K (AtCoder - abc315_e)

- 有 N 本书，第 i 本书有 C_i 本前置读物 $P_{i,1}, P_{i,2} \dots P_{i,c_i}$ 。
- 现在你想读第 1 本书，问在读第 1 本书之前至少还要读哪些书。
- 输出读它们的顺序，多解输出任意一个。
- 保证有解，$1 \leq N, \sum C_i \leq 2 \times 10^5$ ，其他数据在合理范围内。

分析样例，在读第一本书前，先读第二、三、四本，读第二本要读第三、五本，读第四本要读第五本，可知答案符合要求。

根据上面的分析，按照读书的先后顺序，不难想到建图和拓扑排序。然而，本题中有一个关键问题，需要**满足最少的前提下**的读书方案，如果直接拓扑，会导致不需要读的书也被输出。

不过，上面的办法并非毫不可取，我们可以看出上面方法的根本问题在于无法在拓扑时确定当前的点与第一个点是否有关，反过来想，我们只需要保证在某个点后，所有点都和第一个点有关，不就行了？所以，只需**建反图，然后拓扑排序，从一号点开始都存储到数组，最后反过来输出。**

最后，有一个小问题需要解决：要保证在一号点之后都是必需的。其实只需要用从大到小的优先队列，在一号点被出队之前，其他能出队的点都会出队，那么一号点出队后，剩下的都是图中被一号约束的点，也就是必读书。

L (AtCoder - abc351_f)

题目描述：

$$\sum_{i=1}^{N} \sum_{j=i+1}^{N} \max(A_j - A_i, 0)$$

- $2 \leq N \leq 4 \times 10^5$
- $0 \leq A_i \leq 10^8$

N 范围在 $4e5$, $O(n^2)$ 复杂度显然不可解

解法一:（拆式子，不需要数据结构）参考https://www.luogu.com.cn/article/ls4uwhns

解法二：（需要用到树状数组或是线段树维护） 观察到该式类似求解逆序对或正序对，若考虑只统计逆序对的个数，则只需使用一个树状数组维护当前各个数字出现的个数。 于是像求解逆序对一样从左向右遍历，当计算到 i 位置时，考虑当前**小于 A_i 的数**与 A_i 组合做出的贡献，易得该贡献可计算为

> 所有当前（当前指 A_1 到 A_{i-1}）小于A_i的数的个数 * A_i - 所有当前小于A_i的数的和

于是采用两个树状数组，一个维护次数，一个维护和 当然注意到 A_i 在 $1e8$ 的范围，需要先离散化再建立树状数组

参考代码如下：

```
// Created by qyy on 2024/4/27.

#include <bits/stdc++.h>
```

```
using namespace std;

typedef long long ll;

#define PII pair<int, int>
#define endl "\n"

const long long inf = 0x3f3f3f3f3f3f3f3f;
const int N = 4e5 + 10;
const int mod = 1e9 + 7;

ll n, a[N];
struct point{int num; ll val;}p[N];
bool cmp(point x, point y){
    if(x.val == y.val) return x.num < y.num;
    else return x.val < y.val;
}

#define lowbit(x) (x & -(x))
ll tree1[N], tree2[N];
int rk[N];
void update(int x, ll d1, ll d2){
    while(x <= N){
        tree1[x] += d1;
        tree2[x] += d2;
        x += lowbit(x);
    }
}
ll sum1(int x){
    ll ans = 0;
    while(x > 0){
        ans += tree1[x];
        x -= lowbit(x);
    }
    return ans;
}
ll sum2(int x){
    ll ans = 0;
    while(x > 0){
        ans += tree2[x];
        x -= lowbit(x);
    }
    return ans;
}

void solve() {
    cin >> n;
    for(int i = 1; i <= n; i++){
        cin >> a[i];
        p[i].val = a[i];
        p[i].num = i;
    }
    sort(p + 1, p + 1 + n, cmp);
```

```
    for(int i = 1; i <= n; i++) rk[p[i].num] = i;

    ll ans = 0;
    for(int i = 1; i <= n; i++){
        update(rk[i], 1, p[rk[i]].val);
        ans += sum1(rk[i] - 1)*p[rk[i]].val - sum2(rk[i] - 1);
    }
    cout << ans << endl;
}

signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    int t = 1;
    //cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```