普瑞塞斯的游戏

tag

模拟

题解思路

读题可知，答案为寻找$\sum_{i=1}^{n} a_k = i(1 \leq k \leq n)$的答案，注意到$n(1 \leq n \leq 10^5)$，因此直接$n^2$暴力寻找不可行，我们仅需要将查询过程降低至$logn$即可，此处我们使用multiset去查询序列中是否存在$a_k = i$，若存在，答案加i，同时在multiset中删除该数即可。

参考代码

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
int main()
{
    int n, t;
    cin >> n;
    multiset<int> a;
    for (int i = 0; i < n; i++)
    {
        cin >> t;
        a.emplace(t);
    }
    ll ans = 0;
    for (int i = 1; i <= n; i++)
    {
        if (a.find(i) != a.end())
        {
            ans += i;
            a.erase(a.find(i));
        }
    }
    cout << ans;
}
```

一道水题(easy)

题解思路

一个很明显的搜索题，只要带着阀门的扩散系数一直向下搜就行了。一次搜索最大是$O(n)$，q次搜索就是$O(q * n)$次，明显符合题意，只要读懂题了，明显是道签到题啊👺👺👺

参考代码

```python
import sys
from collections import defaultdict
sys.setrecursionlimit(1000000000)#python需要这段代码打开dfs深度限制
class Graph:
    def __init__(self, n):
        self.adj = defaultdict(set)
        self.n = n
        self.edg=[]
        self.cnt=0
        self.dp=[0 for i in range(n+10)] #存一个点是否能被走到
    def addEdge(self, u, v,c):
        self.adj[u].add(self.cnt)
        self.adj[v].add(self.cnt)
        self.edg.append((u,v,c))
        self.cnt+=1
#以上为建图的代码，有点丑陋
#下面这个函数就是主体dfs函数
    def dfs(self,v,now,pa):
        self.dp[v]=1
        for i in self.adj[v]:
            x,y,c=self.edg[i]
            x=x if x!=v else y
            if now>=c and x!=pa:
                self.dfs(x,now-c,v)

n,m,q=map(int,input().split())
a=[int(i) for i in input().split()]
b=[int(i) for i in input().split()]
g=Graph(n)
for i in range(m):
    u,v,c=map(int,input().split())
    g.addEdge(u,v,c)
for i in b:
    g.dfs(i,a[i-1],-1)
cnt=0
for i in range(1,n+1):
    cnt+=g.dp[i]
print(cnt)
```

我穿越了

tag

dfs

题解思路

题意看着很长，实际上简化之后就是寻找元素x是否在一个环上，若在环上就有解，反之则无解.

参考代码

```cpp
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n, x;
    cin >> n >> x;
    vector<vector<int>> e(n + 1);
    vector<int> a(n + 1);
    for (int i = 1; i <= n; i++)
    {
        cin >> a[i];
        e[a[i]].push_back(i);
    }
    bool flg = 0;
    vector<int> ed(n + 1);
    function<void(int)> dfs = [&](int u)
    {
        if (ed[u])
        {
            flg = 1;
            return;
        }
        if (flg)
            return;
        ed[u] = 1;
        for (int v : e[u])
            dfs(v);
    };
    dfs(x);
    cout << (flg ? "Yes\n" : "No\n");
}
```

不取mod见祖宗

tag

数论分块

题解思路

注意到 n 是非常大的，因此我们无法直接暴力 $O(n)$ 求解。考虑优化求解过程：

首先，$n \mod i = n - \lfloor \frac{i}{n} \rfloor \times i$，原式可以展开为：

$$\sum_{i=1}^{n} n \mod i = \sum_{i=1}^{n} n^2 - \lfloor \frac{i}{n} \rfloor \times i = n^2 - \sum_{i=1}^{n} \lfloor \frac{i}{n} \rfloor \times i$$

我们发现对于序列 $1 \sim n$ 来说，$f(i) = \lfloor \frac{i}{n} \rfloor$ 的结果单调不增且呈块状分布，对于同一个块 $[l_i, r_i]$，满足 $\forall x, y \in [l_i, r_i] \lfloor \frac{x}{n} \rfloor = \lfloor \frac{y}{n} \rfloor$。假设这样的块有 k 个，可以证明 $k \leq 2N$：

证：

对于 $l_i \leq N$ 的区间，其个数不超过 N 个，因为 l_i 两两不同。
对于 $l_i > N$ 的区间，其个数也不超过 N 个，我们钦定 $v_i = \lfloor \frac{l_i}{n} \rfloor$，由于 $l_i > N$ 因此 v_i 一定小于等于 N，且 v_i 两两不同，因此满足条件的 i 不超过 N 个。
证明完毕。

同时，对于一个块 $[l_i, r_i]$ 内的和，即为 $\lfloor \frac{l_i}{N} \rfloor \times \sum_{j=l_i}^{r_i} j$，该表达式可以用等差数列求和 $O(1)$ 求出。

最终答案为：

$$n^2-\sum_{i=1}^{k}(\lfloor\frac{l_i}{N}\rfloor\times\sum_{j=l_i}^{r_i}j)$$

对于一个块，如何已知 l_i 如何求 r_i：

二分答案：利用 $\lfloor\frac{i}{n}\rfloor$ 的单调不增性质，可以二分求出 r_i。

根据划分的原理，不难发现，$r_i=\lfloor\frac{\lfloor\frac{l_i}{N}\rfloor}{N}\rfloor$。

以上两种方法均可通过本题（并没有卡掉多一个log的做法）。

参考代码

```cpp
#include <iostream>
using namespace std;
typedef long long i64;
int main() {
    i64 n;
    cin >> n;
    i64 sum = 0;
    i64 i = 1;
    while (i <= n) {
        i64 v = n / i;
        i64 r = n / v;
        sum += v * (i + r) * (r - i + 1) / 2;
        i = r + 1;
    }
    cout << n * n - sum << '\n';
    return 0;
}
```