算法与复杂性　作业五

516021910528 - SHEN Jiamin

2020 年 3 月 16 日

1　0316

1. 求解递推关系

$$T(n) = n + \sum_{i=1}^{n-1} T(i)$$

其中 $T(1) = 1$

$$T(n+1) = (n+1) + \sum_{i=1}^{n} T(i)$$

$$T(n+1) - T(n) = \left[(n+1) + \sum_{i=1}^{n} T(i)\right] - \left[n + \sum_{i=1}^{n-1} T(i)\right] = 1 + T(n)$$

$$T(n+1) = 2T(n) + 1 \Rightarrow T(n+1) + 1 = 2T(n) + 2 \Rightarrow \frac{T(n+1)+1}{T(n)+1} = 2$$

所以当 $n > 2$ 时，

$$T(n) + 1 = T(1) \times 2^n \Rightarrow T(n) = 2^n - 1$$

又当 $n = 1$ 时，$T(1) = 1$ 恰好符合上式，所以对任意的 $n \in \mathbb{N}^*$，

$$T(n) = 2^n - 1$$

2. 在寻找一对一映射问题中，算法结束时集合 S 是否可能为空集？请证明你的结论。

给定一个集合 A 和一个从 A 到自身的映射 f，寻找一个元素个数最多的子集 $S \subseteq A$，满足 f 在 S 上是一一映射

不可能为空集。

证明 设集合 A 中有 n 个元素。

从集合中任取一个元素记作 x_1，记 $i \geq 2$ 时 $x_i = f(x_{i-1})$。

由于对任意的 $x \in A$，都有且仅有一个 $x' \in A, x' = f(x)$，所以序列 $\{x_i\}$ 的长度是无限的。但是集合 A 中元素个数是有限的，所以必然存在至少一对 $m < n$ 使得 $x_m = x_n$。

取一对使 $x_m, \ldots, x_{n-1}$ 各不相同的 m, n

- 若存在 $k : m < k < n, x_m = x_k$，则令 $n = k$
- 若存在 $p, q : m < p < q < n, x_p = x_q$，则令 $m = p, n = q$

由 $\{x_i\}$ 的定义可知，$x_m, \ldots, x_{n-1}$ 构成了一个循环置换

$$\begin{pmatrix} x_m & \cdots & x_{n-1} \end{pmatrix} = \begin{pmatrix} x_m & x_{m+1} & \cdots & x_{n-2} \\ x_{m+1} & \cdots & \cdots & x_{n-1} \end{pmatrix}$$

由置换的定义可知，f 在 $\{x_m, \ldots, x_{n-1}\}$ 上是一一映射，即 $\{x_m, \ldots, x_{n-1}\} \subseteq S$，所以 $S \neq \Phi$。

□

2 0319

3. 有 A、B、C 三个桩子，A 上放有 n 个不同大小的圆盘，按大小递减顺序自下而上。

   (a) 设计算法，用最少的移动步数将 A 上的圆盘都移到 C，每次只能移一个，且任一桩子上的圆盘都必须满足圆盘大小自下而上递减的顺序。

   (b) 给定数组 `p`，其元素取值只有 1，2，3 三种，代表了 n 个圆盘目前的位置，1 代表 A，2 代表 B，3 代表 C，`p[i]` 的值为第 i 个圆盘的位置。例如 `p=[3,3,2,1]`[1]表示共有 4 个圆盘，第一个和第二个都在 C，第三个在 B，第 4 个在 A。如果 `p` 代表了 (a) 中的最优移动过程中某一步的状态，则求出这是第几步。注意诸如 `p=[2,2]` 是不可能出现的，此时算法得到 -1。

(a) 解决汉诺塔问题首要的是注意到当只有两个圆盘存在时问题的解决是十分简单的。因此当解决有多个圆盘的汉诺塔问题时，可以将其中一部分圆盘看作是一个圆盘，另一部分圆盘看作是另一个圆盘，然后就可以应用两个圆盘的解法解决该问题，此时移动的每一步都是一个规模更小的汉诺塔问题。最后要注意到分组时必须将最大的圆盘单独放在一组，其他的圆盘单独放在一组，方可保证在移动的过程中任一桩子上的圆盘都满足大小自下而上递减。

算法 1 汉诺塔

输入: 三个栈 a, b, c，其中 a 中从栈顶到栈底为 0 到 $n-1$ 共 n 个元素，b, c 为空栈

```
1:  procedure MAIN
2:      HANOI(a, b, c, n)
3:  end procedure

4:  procedure HANOI(src, via, dst, n)
                          ▷ 解决以 via 为中介，从 src 的顶端移 n 个圆盘到 dst 上的汉诺塔问题
5:      if n = 1 then
6:          dst.push(src.pop())                ▷ 从 src 上取出最顶端的圆盘，放到 dst 上
7:      else
8:          HANOI(src, dst, via, n − 1)
9:          HANOI(src, via, dst, 1)
10:         HANOI(via, src, dst, n − 1)
11:     end if
12: end procedure
```

[1] `p[0]` 是最小的圆盘

(b) 由上述算法描述可知，解决 n 个圆盘的汉诺塔问题需要移动圆盘的次数 $T(n)$ 符合表达式

$$T(n) = \begin{cases} T(n-1) + T(1) + T(n-1) & , n \geq 2 \\ 1 & , n = 1 \end{cases}$$

可易解得 $T(n) = 2^n - 1$。

分析该算法可知，在解决规模为 n 的问题 $H(n)$ 时，首先要将 $n-1$ 个圆盘从 src 经过 $2^{n-1}-1$ 次操作移至 via 上；然后经 1 次操作将第 n 个圆盘从 src 移至 dst 上；最后经 $2^{n-1}-1$ 次操作将 $n-1$ 个圆盘从 via 经过 $2^{n-1}-1$ 次操作移至 dst 上。因此在这个子问题中，第 n 个圆盘（即该问题中最大的圆盘）只有在 src 和在 dst 上两种状态，不可能存在某一步操作使其出现在 via 上。

因此给出一个解决问题 $H(n)$ 的状态时，考察最大的圆盘的位置：

- 如果该圆盘在问题 $H(n)$ 的 src 上则记 b_n 为 0，此时正在解决的子问题 $H(n-1)$ 是要将 $n-1$ 个节点从 src 移至 via 上
- 如果该圆盘在问题 $H(n)$ 的 dst 上则记 b_n 为 1，此时正在解决的子问题 $H(n-1)$ 是要将 $n-1$ 个节点从 via 移至 dst 上

迭代地进行上述过程，则可得到序列 $\{b_i\}_{i\in[1,n]}$。当 b_i 从 0 变为 1 时，恰好是第 i 个盘子从其 src 移到 dst 上的结果，此时问题 $H(i)$ 恰好执行了 2^i 步。因此 $N = \sum_{i=1}^{n} b_i \cdot 2^{i-1}$ 可表示到此状态时已经经过的步数。

算法 2 检查汉诺塔状态

输入: 数组 $p[0..n-1]$ 表示用上述算法解决 n 个圆盘的汉诺塔问题的一个中间状态

输出: 到达这一状态经过的步数 N，如果该状态非法则为 -1

```
1:  src ← 1, via ← 2, dst ← 3
2:  N ← 0
3:  for i ← n − 1 downto 0 do
4:      if p[i] = src then
5:          N ← 2N + 0
6:          Swap(dst, via)
7:      else if p[i] = dst then
8:          N ← 2N + 1
9:          Swap(src, via)
10:     else
11:         return −1
12:     end if
13: end for
14: return N
```

4. 课上的最大连续子序列是指和最大，如果要求是积最大呢?

给定实数序列 $x_1, x_2, \ldots, x_n$，寻找连续子序列 $x_i, x_{i+1}, \ldots, x_j$ 使得其数值之积在所有连续子序列数值之积中为最大。

设空序列的积为 1，只需求出最大值即可，不需要知道是哪个序列。

考虑乘积，在一个连续子序列后面加上一个元素时，既要考虑乘积绝对值的变化，也要考虑符号的变化。若符号在某一次运算后为负数，且后面还有一个负的元素，则后缀的乘积仍有机会变正并大于现有的最大值。考虑到这一因素，除了 $MaxSuffix$ 以外，还需要一个 $MinSuffix$ 以记录出现负元素时后缀乘积的情况，并按如下规则更新：

- $MaxSuffix = \max\{MaxSuffix(i) \cdot x_{i+1}, MinSuffix(i) \cdot x_{i+1}, 1\}$
- 当 $MinSuffix(i) < 0$ 时，$MinSuffix(i+1) = MinSuffix(i) \cdot x_{i+1}$
- 当 $MinSuffix > 0$ 时，$MinSuffix$ 应当总等于 $MaxSuffix$

算法 3 最大乘积连续子序列

输入: 序列 $\{x_i\}_{1 \le i \le n}$

输出: 乘积最大的连续子序列的乘积 $MaxProd$

$MaxProd \leftarrow 1$ ▷ 最大子序列的乘积
$MaxSuffix \leftarrow 1, MinSuffix \leftarrow 1$ ▷ 最大正后缀乘积、最小负后缀乘积
for $i \leftarrow 1$ to n **do**
    $MaxSuffix \leftarrow MaxSuffix \cdot x_i$
    $MinSuffix \leftarrow MinSuffix \cdot x_i$
    if $MaxSuffix < 1$ **then**
        $MaxSuffix \leftarrow 1$
    end if
    if $MaxSuffix < MinSuffix$ **then**
        $MaxSuffix \leftarrow MinSuffix$
    end if
    if $MinSuffix \ge 0$ **then**
        $MinSuffix \leftarrow MaxSuffix$
    end if
    if $MaxSuffix > MaxProd$ **then**
        $MaxProd \leftarrow MaxSuffix$
    end if
end for

5. 背包问题中如果各种大小的物品的数量不限，那么如何知道背包是否能够恰好放满

给定一个整数 K 和 n 种不同大小的物品，第 i 中物品的大小为整数 s_i，每种物品的数量不限。寻找一个物品的组合，它们的大小之和正好为 K，或者确定不存在这样的组合。

记 $s_0 = 0$，$P(K, S = \{s_1, s_2, \ldots, s_n, s_0\})$ 表示从 S 中选取物品是否能恰好装满一个大小为 K 的背包的问题。

则有

$$P(K, \{s_1, s_2, \ldots, s_n, s_0\}) = \bigvee_{\substack{K - ms_n \geq 0, \\ m \geq 0}} P(K - ms_n, \{s_1, \ldots, s_{n-1}, s_0\})$$

且

$$\forall S \supset \{s_0\}, P(0, S) = \mathsf{true} \qquad \forall K > 0, P(K, \{s_0\}) = \mathsf{false}$$

算法 4 物品数量不限的背包问题

输入: $K, S[1..n]$

输出: 解是否存在

```
1:  P[0..n][0..K] ← {false}
2:  P[0][0] ← true
3:  for k = 0 to K do
4:      for i = 1 to n do
5:          if P[i − 1][k] = true then
6:              P[i][k] ← true
7:          else
8:              r ← [k mod s[i]]
9:              while r < k do
10:                 if P[i][r] = true then
11:                     P[i][k] ← true
12:                     break
13:                 end if
14:                 r ← r + k
15:             end while
16:         end if
17:     end for
18: end for
```