基于Monad的区间选择贪心算法正确性证明说明文档

〇、完成进度概述

完全完成第二档、第三档难度，第四档难度在 AI 辅助下完成。

一、算法概述

完成对一个基于贪心策略的区间选择算法进行形式化证明，该算法输入**右端点递增**的闭区间序列，输出一组两两不相交的区间，需证明：

1. 算法输出的区间数量是所有可行解中的最大值（第二档）；
2. 算法输出的具体区间集合是数量最大的可行解（第三档）；
3. 该区间集合是所有最优解中**区间编号字典序最小**的方案（第四档）。

算法核心逻辑为：从指定左端点 `leftmost` 开始遍历区间，若当前区间左端点大于 `leftmost` 则选择该区间（更新 `leftmost` 为当前区间右端点），否则跳过，最终返回选中的区间数量和集合。

二、核心定义与基础性质

2.1 基础概念定义

定义名称	数学含义	关键说明
`interval`	`Z * Z`	闭区间类型， `left(it)` 取左端点， `right(it)` 取右端点
`right_increasing l`	区间序列 `l` 的右端点非递减	算法输入的前提条件，保证贪心策略有效性
`non_overlap_from leftmost ans`	序列 `ans` 中区间两两不交且均严格在 `leftmost` 右侧	可行解的核心约束（严格不交：左端点> 前一区间右端点）
`valid_solution l leftmost ans`	`ans` 是 `l` 的子序列 ∧ `non_overlap_from leftmost ans`	合法解的完整定义（子序列保证区间来自原序列）
`greedy_list l leftmost`	贪心算法的纯函数实现	遍历 `l` ，按规则选择区间并返回选中集合
`greedy_size l leftmost`	`Zlength (greedy_list l leftmost)`	贪心解的区间数量
`lex_lt l1 l2`	整数序列 `l1` 字典序小于 `l2`	用于比较区间编号的优先级
`greedy_indices l leftmost`	贪心选中区间在原序列中的索引序列	索引从0开始，用于字典序证明

2.2 关键基础引理

（1）子序列性质

引理名称	核心含义	关键作用
`is_subsequence_cons_r`	原序列头部追加元素，不改变其子序列关系	证明跳过原序列首元素时，子序列关系仍成立
`is_subsequence_nil_inv`	空序列的子序列只能是空序列	归纳法基例中证明解的唯一性
`is_subsequence_drop_head`	子序列去掉首元素，仍是原序列的子序列	拆分最优解首元素后，后续子序列仍合法
`is_subsequence_tail`	子序列与原序列同时去掉首元素，剩余保持子序列关系	归约最优解后续子序列的合法性证明
`is_subsequence_cons_skip`	子序列首元素与原序列首元素不同时，可跳过原序列首元素	证明首区间不可选时，最优解必不包含该区间
`is_subsequence_in`	子序列元素必出现在原序列中	结合右端点有序性质推导区间右端点大小关系

（2）右端点有序性质

引理名称	核心含义	关键作用
`right_increasing_tail`	右端点递增序列的尾部仍保持右端点递增	归纳法中归约到子序列的有序性证明

引理名称	核心含义	关键作用
right_increasing_head_le	右端点递增序列中，首区间右端点≤ 后续所有区间右端点	推导最优解首区间右端点≥贪心选择区间的右端点
right_increasing_head_le_subseq	子序列中任意区间的右端点≥原序列首区间右端点	核心用于证明贪心选择的区间右端点更小， 保留更多选择空间

（3）不相交性质

引理名称	核心含义	关键作用
non_overlap_from_weaken_leftmost	若 leftmost' ≤ leftmost 且 ans 在 leftmost 右侧不交， 则在 leftmost' 右侧也不交	证明最优解后续子序列在贪心选择区间的右端点右侧
valid_solution_skip_head	原序列首区间与 leftmost 相交时， 合法解必不包含该区间	归纳法中处理首区间不可选的情况

三、贪心算法正确性证明（第二档）

3.1 证明思路

核心目标：证明 greedy_size l leftmost 是所有合法解的最大长度，即
∀ ans, valid_solution l leftmost ans → Zlength ans ≤ greedy_size l leftmost

采用**数学归纳法**，按原序列 l 的结构归纳：

1. **基例**： l = nil 时，唯一合法解是空序列，长度为0，贪心解长度也为0，不等式成立；
2. **归纳步**：假设 l = (l1, r1) :: rest ，分两种情况：
   - 若 l1 ≤ leftmost ： 首区间不可选，合法解必是 rest 的合法解，由归纳假设得证；
   - 若 l1 > leftmost ： 贪心选择首区间，此时任意合法解 ans 若非空，其首区间右端点≥ r1 ，后续子序列长度≤贪心解后续长度，整体长度≤贪心解长度。

3.2 核心定理

```
Lemma greedy_list_optimal_size:
  forall l leftmost,
    right_increasing l ->
    forall ans,
      valid_solution l leftmost ans ->
      Zlength ans <= greedy_size l leftmost.
```

3.3 支撑引理（第二档专属）

引理名称	核心含义	关键作用
greedy_iter_list_prefix	贪心迭代的结果等于初始空列表的结果追加到输入ans后	证明 greedy_list 的递归展开式 （ greedy_list_cons ）
greedy_list_cons	贪心列表的递归展开式：首区间可选则加入，否则递归处理剩余	归纳法中拆分贪心解的结构
greedy_list_subsequence	贪心解是原序列的子序列	证明贪心解本身是合法解的前提
greedy_list_non_overlap	贪心解满足严格不交约束	证明贪心解本身是合法解的前提
greedy_list_valid	贪心解是合法解（结合上述两个引理）	第三档证明的核心前提
greedy_size_cons	贪心长度的递归展开式：首区间可选则长度+1， 否则等于剩余长度	归纳法中计算贪心解长度
greedy_size_nonneg	贪心长度非负	字典序证明中处理长度相等的约束

3.4 关键步骤

- 拆分合法解 ans 为 (l0, r0) :: ans' ，利用 right_increasing_head_le_subseq 得 r1 ≤ r0 ；
- 由 non_overlap_from_weaken_leftmost 得 ans' 在 r1 右侧合法；

- 结合归纳假设 `Zlength ans' ≤ greedy_size rest r1`，推导出 `Zlength ans ≤ greedy_size l leftmost`。

四、最优方案正确性证明（第三档）

4.1 证明思路

在第二档基础上，需额外证明：

1. 贪心解本身是合法解（`valid_solution l leftmost (greedy_list l leftmost)`）；
2. 贪心解的长度等于 `greedy_size l leftmost`；
3. Monad实现的算法输出与纯函数贪心解等价。

4.2 核心定理

```
Theorem max_interval_opt_solution:
  forall l leftmost,
    right_increasing l ->
    Hoare (max_interval l leftmost)
          (fun '(size, ans) =>
             valid_solution l leftmost ans /\
             size = Zlength ans /\
             (forall ans',
               valid_solution l leftmost ans' ->
               Zlength ans' <= size)).
```

4.3 支撑引理（第三档专属）

（1）Monad与纯函数等价性相关

引理名称	核心含义	关键作用
`greedy_step_size_inv`	贪心状态单步更新后，size仍等于ans的长度	保证状态迭代中size的正确性
`greedy_iter_state_snoc`	贪心状态迭代在列表末尾追加元素等于单步更新	证明状态迭代的递归正确性
`greedy_iter_state_ans`	状态迭代的ans结果与纯函数 `greedy_iter_list` 一致	连接状态迭代与纯函数贪心解
`greedy_iter_state_size`	状态迭代中size始终等于ans的长度	保证最终输出size的正确性
`greedy_state_result`	初始状态的贪心迭代结果等于纯函数贪心解	核心等价性引理
`Hoare_max_interval_body`	Monad单步选择与 `greedy_step` 等价	连接Monad单步操作与纯函数
`Hoare_list_iter_state`	Monad列表迭代与 `greedy_iter_state` 等价	连接Monad迭代与纯函数状态迭代
`Hoare_max_interval_state`	Monad算法输出与纯函数贪心解等价	第三档证明的核心等价性定理

（2）合法性相关

引理名称	核心含义	关键作用
`max_interval_opt_size`	Monad算法输出的size是所有合法解的最大值	第二档结论的Monad版本

五、字典序最小性证明（第四档）

5.1 证明思路

核心目标：贪心解的索引序列 `greedy_indices l leftmost` 是所有最优解索引序列中字典序最小的，即
`∀ ans' il', (valid_solution l leftmost ans' ∧ sincr il' ∧ is_indexed_elements l il' ans' ∧ Zlength ans' = greedy_size l leftmost) → lex_le (`

关键概念：

- `sincr il`：索引序列严格递增（保证区间按原序列顺序选择）；
- `is_indexed_elements l il ans`： `ans` 是 `l` 中索引为 `il` 的元素组成的序列。

5.2 核心定理

```
Lemma greedy_indices_lex_minimal:
  forall l leftmost ans' il',
    right_increasing l ->
    valid_solution l leftmost ans' ->
    sincr il' ->
    is_indexed_elements l il' ans' ->
    Zlength ans' = greedy_size l leftmost ->
    lex_le (greedy_indices l leftmost) il'.
```

5.3 支撑引理（第四档专属）

（1）字典序基础性质

引理名称	核心含义	关键作用
lex_le_refl	字典序具有自反性	基例中证明空序列的字典序最小
lex_lt_cons_lt	首元素更小则字典序更小	证明贪心索引序列首元素0小于任意正整数
lex_lt_cons_eq	首元素相等时，后续序列字典序更小则整体更小	归约到后续索引序列的字典序证明
lex_le_cons_eq	首元素相等时，后续序列字典序更小/相等则整体更小	字典序最小性的递归证明
lex_lt_map_add1	索引序列全+1后字典序关系保持	处理首区间不可选时的索引偏移
lex_le_map_add1	索引序列全+1后字典序≤关系保持	索引偏移后的字典序证明

（2）贪心索引序列性质

引理名称	核心含义	关键作用
greedy_indices_cons	贪心索引序列的递归展开式	归纳法中拆分索引序列结构
greedy_indices_nonneg	贪心索引序列所有索引非负	保证索引的合法性
greedy_indices_sincr	贪心索引序列严格递增	证明贪心索引序列是合法的索引序列
greedy_indices_indexed	贪心索引序列对应原序列中的贪心解	连接索引序列与贪心解

（3）索引偏移性质

引理名称	核心含义	关键作用
is_indexed_elements_shift_up	原序列头部追加元素后，索引+1仍指向同一元素	处理首区间不可选时的索引偏移
is_indexed_elements_shift_down	原序列头部元素跳过且索引全为正，索引-1仍指向同一元素	归约到剩余序列的索引证明
Forall_add1_pos	非负索引+1后全为正	证明索引偏移后的正性
indices_positive_when_skip	首区间不可选时，最优解索引序列全为正	应用 shift_down 的前提条件
sincr_tail	严格递增序列的尾部仍严格递增	拆分索引序列后的单调性证明
map_sub_add1_id	索引先-1后+1等于原序列	索引偏移后的还原

5.4 关键步骤

1. 若首区间不可选（ `l1 ≤ leftmost` ），则最优解索引序列无0，归约到 `rest` 的索引序列（整体+1）；
2. 若首区间可选（ `l1 > leftmost` ）：
   - 若最优解索引序列首元素为0：与贪心索引序列首元素相同，归约到后续索引序列；
   - 若最优解索引序列首元素>0：贪心索引序列字典序更小（首元素0 < 任意正整数）。

5.5 算法层面的字典序保证

```
Theorem max_interval_opt_lexicographic:
  forall l leftmost,
    right_increasing l ->
    Hoare (max_interval l leftmost)
          (fun '(size, ans) =>
             exists il,
               sincr il /\
               is_indexed_elements l il ans /\
                size = Zlength ans /\
                (forall ans' il',
                  valid_solution l leftmost ans' ->
                  sincr il' ->
                  is_indexed_elements l il' ans' ->
                  Zlength ans' = size ->
                  lex_le il il')).
```

六、Monad程序与纯函数的等价性

6.1 核心映射关系

为连接算法的Monad实现与纯函数贪心解，定义：

- `greedy_step`： 单步贪心选择的状态转换函数；
- `greedy_iter_state`： 状态迭代的纯函数实现；
- `greedy_state`： 初始状态下的迭代结果。

6.2 等价性定理

```
Lemma Hoare_max_interval_state:
  forall l leftmost,
    Hoare (max_interval l leftmost)
          (fun '(size, ans) =>
             size = greedy_size l leftmost /\
             ans = greedy_list l leftmost).
```

6.3 证明核心

1. `Hoare_max_interval_body`： Monad的单步选择与 `greedy_step` 等价；
2. `Hoare_list_iter_state`： Monad的列表迭代与 `greedy_iter_state` 等价；
3. 结合上述两点，Monad算法输出与纯函数贪心解完全一致。

七、总结

7.1 核心结论

1. **最大数量正确性**：贪心算法输出的区间数量是所有可行解的最大值，依赖右端点递增的输入特性和归纳法证明；
2. **最优方案正确性**：贪心解本身是合法的最优解，且算法的Monad实现与纯函数贪心解等价；
3. **字典序最小性**：贪心解的区间编号索引序列是所有最优解中字典序最小的，保证了选择策略的“最左”特性。

7.2 关键技术点

1. 用 `is_subsequence` 刻画区间选择的合法性，保证解来自原序列；
2. 用 `non_overlap_from` 严格定义区间不交约束（严格大于保证闭区间不交）；
3. 用索引序列和字典序定义“最小”，将区间选择的字典序转化为整数序列的比较；
4. 基于Hoare逻辑连接Monad程序与纯函数，完成算法实现的正确性证明。

7.3 扩展说明

- 算法的输入前提（右端点递增）是贪心策略有效的核心，若输入无序需先排序；
- 闭区间的严格不交约束（ `left > 前一区间right` ）避免了端点重合的情况，若允许端点重合可调整为 `left >= 前一区间right` ；
- 字典序最小性保证了贪心策略的“贪心选择”特性：每次选择最早结束的区间，从而保留更多后续选择空间。

7.4 引理体系总结

证明档位	核心引理分类	核心作用
第二档（最大数量）	子序列性质+右端点有序+贪心列表基础	证明贪心解长度是所有合法解的最大值
第三档（最优方案）	Monad等价性+贪心解合法性	证明Monad算法输出的具体区间集合是最优解
第四档（字典序最小）	字典序性质+贪心索引序列性质+索引偏移	证明贪心解是所有最优解中字典序最小的