市赛题目讲解

人员

左子毅、杨洋、赵清航、周子航、于珈浩、刘佳赫、王崇宇 到课

作业检查

左子毅 已完成

杨洋 未完成

赵清航 未完成

周子航 已完成

于迦浩 已完成

刘佳赫 已完成

王崇宇 已完成

作业

https://www.luogu.com.cn/contest/174101

8 道题目要求全部完成

课堂表现

课堂上总结了市赛的一些问题，比较突出的几个问题有：

1. 读题不全，题目某些信息读不到
2. 细节考虑不全，同样表现为测试不全面
3. 数组开小RE问题
4. 手跟不上脑子，代码打错问题

希望同学们后面要改进的：

1. 常用的代码，做过的题目要写熟
2. 先想好完整的思路，再去写代码
3. 做足够的测试

课堂内容

skip

```cpp
// 方法一
#include <bits/stdc++.h>

using namespace std;

const int N = 200 + 5, M = 453 + 5;
int w[N], f[M], rk[M];

int main() {
  int n, m; cin >> n >> m;
  for (int i = 1; i <= n; ++i) {
    cin >> w[i], ++f[w[i]];
  }

  if (n < 3) {
    for (int i = 1; i <= n; ++i) cout << 0 << " "; cout << endl;
    return 0;
  }

  int t = (n>=8 ? 9 : n), flag = 1;
  for (int i = M-1; i >= 0; --i) {
    if (!f[i]) continue;
    rk[i] = (i>m ? t*2 : t);
    t -= f[i]+flag; flag = 0;
    if (t < 0) t = 0;
  }

  for (int i = 1; i <= n; ++i) cout << rk[w[i]] << " ";
  cout << endl;
  return 0;
}
```

```cpp
// 方法二
#include <bits/stdc++.h>

using namespace std;

const int maxn = 200 + 5;
struct node {
  int value, id;
  bool operator < (const node& p) const { return value < p.value; }
} w[maxn];
int ans[maxn];

int main() {
  int n, m; cin >> n >> m;
  for (int i = 1; i <= n; ++i) {
    int x; cin >> x; w[i] = {x, i};
  }

  if (n < 3) {
    for (int i = 1; i <= n; ++i) cout << 0 << " "; cout << endl;
    return 0;
  }

  sort(w+1, w+n+1);
  reverse(w+1, w+n+1);

  for (int i = 1, t = (n>=8?9:n); i <= n; ++i) {
    if (i==1 || w[i].value!=w[i-1].value) {
      ans[w[i].id] = t;
      if (w[i].value > m) ans[w[i].id] *= 2;
    } else {
      ans[w[i].id] = ans[w[i-1].id];
    }
    --t;
    if (i == 1) --t;
    if (t < 0) t = 0;
  }

  for (int i = 1; i <= n; ++i) cout << ans[i] << " ";
  cout << endl;
  return 0;
}
```

meeting

1. 优先队列，维护当前每一组的r
2. 按照l排序
3. 处理到第i个时候
   如果w[i].l大于优先队列的最小值，更新
   否则，新开一组

```cpp
// 方法一
#include <bits/stdc++.h>

using namespace std;

const int maxn = 1e5 + 5;
struct node {
  int l, r;
  bool operator < (const node& p) const { return l < p.l; }
} w[maxn];

int main() {
  int n; cin >> n;
  for (int i = 1; i <= n; ++i) {
    int l, r; cin >> l >> r;
    w[i] = {l, r};
  }

  sort(w+1, w+n+1);
  priority_queue<int, vector<int>, greater<int>>q;
  for (int i = 1; i <= n; ++i) {
    if (!q.empty() && w[i].l>q.top()) q.pop();
    q.push(w[i].r);
  }
  cout << q.size() << endl;
  return 0;
}
```

```cpp
// 方法二
#include <bits/stdc++.h>

using namespace std;

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

int main() {
  int n; cin >> n;
  for (int i = 1; i <= n; ++i) {
    int l, r; cin >> l >> r;
    w[i] = {l, 1}, w[n+i] = {r+1, -1};
  }

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

museum

```
1. 先求原本的 123 数量 res
2. 1在最左边 增加量add1
   3在最右边 增加量add3
   2在中间(枚举) 增加量add2

->
1. 维护一个前缀1，维护一个后缀3
2. add1: 对于每个2，求后缀3
   add3: 对于每个2，求前缀1
   add2:

res + max({add1, add2, add3})
```

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

using namespace std;

typedef long long LL;
const int maxn = 1e5 + 5;
char str[maxn];
int p1[maxn], s3[maxn];

int main() {
  int n; cin >> n;
  cin >> (str+1);
  for (int i = 1; i <= n; ++i) p1[i] = p1[i-1] + (str[i]=='1');
  for (int i = n; i >= 1; --i) s3[i] = s3[i+1] + (str[i]=='3');

  LL res = 0, add1 = 0, add2 = 0, add3 = 0;
  for (int i = 1; i <= n; ++i) {
    if (str[i] == '2') {
      res += (LL)p1[i-1] * s3[i+1];
      add1 += s3[i+1], add3 += p1[i-1];
    }
    add2 = max(add2, (LL)p1[i-1]*s3[i]);
  }

  cout << res + max({add1, add2, add3}) << endl;
  return 0;
}
```

P1803 凌乱的yyy / 线段覆盖

贪心，按照右端点排序，能选则选 即可

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

using namespace std;

const int maxn = 1e6 + 5;
struct node {
  int l, r;
  bool operator < (const node& p) const { return r < p.r; }
} w[maxn];

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

  int res = 0;
  for (int i = 1, lst = -1; i <= n; ++i) {
    if (w[i].l >= lst) ++res, lst = w[i].r;
  }
  cout << res << endl;
  return 0;
}
```

T457949 第k大数

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

using namespace std;

int main()
{
    vector<int> a = {0, 5, 7, 9, 2, 10, 11, 6, 8, 4};
    for (int i = 1; i <= 9; ++i) {
        nth_element(a.begin()+1, a.begin()+i, a.end());
        for (int j = 1; j <= 9; j ++) cout << a[j] << " "; cout << endl;
        cout << i << " --- " << a[i] << endl;
        cout << "================================" << endl;
    }
    return 0;
}
```

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

using namespace std;

const int maxn = 1e7 + 5;
int w[maxn];

int main()
{
    int n, k; scanf("%d%d",&n,&k);
    for (int i = 1; i <= n; ++i) scanf("%d",&w[i]);
    nth_element(w+1, w+n-k+1, w+n+1);
    printf("%d\n", w[n-k+1]);
    return 0;
}
```

U221939 区间和

什么是离散化?

-> 把一些数量不多，但是值域比较广的数，映射到一些值域比较小的数上，从而可以使用下标进行操作

```cpp
// 离散化 模板
int yFind(int x) {
    return lower_bound(vec.begin(), vec.end(), x) - vec.begin() + 1;
}
sort(vec.begin(), vec.end())
vec.erase(unique(vec.begin(), vec.end()), vec.end())
```

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

using namespace std;

vector<int> ys;
int yFind(int x) { return lower_bound(ys.begin(), ys.end(), x) - ys.begin(); }

const int maxn = 3e5 + 5;
struct node {
  int x, c;
} w[maxn];
struct node2 {
  int l, r;
} q[maxn];
int p[maxn];

int sum(int l, int r) { return p[r] - p[l-1]; }

int main()
{
  int n, m; cin >> n >> m;
  for (int i = 1; i <= n; ++i) {
    int x, c; cin >> x >> c;
    w[i] = {x, c};
    ys.push_back(x);
  }
  for (int i = 1; i <= m; ++i) {
    int l, r; cin >> l >> r;
    q[i] = {l, r};
    ys.push_back(l), ys.push_back(r);
  }

  sort(ys.begin(), ys.end());
  ys.erase(unique(ys.begin(), ys.end()), ys.end());

  for (int i = 1; i <= n; ++i) {
    int x = w[i].x, c = w[i].c;
    int pos = yFind(x) + 1;
    p[pos] += c;
  }

  for (int i = 1; i < maxn; ++i) p[i] += p[i-1];

  for (int i = 1; i <= m; ++i) {
    int l = yFind(q[i].l)+1, r = yFind(q[i].r)+1;
    cout << sum(l, r) << endl;
  }
  return 0;
}
```