

离散化

离散化的使用场景

例题: <https://www.acwing.com/problem/content/description/804/>

考虑这样一个问题: 假定有一个无限长的数轴, 数轴上每个坐标上的数都是0。现在, 我们首先进行 n 次操作, 每次操作将某一位置 x 上的数加 c 。接下来, 进行 m 次询问, 每个询问包含两个整数 l 和 r , 你需要求出在区间 $[l, r]$ 之间的所有数的和。

数据范围方面:

$$-10^9 \leq x \leq 10^9 \quad 1 \leq n, m \leq 10^5 \quad -10^9 \leq l \leq r \leq 10^9 \quad -10000 \leq c \leq 10000$$

如果 x 的数据范围仅仅是非负整数, 并且范围比较小的话, 那我们可以直接开一个数组`a`, 然后每次`a[x] += c`, 求区间和的话可以用前缀和数组来快速求出。

但是这道题的困难之处就在于 x 的范围非常大, 并且还包含负数, 如果强行开数组, 会导致空间非常浪费。但我们同时注意到, 整个用到的坐标的数量实际上最多是 $10^5 + 2 \times 10^5 = 3 \times 10^5$, 这个长度的数组空间方面是可以接受的。这里我们就需要“离散化”来解决了。

离散化的思想就是, 先将需要用到的数都映射到范围较小的数, 一般就直接映射到 $1, 2, 3, \dots$ 。具体做法是:

1. 先将所有要用到的数存入一个vector
2. 将vector排序, 然后去重
3. 开一个哈希表存储从离散化之前到之后的映射关系。当然从离散化之后到之前的映射关系已经可以通过上面的vector来得到了

代码如下:

```
1 // xs存了所有需要用到的离散化之前的数, 先排序
2 sort(xs.begin(), xs.end());
3 // 去重
4 xs.erase(unique(xs.begin(), xs.end()), xs.end());
5 // 开一个unordered_map, 存储一下离散化的映射关系。一般映射到1,2,...
6 mp.reserve(xs.size());
7 for (int i = 0; i < xs.size(); i++) mp.emplace(xs[i], i + 1);
```

后面应答询问的时候, 可以先将询问的左右边界求成离散化之后的值, 然后根据新值来应答询问即可。代码如下:

```

1  #include <algorithm>
2  #include <iostream>
3  #include <unordered_map>
4  #include <vector>
5  using namespace std;
6  using PII = pair<int, int>;
7
8  const int N = 3e5 + 10;
9  int n, m;
10 int a[N], s[N];
11 vector<int> xs;
12 vector<PII> adds, qs;
13 unordered_map<int, int> mp;
14
15 int main() {
16     scanf("%d%d", &n, &m);
17     xs.reserve(n + 2 * m);
18     for (int i = 1; i <= n; i++) {
19         int x, c;
20         scanf("%d%d", &x, &c);
21         adds.emplace_back(x, c);
22         xs.push_back(x);
23     }
24
25     for (int i = 1; i <= m; i++) {
26         int l, r;
27         scanf("%d%d", &l, &r);
28         qs.emplace_back(l, r);
29         xs.push_back(l), xs.push_back(r);
30     }
31
32     sort(xs.begin(), xs.end());
33     xs.erase(unique(xs.begin(), xs.end()), xs.end());
34     mp.reserve(xs.size());
35     for (int i = 0; i < xs.size(); i++) mp.emplace(xs[i], i + 1);
36
37     for (auto [x, c] : adds) a[mp[x]] += c;
38     for (int i = 1; i <= xs.size(); i++) s[i] = s[i - 1] + a[i];
39
40     for (auto [l, r] : qs) printf("%d\n", s[mp[r]] - s[mp[l] - 1]);
41 }

```

作业

上面例题