

_04贪心算法(2)

贪心：线段覆盖问题

问题：在数轴上有 n 条线段 $[l_i, r_i]$ ，选择最多的不相交线段

贪心策略：按右端点从小到大排序

正确性证明：

- 选择结束时间最早的线段，为后续线段留下更多空间
- 这是最优选择，因为任何其他选择都不会比这个更好

```
struct Segment {
    int l, r;
    bool operator<(const Segment& other) const {
        return r < other.r; // 按右端点排序
    }
} seg[N];

int main() {
    int n;
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> seg[i].l >> seg[i].r;
    }

    sort(seg, seg + n);

    int cnt = 0, last_r = -1e9;
    for (int i = 0; i < n; i++) {
        if (seg[i].l >= last_r) { // 不相交
            cnt++;
            last_r = seg[i].r;
        }
    }

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

贪心：区间点覆盖问题

问题：选择最少的点，使得每个区间至少包含一个点

贪心策略：按右端点从小到大排序，在区间右端点放置点

正确性证明：

- 在区间右端点放点可以覆盖尽可能多的后续区间
- 这是最优选择，因为任何其他位置都不会比右端点更好

```
struct Interval {
    int l, r;
    bool operator<(const Interval& other) const {
        return r < other.r;
    }
} intervals[N];

int main() {
    int n;
    cin >> n;
    for (int i = 0; i < n; i++)
        cin >> intervals[i].l >> intervals[i].r;

    sort(intervals, intervals + n);

    int cnt = 0, last = -1e9;
    for (int i = 0; i < n; i++)
        if (intervals[i].l > last) {
            cnt++;
            last = intervals[i].r;
        }

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

贪心法示例：区间完全覆盖问题

问题：用最少的线段覆盖指定区间 $[s, t]$

贪心策略：

1. 按左端点从小到大排序
2. 每次选择能覆盖当前起点且右端点最大的线段

```
struct Segment {  
    int l, r;  
    bool operator<(const Segment& other) const {  
        return l < other.l; // 按左端点排序  
    }  
} seg[N];
```

```
int main() {
    int s, t, n;
    cin >> s >> t >> n;
    for (int i = 0; i < n; i++)
        cin >> seg[i].l >> seg[i].r;

    sort(seg, seg + n);

    int cnt = 0, current = s, i = 0;
    bool success = false;

    while (current < t) {
        int max_r = current;
        // 找能覆盖current且右端点最大的线段
        while (i < n && seg[i].l <= current) {
            max_r = max(max_r, seg[i].r);
            i++;
        }

        if (max_r == current) break; // 无法继续覆盖

        current = max_r, cnt++;

        if (current >= t) {
            success = true;
            break;
        }
    }

    if (success) cout << cnt << endl;
    else cout << -1 << endl;
}
```


贪心法总结

核心思想： 每一步都采取当前状态下最优的选择

适用条件：

- 最优子结构
- 贪心选择性质

常见题型：

- 区间问题（覆盖、选点、分组）
- 背包问题的特殊情况（部分背包）
- 调度问题
- 哈夫曼编码

解题步骤：

1. 分析问题特性
2. 设计贪心策略
3. 证明策略正确性（不用详细证明，简单证即可），常见的数学归纳法，反证法。
4. 代码实现