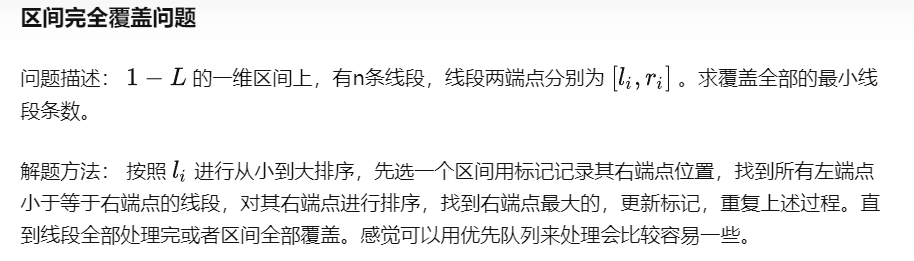
线段覆盖

给定x轴上的N（0<N<100）条线段，每个线段由它的二个端点a\_I和b\_I确定，I=1,2,……N.这些坐标都是区间（－999，999）的整数。有些线段之间会相互交叠或覆盖。请你编写一个程序，从给出的线段中去掉尽量少的线段，使得剩下的线段两两之间没有内部公共点。所谓的内部公共点是指一个点同时属于两条线段且至少在其中一条线段的内部（即除去端点的部分）。

基础线段覆盖模型 -> 贪心

直接按照右端点从小到大排序。优先选排在前面的。

证明：排序后显然当你选了一个线段后，再要从后面选一个线段，其左断点必须>=你的右端点。右端点当然越小越好。



上面这种情况是线段不带权值的，那么假如线段是带有权值的呢？

题目：

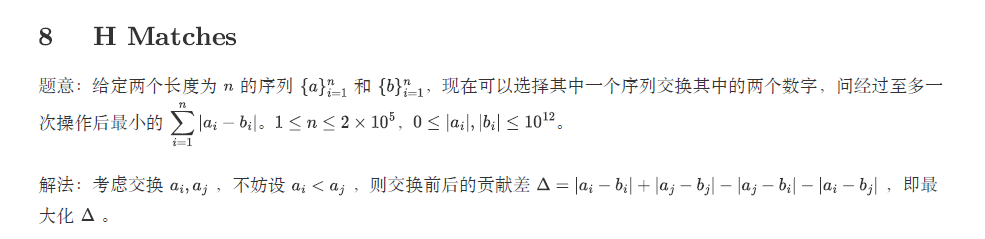
线段覆盖：数轴上有n条线段，线段的两端都是整数坐标，坐标的范围为0~1000000，每条线段有一个价值，请从n条线段中挑出若干条线段，使得这些线段两两不覆盖（端点可以重合），且线段价值之和最大。输入第一行一个整数n(n<=1000)，表示有多少条线段接下来的n行，每行3个整数ai,bi,ci，分别代表第i条线段的左端点ai,右端点bi,价值ci。输出能够获得的最大价值。

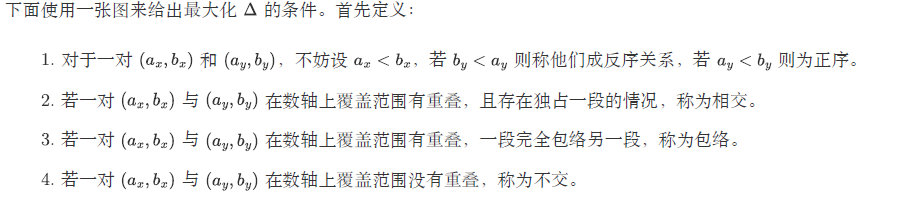
思路：

①首先定义一个结构体Line，存放各线段的左端点、右端点以及其价值；

②再将线段按右端点的升序进行排序;定义一个maxValue数组，maxValue[i]表示以线段 i 为右边界的线段最大价值。

③对于每一条线段 i，计算其最大价值的公式为：maxValue[i] = max{ maxValue[i-1] , line[i].v + maxValue[j] }，其中线段 j 为线段 i 之前最靠近线段 i 且与线段 i 不重合的线段。line[i]是一个结构体，line[i].v表示线段 i 的价值。因此可以按照排序后的下标从小到大对线段进行遍历，求其最大价值。





然后可以将两部分线段按左端点排序,然后跑一个双指针,每次计算相交值,就行,本质上是个贪心

#include <iostream>

#include <algorithm>

using namespace std;

const int N = 1e6;

using P = pair<int, int>;

using ll = long long;

int a[N], b[N];

P x[N], y[N];

int n;

int xl, yl;

void solve()

{

for (int i = 0; i < n; i++)

if (a[i] < b[i])

x[xl++] = {a[i], b[i]};

else

y[yl++] = {b[i], a[i]};

sort(x, x + xl);

sort(y, y + yl);

ll ans = 0;

int px = 0, py = 0;

while (px < xl && py < yl)

{

auto [xa, xb] = x[px];

auto [ya, yb] = y[py];

ll overlap = max(0, min(xb, yb) - max(xa, ya));

ans = max(ans, overlap);

if (xb < yb)

px++;

else

py++;

}

ans = -(ans \* 2);

for (int i = 0; i < n; i++)

ans += abs(a[i] - b[i]);

cout << ans << '\n';

}

int main()

{

ios::sync\_with\_stdio(false);cin.tie(0);

cin >> n;

for (int i = 0; i < n; i++)

cin >> a[i];

for (int i = 0; i < n; i++)

cin >> b[i];

solve();

return 0;

}