第十三周讲题

Saya

T1 MCSP

利用 string 中的 substr 函数，枚举切割出来的 k 个字符串的大小，直接取出相应的串并重排拼接比较即可。

code

```
if (k == 3) {
    bool flag = false;
    for (int i = 1; i < s.size(); i++)
        for (int j = 1; i + j < s.size(); j++) {
            string s1 = s.substr(0, i);
            string s2 = s.substr(i, j);
            string s3 = s.substr(i + j);
            if (t == s1 + s2 + s3) flag = true;
            if (t == s1 + s3 + s2) flag = true;
            if (t == s2 + s1 + s3) flag = true;
            if (t == s2 + s3 + s1) flag = true;
            if (t == s3 + s1 + s2) flag = true;
            if (t == s3 + s2 + s1) flag = true;
        }
    cout << (flag ? "YES\n" : "NO\n");
}
```

T2 mms

被取模的数一定是最大的数，但模数不一定是最小的，需要枚举选择出模数应该选择哪一个。

时间复杂度 $O(n)/O(n\log n)$

code

```cpp
int main() {
    cin >> n;
    for (int i = 1; i <= n; i++) cin >> a[i], ans += a[i];
    int mx = 0;
    sort(a + 1, a + n + 1);
    for (int i = 1; i < n; i++) mx = max(mx, a[n] - a[n] % a[i]);
    cout << ans - mx << "\n";
    return 0;
}
```

T3 minmax

看到最大值最小是多少　→　二分

二分出一个值 x，计算在确保每一段最大子段和不超过 x 的情况下。最少需要划分出多少段。如果段数小于等于 k，则将 x 调小，否则将 x 调大。

code

```
int check(ll x) {
    int cnt = 1; ll s = 0;
    for (int i = 1; i <= n; i++) {
        s += a[i];
        if (s < 0) s = 0;
        if (s > x) {
            cnt++;
            s = a[i];
            if (a[i] > x) return k + 1;
        }
    }
    return cnt;
}
```

T4 gcd

先考虑序列上怎么做。

求一个区间所有子区间 GCD，暴力复杂度为 $O(n^2)$。

求所有前缀的所有子区间 GCD，可以递推，总时间复杂度也为 $O(n^2)$。

部分分 $\rightarrow$ 随机生成树高为 $O(\log n)$

T4 gcd

一个数和其它数取 gcd，得到的结果要么不变，要么至多变为原来的一半。

所以可能的子区间 GCD 的值不会很多，大量的子区间 GCD 是相同的，区间 [1,1],[1,2],[1,3],……,[1,n] 中区间 GCD 值至多只有 $\log n$ 种，且值相同的区间排列在一起。

值相同的区间我们一起计算即可。

code

```cpp
void dfs(int u, int fa) {
    vector<pair<int, int>> tmp;
    tmp.emplace_back(a[u], 1);
    for (auto [p, q] : s[fa]) {
        int o = gcd(p, a[u]);
        if (tmp.back().first == o) tmp.back().second += q;
        else tmp.emplace_back(o, q);
    }
    swap(s[u], tmp);
    ans[u] = ans[fa];
    for (auto [p, q] : s[u]) {
        ans[u] += 1ll * p * q;
    }
    res ^= ans[u];
    for (int v : adj[u]) {
        if (v == fa) continue;
        dfs(v, u);
    }
}
```

P6188 文具订购

给定一个正整数 n，找到三个正整数 a,b,c，满足7a+4b+3c=n。有多解的情况下，选取min(a,b,c)最大的解。若还有多解，则选取a+b+c最大的解。

$0 \leq n \leq 10^5$

Solution

任意一组解(a,b,c)，若b>0且c>0，则可以找到另一组解(a+min(b,c),b-min(b,c),c-min(b,c))。

这样后两个数，就至少有一个为0。故我们只要枚举第一个数a即可，然后再直接计算出c=0时b为多少（b=0时c为多少）。假定找到了一组解(a,b,0)，则它能转化出的最符合题目要求的解为(a/2,b+(a+1)/2,(a+1)/2)。

P6566 观星

给定一个 $n \times m$ 的地图，地图上.代表空位，*代表一颗星星。相邻（八联通）的星星会被视作在同一个星座，含有星星个数相同的星座会被视为在同一个星系中。要计算有多少个星系以及一个星系中最多有多少颗星星。

Solution

DFS+桶计数

从每一颗没被统计过的星星出发进行DFS，标记所有能走到的星星，把它们记在一个星座里，并统计每次DFS出发能到达的数量（即该星座有多少颗星星）。

开一个桶t[]，假设我们找到一个大小为cnt的星座。

则检查t[cnt]是否为0，若为0，则我们找到一个新的星系。

无论是否为0，都要执行t[cnt]+=cnt。

最后统计t数组中的最大值。

P6625 卡牌游戏

轩轩某天想到了一个卡牌游戏，游戏规则如下：

1. 初始时轩轩的手中有自左向右排成一排的n张卡牌，每张卡牌上有一个整数分值。
2. 接下来，轩轩每次可以选取卡牌序列最左边的连续若干张卡牌（至少2张），将它们替换为一张新卡牌。新卡牌将插入到序列的最左端，它的分值为本次操作中被替换掉的卡牌的分值之和。
3. 初始时轩轩总分为 0，每执行一次卡牌替换操作，新卡牌的分值将加到总分中。当序列长度为1时游戏结束，轩轩也可以在任意时刻结束游戏。

现在给出序列中各个卡牌的分值，请你来帮助轩轩计算他能够获得的最高总分是多少？

Solution

我们把所有被合并的前缀展开来看，题目条件改为下次合并的数量必须大于上一次合并的数量。

修改后的题目答案和之前是一样的。故我们可以直接把所有大于 0 的前缀和相加即可。

P6568 水壶

给定一个正整数序列 A，可以执行k次操作，每次操作可以令 A[i+1]+=A[i],A[i]=0。问在执行k次操作后，序列中最大的数是多少。

Solution

执行操作的顺序一定是从左往右连续的。

所以我们只要求所有长度为k+1的区间中，最大的区间和是多少即可。

做一遍前缀和可以快速求出区间和。

P10373 立方根

q 组询问，每次询问给出一个x_j，要求计算

$$\sum_{i=1}^{x_j} \lfloor i^{1/3} \rfloor$$

题目保证 x_j 升序给出，$1 \le x_j \le 10^{12}$。

Solution

因为定义域较大，所以直接对于每个数计算$\lfloor i^{1/3} \rfloor$复杂度太大。

但上式值域非常小，有大量的数计算完结果相同，所以我们可以枚举结果，计算这样的结果会有哪些数计算出来。

由于有多组询问，且x_i升序给出，所以当前询问的答案只要在上一个询问的答案上进行增加即可。

P1890 gcd区间

给定一个长度为 n 的正整数序列 a。

m组询问，每次询问给出一个区间[l,r]，要求输出这个区间所有数的最大公约数。

$1 \leq n \leq 10^3, 1 \leq m \leq 10^6$。

Solution

gcd(a,b,c)=gcd(gcd(a,b),c)

设一个数组 g[l][r]，表示区间(l,r)的最大公约数，则g[l][r]=gcd(g[l][r-1],a[r])。可以先用$O(n^2)$的时间预处理出g数组。

然后对于每一个询问都可以$O(1)$回答。

P6591 植树

给定一棵无根树T，要为其选定一个根。一个点可以成为根的条件是，选它作为根后，所有与它直接相连的所有子树大小相同。

要求找出所有的根。

$1 \leq n \leq 10^6$。

Solution

先假定1为根，从1出发进行DFS，计算以每个点u为根的子树大小sze[u]。

当我们以u为根时，对于所有与它相连的节点v：

若在以1为根的树上，u是v的父亲，则以u为根时，对应子树大小为sze[v]。

若在以1为根的树上，v是u的父亲，则以u为根时，对应子树大小为n-sze[u]。

T600217 最小幂模数

给定两个正整数 a, b，计算 $a^x \% b$(x为正整数) 的最小值。

$1 \leq a \leq 10^9, 1 \leq b \leq 10^5$。

Solution

当 x 依次等于 $1,2,3,\cdots\cdots$ 时， $a^x\%b$ 是存在循环节的，所以我们只要计算到一个模数出现两次时，就可以终止循环。

这个循环节大小会小于等于 b，故我们可以直接暴力。

T600218 区间和

给定一个长度为 n 的序列 $a_1, a_2, \cdots\cdots, a_n$。

要求找出一个区间，其区间和大于等于 s，且是所有满足条件的区间中长度最小的。

$1 \leq n \leq 10^5$。

Solution

记 $f(r)$ 为使得区间 $[l,r]$ 和大于等于 s 的最大 l。

若存在 $f(r)$，则 $f(r+1) \geq f(r)$。

故我们先找出最小的 r，使得 $[1,r]$ 的区间和大于等于 s 。

对于每个 r，都要计算使得区间 $[l,r]$ 和大于等于 s 的最大 l，由于 l 单调不降，所以可以在上一次的基础上修改。

总时间复杂度 $O(n)$。

T600216 最大最大子段和之和

给定一个长度为 n 的序列 $a_1, a_2, \cdots, a_n$ 。

要求将这个序列划分为 k 段，使每一段的最大子段和之和最大，输出这个最大和。

Solution

题目可以转化成，选取k个互不相交的段，使得选出来的所有数和最大。这样的题意限制更小，更方便进行动态规划。

设 dp[i][j][1/0] 表示在前 i 个数已选出 j 个段，第 i 个数是否被选出（1为被选出，0为未被选）的情况下选出数的总和最大是多少。

转移为：

```
dp[j][i][0] = max(dp[j][i - 1][0], dp[j][i - 1][1]);
dp[j][i][1] = max(dp[j][i - 1][1], dp[j - 1][i - 1][0]) + a[i];
```

T600215 最小栈

实现一个栈，要求其可以满足以下功能：

1. 将一个元素 x 入栈
2. 将栈顶元素出栈，并输出其值
3. 查询栈中最小元素，输出其值

Solution

记 S_1 为题目所说的栈，我们另外维护一个辅助的单调栈 S_2。

当我们将元素 x 入栈 S_1 时，同时比较栈 S_2 的栈顶元素和 x，若 x 小于 S_2 的栈顶元素，则将其入栈 S_2。

若出栈时，栈 S_1 和 S_2 的栈顶元素相同（为同一元素，不只是值相同），则 S_1 和 S_2 都执行出栈操作。

则 S_2 栈顶的值就是 S_1 栈中的最小值。

所有操作复杂度均为 $O(1)$。总时间/空间复杂度均为 $O(n)$。

P5686 和积和

定义函数 $S(l,r)=\sum_{i=l}^{r}a_i\times\sum_{i=l}^{r}b_i$。

要求计算 $\sum_{i=1}^{n}\sum_{j=1}^{n}S(i,j)\bmod(10^9+7)$的结果。

$3\le n\le 5\times 10^5$。

Solution

对于每一个 b_i ，计算它所在所有区间的 $\sum_{j=l}^{r} a_j$ 之和，将 $b_i \times \sum_{j=l}^{r} a_j$ 相加就可以得到答案。

定义两个数组 pre、suf，

pre[i]为a[i]*i的前缀和，suf[i]为a[i]*(n-i+1)的后缀和。

$\sum_{j=l}^{r} a_j$ 可以由式子 pre[i]*(n-i+1)+suf[i+1]*i 得到。

P7910 插入排序

给定长度为 n 的序列 a，要维护单点修改和进行插入排序后某一元素的排名。

单点修改次数不超过 5000。

$n \le 8000, Q \le 2 \times 10^5$。

Solution

根据数据范围可以猜测出，要求修改的复杂度为 O(n)，查询的复杂度为 O(1)。

同时维护原序列和排序后的序列（因为插入排序是稳定的，排序后的序列需要记录在原序列中的下标），修改时在原序列中修改并找到当前修改元素排序后在哪个位置，如果元素被改小，则往前调整（往前进行插入排序）；被改大则往后调整。

为了快速找到当前元素排序后在哪个位置，还要额外维护一个 pos 数组，pos[i] 表示原序列中第 i 个数当前在哪个位置，每次修改序列时，pos 数组也要对应修改。

P8809 近似 GCD

小蓝有一个长度为 n 的数组 A，数组的子数组被定义为从原数组中选出连续的一个或多个元素组成的数组。数组的最大公约数指的是数组中所有元素的最大公约数。

如果最多更改数组中的一个元素之后，数组的最大公约数为 g，那么称 g 为这个数组的近似 GCD。一个数组的近似 GCD 可能有多种取值。

小蓝想知道，数组 A 有多少个长度大于等于 2 的子数组满足近似 GCD 的值为 g。

$2 \leq n \leq 10^5$。

Solution

一个简单的贪心：为了让近似 GCD 尽可能接近 g，我们一定是选取一个元素把它修改为 g。

在这样修改后，一个子数组的近似 GCD 是否为 g，只要看数组中是否有数不是 g 的倍数即可。

所以我们把所有数字对 g 取模，一个子数组近似 GCD 是否可以为 g 只要看它内部是不是至多有一个数不为 g 的倍数。

维护一个数组 nex[i],表示从 i+1 开始，第一个不为 g 倍数的数的位置。以此计算以 i 开头的合法子数组长度最大为多少即可。

P2476 着色方案

有 n 个木块排成一行，从左到右依次编号为 1 至 n。

你有 k 种颜色的油漆，第 i 种颜色的油漆足够涂 c_i 个木块。

所有油漆刚好足够涂满所有木块，即 $\sum_{i=1}^{k} c_i = n$。

由于相邻两个木块涂相同色显得很难看，所以你希望统计任意两个相邻木块颜色不同的着色方案。

由于答案可能很大，请输出对 $10^9 + 7$ 取模的结果。

$1 \leq k \leq 15$，$1 \leq c_i \leq 5$。

Solution

本题方法较多，讲一种比较简单的方法。

注意到 c_i 较小，所以我们设计状态是可以按照它来考虑。

设 dp[a][b][c][d][e][las] 表示剩下1个的颜色有a种，剩下2个的颜色有b种，剩下3个的有c种，剩下4个的有d种，剩下5个的有e种，当前填的最后一个颜色剩下las个。

这样状态数是 $15^5 \times 6$ 的，可以接受。具体转移见代码，本题用记忆化搜索实现较为方便。

code

```
int dfs(int c1, int c2, int c3, int c4, int c5, int las) {
    if (dp[c1][c2][c3][c4][c5][las] != -1) return dp[c1][c2][c3][c4][c5][las];
    int res = 0;
    if (c1 > 0) res = (res + 1ll * (c1 - (las == 1)) * dfs(c1 - 1, c2, c3, c4, c5, 0)) % mod;
    if (c2 > 0) res = (res + 1ll * (c2 - (las == 2)) * dfs(c1 + 1, c2 - 1, c3, c4, c5, 1)) % mod;
    if (c3 > 0) res = (res + 1ll * (c3 - (las == 3)) * dfs(c1, c2 + 1, c3 - 1, c4, c5, 2)) % mod;
    if (c4 > 0) res = (res + 1ll * (c4 - (las == 4)) * dfs(c1, c2, c3 + 1, c4 - 1, c5, 3)) % mod;
    if (c5 > 0) res = (res + 1ll * (c5 - (las == 5)) * dfs(c1, c2, c3, c4 + 1, c5 - 1, 4)) % mod;
    return dp[c1][c2][c3][c4][c5][las] = res;
}
```

P2671 求和

一条狭长的纸带被均匀划分出了 n 个格子，格子编号从 1 到 n。每个格子上都染了一种颜色 $color_i$ 用 $[1,m]$ 当中的一个整数表示），并且写了一个数字 $number_i$。

定义一种特殊的三元组：(x,y,z)，其中 x,y,z 都代表纸带上格子的编号，这里的三元组要求满足以下两个条件：

1. x,y,z 都是整数，$x<y<z, y-x=z-y$。

2. $color_x=color_z$ 。

满足上述条件的三元组的分数规定为 $(x+z)\times(number_x+number_z)$。整个纸带的分数规定为所有满足条件的三元组的分数的和。这个分数可能会很大，你只要输出整个纸带的分数除以 10007 所得的余数即可。

Solution

题目的限制是在说：只要把颜色相同，并且奇偶相同的格子分作一类去计数即可。

将式子 $(x+z)\times(number_x+number_z)$ 展开为 $x\cdot number_x+z\cdot number_z+x\cdot number_z+z\cdot number_x$，求和可以得到 $\sum x\cdot\sum number_x+(cnt-2)\times\sum x\cdot number_x$。

`cnt` 指同一类格子的个数。

P2672 推销员

阿明是一名推销员，他奉命到螺丝街推销他们公司的产品。螺丝街是一条死胡同，出口与入口是同一个，街道的一侧是围墙，另一侧是住户。螺丝街一共有 N 家住户，第 i 家住户到入口的距离为 S_i 米。由于同一栋房子里可以有多家住户，所以可能有多家住户与入口的距离相等。阿明会从入口进入，依次向螺丝街的 X 家住户推销产品，然后再原路走出去。

阿明每走 1 米就会积累 1 点疲劳值，向第 i 家住户推销产品会积累 A_i 点疲劳值。阿明是工作狂，他想知道，对于不同的 X，在不走多余的路的前提下，他最多可以积累多少点疲劳值

Solution

性质 1：当阿明走到第 k 户再走回去时，一定是推销了第 k 户，以及所有 S 小于第 k 户的 A 前 $i-1$ 大个住户。

性质 2：$X=i$ 的答案一定是在 $X=i-1$ 的方案基础上增加一个住户。

所以我们只要每次贪心地寻找这个贡献最大的新住户即可。贪心寻找这一步可以用堆、线段树等，这里提供一种只需要排序的方法，具体见代码。

code

```
for (int i = 1; i <= n; i++) cin >> a[i], qa[i] = {a[i], i}, qs[i] = {2 * s[i] + a[i], i};
sort(qs + 1, qs + n + 1);
sort(qa + 1, qa + n + 1);
int j = n, k = n, mx = 0, res = 0;
for (int i = 1; i <= n; i++) {
    while (j >= 1 && qs[j].second <= mx) j--;
    while (k >= 1 && vis[qa[k].second]) k--;
    if (qs[j].first - 2 * s[mx] >= qa[k].first) {
        res -= 2 * s[mx];
        res += qs[j].first;
        mx = qs[j].second;
        vis[qs[j].second] = true;
    }
    else {
        res += qa[k].first;
        vis[qa[k].second] = true;
    }
    ans[i] = res;
}
```

P3956 棋盘

给定一个 m*m 的棋盘，棋盘上有些格子是黄色的（标记为1），有些格子是红色的（标记为0），有些格子是无色的。你需要从左上角走到右下角，任何时刻，你必须站在有颜色的格子上。在颜色相同的格子间移动不需要代价，在颜色不同的格子间移动要消耗一个金币，把无色格子暂时变成一个你想要的颜色要消耗两个金币（不能连续使用）。

要求走到右下角的最小花费。

Solution

设 f[x][y] 为从（1,1）走到(x,y）所需要的最小花费。

然后直接从起点开始进行搜索，分类讨论不同的走法需要多少金币，然后往相应的方向走（进行递归）。

搜索时加入一条记忆化剪枝，比较 f[x][y] 与 s（当前方案的花费），如果 f[x][y]<=s，则不用再搜索了，已经走过一种更好的方案了。

code

```
void dfs(int dx,int dy,int last,int s)
{
    if(dx==0||dx>m||dy==0||dy>m) return;
    if(last==0&&b[dx][dy]==0) return;
    if(f[dx][dy]<=s) return;
    else f[dx][dy]=s;
    for(int i=0;i<4;i++)
    {
        int w=dx+fx[i];
        int u=dy+fy[i];
        if(b[dx][dy]==0)
        {
            if(last==b[w][u]) dfs(w,u,last,s);
            else if(last!=b[w][u]&&b[w][u]) dfs(w,u,last,s+1);
        }
        if(b[w][u]==b[dx][dy]) dfs(w,u,b[dx][dy],s);
        else if(b[w][u]!=b[dx][dy]&&b[w][u]) dfs(w,u,b[dx][dy],s+1);
        else if(b[w][u]!=b[dx][dy]&&!b[w][u]) dfs(w,u,b[dx][dy],s+2);
    }
}
```

P8675 搭积木

给定一张 n*m 的图，上面有一些小明不喜欢的位置，小明要在上面搭积木，搭积木有三种规则：

规则 1：每块积木必须紧挨着放置在某一块积木的正上方，与其下一层的积木对齐；

规则 2：同一层中的积木必须连续摆放，中间不能留有空隙；

规则 3：小明不喜欢的位置不能放置积木。

小明想知道有多少种搭积木的方案（方案数对 10^9+7 取模）。

Solution

设 `dp[k][l][r]` 表示第 `k` 层位置 `l` 到位置 `r` 放置了积木的方案数。

转移为：`dp[k][l][r]+=dp[k-1][i][j](i<=l<=r<=j)`

朴素的转移是 $O(\mathrm{m}^2)$ 的，这样总复杂度达到了 $O(nm^4)$，不能通过本题。

注意到这个转移实际上是个二维前缀和，可以 $O(\mathrm{m}^2)$ 预处理出所有的，转移时再 $O(1)$ 查询。

P8792 最大公约数

给定一个数组，每次操作可以选择数组中任意两个相邻的数 x,y，并将其中一个替换为 $\gcd(x,y)$。问最少需要多少次才能让整个数组变为 1。

Solution

若数组中存在元素 1，则通过元素 1 把剩下所有非 1 数字变成 1 即可。

若不存在，则先操作出一个 1。

一个贪心：操作的序列一定是一个连续的区间。

所以我们对于每个固定的 l，都去计算最小的 r，满足区间 [l,r] 的最大公约数为 1。找到这个最小的区间长度然后再加上 n-1 就得到了答案。

找最小的符合条件的区间，需要使用ST表+二分/双指针。

P8865 种花

给定一张 n*m 的网格图，图上有些位置不能种花，要求计算出选定一个 C 型和 F 型的区域种花的方案数。

Solution

维护后缀和 suf[i][j],d[i][j],sumc[i][j],sumf[i][j]。

suf[i][j]为从(i,j)开始，在第i行选出一个可以种花连续区间有多少种方案。

sumc[i][j]=(suf[i][j]-1)+(suf[i+1][j]-1)+(suf[i+2][j]-1)+……+(suf[k][j]-1)（(k+1,j)为第一个不可以种花的位置）。

d[i][j]为从(i,j)开始，在第j列选出一个可以种花连续区间有多少种方案。

sumf[i][j]为(d[i][j]-1) * (suf[i][j]-1)在第j列的后缀和，同sumc，计算到不可以种花的位置就清零。

最后答案可以通过这几个数组计算得到，具体见代码。

code

```
for (int i = n; i >= 1; i--)
    for (int j = m; j >= 1; j--) {
        if (s[i][j] == '1') suf[i][j] = d[i][j] = sumc[i][j] = sumf[i][j] = 0;
        else {
            suf[i][j] = suf[i][j + 1] + 1, d[i][j] = d[i + 1][j] + 1;
            inc(sumc[i][j] = sumc[i + 1][j], suf[i][j] - 1);
            sumf[i][j] = (sumf[i + 1][j] + (d[i][j] - 1ll) * (suf[i][j] - 1ll)) % mod;
            if (i + 2 <= n && s[i + 1][j] == '0') {
                ansc = (ansc + (suf[i][j] - 1ll) * sumc[i + 2][j]) % mod;
                ansf = (ansf + (suf[i][j] - 1ll) * sumf[i + 2][j]) % mod;
            }
        }
    }
```

考试策略

- 仔细读题，算好时空限制，文件保存路径和名称要对仔细
- 根据数据范围推测算法时间复杂度，不会做的题目要留足时间写部分分
- 自己出小数据，可以的话尽量对拍
- 注意数组越界，long long等问题
- 注意多组数据
- 不会做的题、不会做的分也要勇敢地写暴力剪枝/贪心等做法，有时候会有良好的神秘效果