双指针

基本概念

双指针是一种简单而又灵活的技巧和思想，单独使用可以轻松解决一些特定问题，和其他算法结合也能发挥多样的用处。

双指针顾名思义，就是同时使用两个指针，在序列、链表结构上指向的是位置，在树、图结构中指向的是节点，通过或同向移动，或相向移动来维护、统计信息。

我们将以下面几道题作为引入，讲解双指针的关键用法和复杂度分析

49761 「CF EDU 双指针 Step1 A」Merging Arrays

题目描述

给定两个数组，它们以非递减顺序排序。将它们合并成一个有序数组。要求用 $O(n)$ 算法实现。

输入格式

第一行包含整数 n 和 m，即两个数组的大小（$1 \leq n, m \leq 10^5$）。

第二行包含 n 个整数 a_i，为第一个数组的元素；

第三行包含 m 个整数 b_i，为第二个数组的元素（$-10^9 \leq a_i, b_i \leq 10^9$）。

输出格式

打印 $n + m$ 个整数，即合并后的数组。

样例输入

```
6 7
1 6 9 13 18 18
2 3 8 13 15 21 25
```

样例输出

```
1 2 3 6 8 9 13 13 15 18 18 21 25
```

Solution

首先，我们使用两个**整数** $p1, p2$ 代表我们**即将添加到有序数组中**的下一个元素的下标，即即将添加的元素分别是 a_{p1}, b_{p2} 。

正确性毋庸置疑，因为 $\forall p > p1, a_p > a_{p1}$ ，因此新的有序数组的后一个元素不可能是 a_p；对 b 数组同理

因此， 新的有序数组的下一个元素，必然从 a_{p1}, b_{p2} 之中产生，我们选择其中较小的一个就可以了，这个算法在**归并排序**中被广泛地应用

由于指针 $p1, p2$ 至多移动 n, m 次，且每次移动的开销是 $O(1)$，因此时间复杂度是 $O(n+m)$

需要注意的是，在归并排序中，我们假设 a 数组是原数组的**前半段**， b 数组是原数组的**后半段**，那么，在合并的时候，$a_{p1} = b_{p2}$ 时，若我们选择 a_{p1} 添加进新的有序数组，那么我们就称此时的排序是**稳定的**

一个排序被称为**稳定的**，当且仅当假设原数组（设为 a ）中任意的一个下标集合 $b_1 < b_2 < \cdots < b_m$ ，使得 $a_{b_1} = a_{b_2} = \cdots = a_{b_m}$ （这里的相同是在我们自定义的 `cmp` 函数下，$\text{cmp}(a, b) = \text{cmp}(b, a)$），假设这些相同的元素在排序后的下标集合为 $c_1, c_2, \cdots c_m$ （其中 a_{b_i} 被排到了位置 c_i ），且有着 $c_1 < c_2 < \cdots c_m$ 。

换言之，就是说，对于数组里**相同的元素**，他们在**原数组**中是什么顺序，在**排序后的数组**中就是什么顺序

Code

```
void solve() {
    int n, m; cin >> n >> m;
    vector a(n, 0), b(m, 0);
    for (auto& v: a) cin >> v;
    for (auto& v: b) cin >> v;

    int p1 = 0, p2 = 0;
    while (p1 < n && p2 < m) {
        if (a[p1] < b[p2]) cout << a[p1++] << ' ';
        else cout << b[p2++] << ' ';
    }

    while (p1 < n) cout << a[p1++] << ' ';
    while (p2 < m) cout << b[p2++] << ' ';
}
```

49762 「CF EDU 双指针 Step1 B」Number of Smaller

题目描述

给定两个数组，它们以非递减顺序排序。对于第二个数组中的每一个元素，寻找第一个数组中严格小于它的元素的个数。

输入格式

第一行包含整数 n 和 m，即两个数组的大小（$1 \le n, m \le 10^5$）。

第二行包含 n 个整数 a_i，为第一个数组的元素；

第三行包含 m 个整数 b_i，为第二个数组的元素（$-10^9 \le a_i, b_i \le 10^9$）。

输出格式

打印 m 个数字，表示第一个数组中小于第二个数组中每个元素的元素数量。

样例输入

```
6 7
1 6 9 13 18 18
2 3 8 13 15 21 25
```

样例输出

```
1 1 2 3 4 6 6
```

Solution

首先，我们观察到，如果说我们从小到大遍历 b 数组的元素的话，答案一定是单调不减的，而且比当前元素小的元素一定集中在 a 数组的**头部**

因此，我们选用一个指针 $p2$ ，从小到大地遍历 b 数组，同时选择一个指针 $p1$，让 $p1$ 指向 a 数组中，首个比 b_{p2} 大的元素（或指向 a 数组末尾），每次 $p2$ 向后移一个单位时，我们就将 p_1 也后移以满足题设，时间复杂度为 $O(n+m)$

Code

```
void solve() {
    int n, m; cin >> n >> m;
    vector a(n, 0), b(m, 0);
    for (auto& v: a) cin >> v;
    for (auto& v: b) cin >> v;
    a.push_back(inf);

    for (int p1 = 0, p2 = 0; p2 < m; p2++) {
        while (b[p2] > a[p1]) p1++;
        cout << p1 << ' ';
    }
}
```

49763 「CF EDU 双指针 Step1 C」Number of Equal 相等的数

题目描述

给定两个数组，它们以非递减顺序排序。找出满足 $a_i = b_j$ 的数对 (i, j) 的个数。

输入格式

第一行包含整数 n 和 m，即两个数组的大小（$1 \le n, m \le 10^5$）。

第二行包含 n 个整数 a_i，为第一个数组的元素；

第三行包含 m 个整数 b_i，为第二个数组的元素（$-10^9 \le a_i, b_i \le 10^9$）。

输出格式

输出 1 个整数，表示答案。

样例输入

```
8 7
1 1 3 3 3 5 8 8
1 3 3 4 5 5 5
```

样例输出

```
11
```

Solution

首先，我们仍然使用两个指针 $p1, p2$，分别指向 a, b 两个数组的首地址

对于 a 数组，我们首先统计出所有 $a_i = a_{p_1}$ 的数字的个数，并将其记为 $l1$，同时，我们在 b 数组中，利用 $p2$ 统计所有 $b_i = a_{p1}$ 的个数（具体操作步骤可以是，首先**略过**所有 $b_i < a_{p1}$ 的部分，再统计相等的部分），将这部分记为 $l2$，那么，这两部分对答案的贡献就是 $l1 \times l2$，注意到 $l1, l2$ 最大可能到 10^5，因此我们需要用 `long long` 数据类型

Code

```cpp
void solve() {
    int n, m; cin >> n >> m;
    vector a(n, 0), b(m, 0);
    for (auto& v: a) cin >> v;
    for (auto& v: b) cin >> v;
    a.push_back(inf);

    ll ans = 0;
    for (int p1 = 0, p2 = 0; p2 < m; p2++) {
        while (a[p1] < b[p2]) p1++;
        ll l1 = p1, l2 = 1;
        while (a[p1] <= b[p2]) p1++;
        l1 = p1 - l1;
        while (p2 + 1 < m && b[p2 + 1] == b[p2]) p2++, l2++;
        ans += l1 * l2;
    }

    cout << ans << endl;
}
```

49768 [「CF EDU 双指针 Step2 A」Segment with Small Sum]

题目描述

给定一个包含 n 个整数 a_i 的数组。如果这个数组的一个子段 $a[l..r]$（$1 \le l \le r \le n$）的元素总和至多为 s，则称这个子段是好的。你的任务是找到最长的好子段。

输入格式

第一行包含整数 n 和 s（$1 \le n \le 10^5, 1 \le s \le 10^{18}$）。

第二行包含整数 a_i（$1 \le a_i \le 10^9$）。

输出格式

打印一个整数，即最长好子段的长度。如果没有这样的段，则打印 0。

样例输入

```
7 20
2 6 4 3 6 8 9
```

样例输出

```
4
```

Solution

此题就是双指针的另一种典型应用

一个很显然的暴力做法是，枚举字段的所有起点 $1, 2, \cdots, n$，每个起点都会有一个对应的，最远的终点，我们将其记为 $r_1, r_2, \cdots, r_n$

显然，$r_1 \le r_2 \le \cdots \le r_n$，因为 $\sum_{j=i}^{r_i} \le \sum_{j=i+1}^{r_i}$，如果 $\sum_{j=i}^{r_i} \le s$，同样的，也有 $\sum_{j=i+1}^{r_i} \le s$，因此，我们 $r_i \le r_{i+1}$ 永远成立

那么，我们就可以考虑使用两个指针 $p1, p2$，其中一个指向我们的**区间起点**，另一个指向我们的**区间终点**，每次，区间起点往后挪一位，将 a_{p1} 从和里面删去，接着，我们尝试添加 a_{p2} 进入集合，如果说发现 $sum + a_{p2} > s$，那说明我们 $p2$ 不能继续后移，那我们统计此时的长度，并且和答案取 $\max$ 即可

可以看到，我们双指针在这种情况下，维护的是一个**可删除**，**可添加**的，**单调的**区间信息，对于区间和而言，显然，我们删除和添加都很方便

此时，左指针一共移动了 $O(n)$ 次，右指针一共挪动了 $O(n)$ 次，加数、删数的操作都是 $O(1)$ 的，因此算法的时间复杂度为 $O(n)$

Code

```
void solve() {
    int n; ll s; cin >> n >> s;
    vector a(n, 0);
    for (auto& v: a) cin >> v;

    int ans = 0;
    for (int p1 = 0, p2 = 0; p1 < n; p1++) {
        while (p2 < n && s - a[p2] >= 0) {s -= a[p2]; p2++;}
        cmax(ans, p2 - p1);
```

```
        s += a[p1];
    }

    cout << ans << endl;
}
```

49769 「CF EDU 双指针 Step2 B」Segment with Big Sum

题目描述

给定一个包含 n 个整数 a_i 的数组。如果这个数组的一个子段 $a[l..r]$（$1 \le l \le r \le n$）的元素总和至少为 s，则称这个子段是好的。你的任务是找到最短的好子段。

输入格式

第一行包含整数 n 和 s （$1 \le n \le 10^5, 1 \le s \le 10^{18}$）。

第二行包含整数 a_i （$1 \le a_i \le 10^9$）。

输出格式

打印一个整数，即最长好子段的长度。如果没有这样的段，则打印 -1。

样例输入

```
7 20
2 6 4 3 6 8 9
```

样例输出

```
3
```

Solution

本题和上题较为类似，唯一不同的是字段和必须**大于** s

注意坑点是，当右指针 $p2$ 移到数组末尾时，若仍无法满足题目所设条件，那我们就**不更新**答案

Code

```
void solve() {
    int n; ll s; cin >> n >> s;
    vector a(n, 0);
    for (auto& v: a) cin >> v;

    int ans = inf;
    for (int p1 = 0, p2 = 0; p1 < n; p1++) {
        while (p2 < n && s > 0) {s -= a[p2]; p2++;}
        if (s <= 0) cmin(ans, p2 - p1);
        s += a[p1];
    }
```

```
    cout << (ans == inf ? -1 : ans) << endl;
}
```

49770 「CF EDU 双指针 Step2 C」Number of Segments with Small Sum

题目描述

给定一个包含 n 个整数 a_i 的数组。如果这个数组的一个子段 $a[l..r]$（$1 \le l \le r \le n$）的元素总和至多为 s，则称这个子段是好的。你的任务是找到好子段的数量。

输入格式

第一行包含整数 n 和 s（$1 \le n \le 10^5, 1 \le s \le 10^{18}$）。

第二行包含整数 a_i（$1 \le a_i \le 10^9$）。

输出格式

打印一个整数，即好子段的数量。

样例输入

```
7 20
2 6 4 3 6 8 9
```

样例输出

```
19
```

Solution

对于每个左端点 i，我们找到最右边的，满足 $\sum_{j=i}^{r_i} a_j \le s$ 的 r_i，很明显，区间 $[i, r_i], [i, r_i - 1] \cdots [i, i]$ 均满足要求，此时，以 i 为左端点的，满足题设要求的区间个数为 $r_i - i + 1$ 个，将其累加进入 ans 即可

需要注意的是，如果我们双指针 $p1, p2$ 满足**左闭右开**原则，即 $p1, p2$ 代表的区间是 $[a_{p1}, a_{p2})$，那么，我们在统计区间长度等操作会比较方便

Code

```
void solve() {
    int n; ll s; cin >> n >> s;
    vector a(n, 0);
    for (auto& v: a) cin >> v;

    ll ans = 0;
    for (int p1 = 0, p2 = 0; p1 < n; p1++) {
        while (p2 < n && s >= a[p2]) {s -= a[p2]; p2++;}
        ans += (p2 - p1);
        s += a[p1];
```

```
    }

    cout << ans << endl;
}
```

49771 「CF EDU 双指针 Step2 D」Number of Segments with Big Sum

题目描述

给定一个包含 n 个整数 a_i 的数组。如果这个数组的一个子段 $a[l..r]$（$1 \le l \le r \le n$）的元素总和至少为 s，则称这个子段是好的。你的任务是找到好子段的数量。

输入格式

第一行包含整数 n 和 s （$1 \le n \le 10^5, 1 \le s \le 10^{18}$）。

第二行包含整数 a_i （$1 \le a_i \le 10^9$）。

输出格式

打印一个整数，即好子段的数量。

样例输入

```
7 20
2 6 4 3 6 8 9
```

样例输出

```
9
```

Solution

该题和上题比较类似，我们仍然找到从 i 开始的，最小的，和大于 s 的区间 $[i, r_i]$ ，那么，显然，以 i 为左端点的区间中， $[i, r_i], [i, r_i + 1] \cdots [i, n]$ 均满足要求，这样的区间共有 $n - r_i + 1$ 个，将其统计进答案即可

Code

```
void solve() {
    int n; ll s; cin >> n >> s;
    vector a(n, 0);
    for (auto& v: a) cin >> v;

    ll ans = 0;
    for (int p1 = 0, p2 = 0; p1 < n; p1++) {
        while (p2 < n && s > 0) {s -= a[p2]; p2++;}
        if (s <= 0) ans += n - p2 + 1;
        s += a[p1];
    }
```

```
    cout << ans << endl;
}
```