区间贪心问题

人员

柳力玮、李瑞涵、刘宸熙、王馨琪、初锦阳、赵牧之、韩昱辰、苑钊、田心一、蒋叔璋、刘派、宋云朗、高健桓 到课, 刘子轩 线上

上周作业检查

上周作业链接: https://cppoj.kids123code.com/contest/795

2025-0914 周日15:30 (综合练习)

刷新

#	用户名	姓名	编程分	时间	A	B	C	D	E
1	yuanzhao	苑钊	500	26690	100	100	100	100	100
2	tianxinyi	田心一	500	29431	100	100	100	100	100
3	zhaomuzhi	赵牧之	407	24418	100	100	100	100	7
4	liuliwei	柳力玮	403	24566	100	100	100	100	3
5	hanyuchen	韩昱辰	400	24950	100	100	100	100	0
6	gaojianhuan	高健桓	400	25415	100	100	100	100	
7	wangenze	王恩泽	300	11038	100	100	100		
8	chujinyang	初锦阳	300	14758	100	100	100		
9	wangxinqi	王馨琪	300	19135	100	100	100		
10	jiangshuzhang	蒋叔璋	300	19258	100	100	100		
11	liuchenxi	刘宸熙	300	19266	100	100	100		
12	liruihan	李瑞涵	300	22028	100	100	100		
13	liupai	刘派	204	13285	100	100	4		
14	jianghaoxuan	姜皓轩	100	2979	100				
15	songyunlang	宋云朗	100	6924		100			

您正在共享屏幕 结束共享

本周作业

https://cppoj.kids123code.com/contest/848 (课上讲了 A ~ F 题, 课后作业是 G 题)

课堂表现

今天的 F 题 区间覆盖 会相对比较难一些, 这个题同学们实在不好写没有关系

A ~ E 这 5 道题非常重要而且是很经典的题目, 这 5 道题同学们一定要掌握。

课堂内容

Ghost Ants (上周作业)

上周作业这道题目确实比较有难度一些, 思路是把 向左的蚂蚁 和 向右的蚂蚁 进行分类, 然后针对每个向左的蚂蚁, 可以用 二分 求他会跟多少个 向右的蚂蚁 相交

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

using namespace std;
```

```
typedef long long LL;
const int maxn = 2e5 + 5;
char s[maxn];
int w[maxn];
vector<LL> vec[2];

int calc(LL l, int r) {
  int lpos = lower_bound(vec[1].begin(), vec[1].end(), l) - vec[1].begin();
  int rpos = upper_bound(vec[1].begin(), vec[1].end(), r) - vec[1].begin() - 1;
  return rpos - lpos + 1;
}

int main()
{
  int n, t; cin >> n >> t;
  cin >> (s+1);
  for (int i = 1; i <= n; ++i) {
    cin >> w[i]; vec[s[i]-'0'].push_back(w[i]);
  }

  sort(vec[1].begin(), vec[1].end());

  LL res = 0;
  for (LL i : vec[0]) res += calc(i-2*t, i-1);
  cout << res << endl;
  return 0;
}
```

凌乱的yyy / 线段覆盖

贪心, 按照右端点排序, 如果一个区间的左端点跟前面选过的点不重合, 这个区间就是要选的

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

using namespace std;

const int maxn = 1e6 + 5;
struct node {
  int a, b;
} w[maxn];
bool cmp(node p, node q) { return p.b < q.b; }

int main()
{
  int n; cin >> n;
  for (int i = 1; i <= n; ++i) cin >> w[i].a >> w[i].b;
  sort(w+1, w+n+1, cmp);

  int res = 0, last = -1e9;
  for (int i = 1; i <= n; ++i) {
    if (w[i].a >= last) last = w[i].b, ++res;
```

```
  }
  cout << res << endl;
  return 0;
}
```

区间选点

本质上跟上一道题目一样, 只是 >= 和 > 的区别

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

using namespace std;

const int maxn = 1e6 + 5;
struct node {
  int a, b;
} w[maxn];
bool cmp(node p, node q) { return p.b < q.b; }

int main()
{
  int n; cin >> n;
  for (int i = 1; i <= n; ++i) cin >> w[i].a >> w[i].b;
  sort(w+1, w+n+1, cmp);

  int res = 0, last = -1e9-10;
  for (int i = 1; i <= n; ++i) {
    if (w[i].a > last) last = w[i].b, ++res;
  }
  cout << res << endl;
  return 0;
}
```

最大不相交区间数量

跟上一道题一模一样

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

using namespace std;

const int maxn = 1e6 + 5;
struct node {
  int a, b;
} w[maxn];
bool cmp(node p, node q) { return p.b < q.b; }

int main()
{
```

```
  int n; cin >> n;
  for (int i = 1; i <= n; ++i) cin >> w[i].a >> w[i].b;
  sort(w+1, w+n+1, cmp);

  int res = 0, last = -1e9-10;
  for (int i = 1; i <= n; ++i) {
    if (w[i].a > last) last = w[i].b, ++res;
  }
  cout << res << endl;
  return 0;
}
```

区间分组

想把区间分成最少的组, 就是看在哪个点上重合的区间数最多

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

using namespace std;

const int maxn = 2e5 + 5;
struct node {
  int pos, value;
} w[maxn];
bool cmp(node p, node q) {
  if (p.pos != q.pos) return p.pos < q.pos;
  return p.value < q.value;
}

int main()
{
  int n; cin >> n;
  for (int i = 1, id = 0; i <= n; ++i) {
    int l, r; cin >> l >> r;
    w[++id] = {l, 1}, w[++id] = {r+1, -1};
  }
  sort(w+1, w+2*n+1, cmp);

  int res = 0, sum = 0;
  for (int i = 1; i <= 2*n; ++i) {
    sum += w[i].value;
    res = max(res, sum);
  }
  cout << res << endl;
  return 0;
}
```

区间能否分组

跟上一道题一样

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

using namespace std;

const int maxn = 4e5 + 5;
struct node {
  int pos, value;
} w[maxn];
bool cmp(node p, node q) {
  if (p.pos != q.pos) return p.pos < q.pos;
  return p.value < q.value;
}

int main()
{
  int n; cin >> n;
  for (int i = 1, id = 0; i <= n; ++i) {
    int l, r; cin >> l >> r;
    w[++id] = {l, 1}, w[++id] = {r+1, -1};
  }
  sort(w+1, w+2*n+1, cmp);

  int res = 0, sum = 0;
  for (int i = 1; i <= 2*n; ++i) {
    sum += w[i].value;
    res = max(res, sum);
  }
  cout << (res<=2 ? "YES" : "NO") << endl;
  return 0;
}
```

区间覆盖

1. 把所有区间按照左端点排序
2. 设定一个界限值 limit, 这个界限值 limit 初始为 L, 每次遍历找到所有 左端点<=limit 的区间, 找他们最大的右端点
3. 当遇到 左端点>limit 的情况时, 把界限 limit 设为 刚才的最大右端点, 重复这个过程

注意 中间断开 和 最后不能超过R 的两种 -1 的情况

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

using namespace std;

const int maxn = 1e5 + 5;
const int inf = 1e9+10;
struct node {
    int l, r;
```

```
} w[maxn];
bool cmp(node p, node q) { return p.l < q.l; }

int main()
{
    int L, R; cin >> L >> R;
    int n; cin >> n;
    for (int i = 1; i <= n; ++i) cin >> w[i].l >> w[i].r;
    sort(w+1, w+n+1, cmp);

    int rmax = -inf, cnt = 0, limit = L;
    for (int i = 1; i <= n; i++) {
        if (w[i].l <= limit) {
            rmax = max(rmax, w[i].r);
            if (rmax >= R) { cout << cnt+1 << endl; return 0; }
        }
        else {
            if (rmax == -inf) { cout << -1 << endl; return 0; }
            limit = rmax, rmax = -inf, cnt++;
            i--;
        }
    }
    cout << -1 << endl;
    return 0;
}
```