \times n$，与右边有联系的地方就是后缀的一段乘号
右边需要给左边提供的信息是前缀的乘号延伸到哪里，以及除开这个的算式结果为多少。
具体来说就是形如 $A + B \times C + D$ 的结构，右边需要存到 $cnt[pos][D]$ 中 (第二维是哈希表)
然后左半边 dfs 到 (A,B) 时，需要枚举右半边的前缀位置 pos，用 $K - A - B * C$ 算出 D。
复杂度 $2^n \times n^2$

记忆化搜索

是搜索的一种，把结果记录在数组或哈希表中，下次再遇到相同的状态/局面时，把这个值取出直接返回。
某种程度上你可以看成是 DP 的逆向版。相比于 DP 的好处是直观好推、不需要遍历状态空间中的所有状态。

数位 DP 类

有些题是数位 DP 的标签，但实际上都会写成记忆化搜索模板的形式（因为这样好写）

小蓝正在一个瓜摊上买瓜。瓜摊上共有 n 个瓜，每个瓜的重量为 A_i。小蓝刀功了得，他可以把任何瓜劈成完全等重的两份，不过每个瓜只能劈一刀。

小蓝希望买到的瓜的重量的和恰好为 m。

请问小蓝至少要劈多少个瓜才能买到重量恰好为 m 的瓜。如果无论怎样小蓝都无法得到总重恰好为 m 的瓜，请输出 -1。

需要转成 $solve(r) - solve(l-1)$ 的形式。怎么写 solve ?

```
int solve(int x){//[0,x] 的和
    if (x==0) return 1;
    digit.clear();
    for (;x;x/=2) digit.push_back(x%2);
    return dfs(digit.size()-1,1,1,0);
}
```

Round Numbers S

P6218

记忆化搜索部分呢？

pos 表示从高往低做到了第几位，eq 代表前缀部分是否仍然处于相等状态，lead 代表是否还在前导 0

diff 部分就是你需要对题目维护的状态。（不同题目不一样）

需要 eq 不处于相等状态，lead 不在前导 0 的时候才能取记忆化的值，因为后面的数字可以保证去遍 $[0, B-1]$，并且方案数只和 pos 以及 diff 有关

```
map<pair<int,int>,int> rec;
vector<int> digit;
int dfs(int pos,int eq,int lead,int diff){
    if (pos==-1) return diff>=0;
    auto PAIR=make_pair(pos,diff);
    if (!eq && !lead && rec.count(PAIR)) return rec[PAIR];
    int ans=0;
    int up=eq?digit[pos]:1;
    for (int i=0;i<=up;i++){
        int nxt_eq=eq&(i==digit[pos]);
        int nxt_lead=lead&(i==0);

        int nxt_diff=diff;
        if (i==1) nxt_diff--;
        else if (!lead) nxt_diff++;

        ans=ans+dfs(pos-1,nxt_eq,nxt_lead,nxt_diff);
    }
    if (!eq && !lead) rec[PAIR]=ans;
    return ans;
}
```

windy 数

P2657

题目描述

不含前导零且相邻两个数字之差至少为 2 的正整数被称为 windy 数。windy 想知道，在 a 和 b 之间，包括 a 和 b ，总共有多少个 windy 数?

输入格式

输入只有一行两个整数，分别表示 a 和 b。

windy 数

P2657

怎么改上面的 dfs 代码？我们只用改 diff 为 las，代表上一个选的数位是什么
如果前导 0 状态还在，这个 las 是无效的
那就是在枚举 i 的时候如果 $las - 1 \le i \le las + 1$ 就 continue

手机号码

P4124

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，且不含前导的 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。

手机号码

P4124

维护什么状态呢？
是否出现 8，是否出现 4，上一个数字，上上一个数字是否和上一个数字相等，3 连任务是否做完。

more 记忆化？

别的 DP 的记忆化，像什么区间 DP 写记忆化也是很典型的。

石子合并

P1880

在一个圆形操场的四周摆放 N 堆石子，现要将石子有次序地合并成一堆，规定每次只能选相邻的 2 堆合并成新的一堆，并将新的一堆的石子数，记为该次合并的得分。

试设计出一个算法，计算出将 N 堆石子合并成 1 堆的最小得分和最大得分。

可以这样写：

```cpp
int solve(int l,int r){
	if (l==r) return 0;
	if (vis[l][r]) return rec[l][r];
	int ans=1e9;
	for (int mid=l;mid<=r;mid++){
		ans=min(ans,dp[l][mid]+dp[mid+1][r]+sum[l][r]);
	}
	vis[l][r]=1;
	return rec[l][r]=ans;
}
```

谢谢