06 - Interval Scheduling

区间调度（Interval Scheduling）

- 一组请求 $\{1, 2, ..., n\}$
  - 每个请求都有一个开始时间 $s(i)$ 和一个结束时间 $f(i)$
  - 另一种观点是：每个请求是一个区间 $[s(i), f(i)]$
- 两个请求 i 和 j 是互相**兼容**的，如果他们的区间**不重叠**
- **目标**是输出一个使得兼容区间数量最大的调度

贪心算法（The Greedy Approach）

贪心算法：

1. 目标是得到一个全局解决方案
2. 解决方案是通过小的连续步骤建立起来的
3. 在每一步，都会根据某个标准选择当前最优的解决方案

贪心算法解决区间调度

- 从某个请求 i 的区间 $[s(i), f(i)]$ 开始
- 具体化选择方法：
  1. 选择最早开始的可用区间（显然不行）
  2. 选择最小的可用区间（显然也不行）
  3. 找到使“冲突”最少的区间（问题有些隐蔽，但是也不行）
  4. **更聪明的做法**：选择结束时间最早的区间 $[s(i), f(i)]$，即最小的 $f(i)$

```
IntervalScheduling([(s(i), f(i)) | i = 1 to n])
        Let R be the set of requests, let A be empty
        While R is not empty
            Choose a request i with the smallest f(i)
            Add i to A
            Delete all requests from R that are not compatible with request i

        Return the set A of accepted requests
```

1. 令 R 为请求集合，A 为空集合
2. R 非空时：
   1. 选择 R 中结束时间最早的请求 i，即最小的 $f(i)$
   2. 将 i 添加到集合 A 中
   3. 从 R 中删除与请求 i 不兼容的所有请求
3. 返回集合 A 作为最终选择的请求集合

（注意，上面的并不是完整算法）

算法正确性

- 问题1：贪心算法能否产生最优的调度?
- 问题2：贪心算法能否产生一个可行（或可接受）的调度?

引入符号：

- O 是最优调度
- A 是贪心算法产生的调度
- $i_1, \ldots, i_k$ 是加入 A 的请求的顺序，$|A| = k$
- $j_1, \ldots, j_m$ 是 O 中的请求集合，$|O| = m$
- 目标：证明 $m = k$

不失一般性，假设这些请求按照开始时间 $s(j_h)$ 的递增顺序排列，那么这些请求也是按照结束时间 $f(j_h)$ 的递增顺序排列的

声明：$f(i_1) \leq f(j_1)$，这是因为 i_1 是选择了结束时间最早的区间

引理:

- 对于所有 $r \leq k$ 的索引，$f(i_r) \leq f(j_r)$

  证明如下:
  $r = 1$ 时，由声明，$f(i_1) \leq f(j_1)$

  假设 $r-1$ 时假设成立，我们需要证明对于 r 也成立。即：假设 $f(i_{r-1}) \leq f(j_{r-1})$，证明 $f(i_r) \leq f(j_r)$

  我们已知 $f(j_{r-1}) \leq s(j_r)$，这是因为 j 是 O 的调度；并且已知 $f(i_{r-1}) \leq f(j_{r-1})$，这是由归纳假设得到的

  所以 $f(i_{r-1}) \leq s(j_r)$

  因此，当贪心算法在最坏的可能下才会选到 j_r，这意味着 $f(i_r) \leq f(j_r)$

算法正确性证明:
假设 $m > k$

对于 $r = k$，引理说明了 $f(i_k) \leq f(j_k)$
既然 $m > k$，O 中就存在一个额外的请求 j_{k+1}
我们有

$$s(j_{k+1}) > f(j_k) \geq f(i_k)$$

贪心算法会继续选择 j_{k+1}，即 $k > m$，这与我们的假设矛盾，因此命题得证

时间复杂度

算法如下：

- 将区间按照结束时间 $f(i)$ 的递增顺序排序
- 按照排序后的顺序，第一个区间的结束时间最早，因此被选中
- 对于排序后的任意连续区间 j，检查是否 $f(i) \leq s(j)$
  - 如果是，选择区间 j，并继续检查下一个区间
  - 如果否，跳过区间 j，并继续检查下一个区间

排序时间：$O(n\log n)$
选择与检查时间：$O(n)$
因此这个贪心算法的时间为 $O(n\log n)$