数算作业

林涛
1600012773

2017 年 10 月 28 日

1 计算循环次数

(1) 设次数为F(n)

$$F(1) = F(2) = 0$$

$n \geq 3$时，

$$\begin{aligned} F(n) &= \sum_{i=1}^{n-2} n - i + 1 \\ &= \sum_{k=3}^{n} k \\ &= \frac{(n+3)(n-2)}{2} \end{aligned}$$

(2) 先考虑i=n的情况，设次数为F(n)

$$
\begin{aligned}
F(1) &= 0 \\
F(n) &= 1 + F(\lfloor \frac{n}{2} \rfloor) \\
&= 1 + 1 + F(\lfloor \frac{\lfloor \frac{n}{2} \rfloor}{2} \rfloor) \\
&= 2 + F(\lfloor \frac{n}{4} \rfloor) \\
&= 3 + F(\lfloor \frac{n}{8} \rfloor) \\
&= ... \\
&= k + F(\lfloor \frac{n}{2^k} \rfloor)
\end{aligned}
$$

令$\lfloor \frac{n}{2^k} \rfloor = 1$解得：$k = \lfloor log_2 n \rfloor$ 所以$F(n) = \lfloor log_2 n \rfloor$
所以i=n*n的循环次数为：$F(n^2) = \lfloor 2log_2 n \rfloor (n \geq 1)$

(3) 第k次循环后：$i = k$, $s = 5k(k+1) \geq i$，只需考虑结束条件：$s \geq n$，
解得：$k \geq \frac{\sqrt{20n+25}-5}{10}$
因此次数F(n)为：

$$
F(n) = \begin{cases} \lceil \dfrac{\sqrt{20n+25}-5}{10} \rceil, & n \geq 1 \\ 1, & n = 0 \end{cases}
$$

(4) 执行k次 $\iff$ $2^{k-1} \leq n$ 且 $2^k > n$ $\iff$ $k - 1 \leq log_2 n < k$ $\iff$
$k = \lfloor log_2 n \rfloor + 1$

2 证明

1) $\because \lim_{n \to +\infty} \frac{b^n}{a^n} = \lim_{n \to +\infty} (\frac{b}{a})^n = 0$
$\therefore$ 令c = 1, $\exists$ N, s.t. $\forall\ n > N, b^n < ca^n$ $\therefore b^n = O(a^n)$
但是$\lim_{n \to +\infty} \frac{a^n}{b^n} = \infty$,$\therefore \forall c > 0, \exists n, s.t. a^n > cb^n$ $\therefore a^n \neq O(b^n)$

2) $\because \lim_{n \to +\infty} \frac{\sqrt[n]{n!}}{n} = \frac{1}{e} < 1$
$\therefore \lim_{n \to +\infty} \frac{n!}{n^n} = 0$
$\therefore n! = O(n^n)$

同时$\lim_{n\to+\infty}\frac{n^n}{n!}=+\infty$
$\therefore n^n \neq O(n!)$

3)

$$\because \lim_{n\to+\infty}\frac{\sqrt[n]{n!}}{n}=\frac{1}{e}$$
$$\text{取对数，}\lim_{n\to+\infty}\frac{1}{n}ln(n!)-ln(n)=-1$$
$$\text{除以ln(n)}\lim_{n\to+\infty}\frac{ln(n!)}{nln(n)}-1=0$$
$$\therefore \lim_{n\to+\infty}\frac{ln(n!)}{ln(n^n)}=1$$
$$\therefore ln(n!)=\Theta(ln(n^n))$$
$$\therefore log(n!)=\Theta(log(n^n)$$

4) 已知：$T(N)=T(\frac{N}{2})+1$，证明：$T(N)=O(logN)$

$$\begin{aligned}T(1)&=c,\\ T(N)&=T(N/2)+1\\ &=T(N/4)+1+1\\ &=...\\ &=T(N/2^k)+k\\ &=T(1)+log_2N\\ &=O(logN)\end{aligned}$$

3 元素右移k位的算法

描述：
1.按k将数组a分成m+1段，前m段每段长度为k，最后一段长度为r = n mod k，编号为0,...,m-1,m
2.交换第0段与第1段的位置，保持段中顺序不变。交换新的第0段与第2段，...，交换新的第0段与第m-1段。使第0段到第m-2段整体后移k位，第m-1段

移到第0段的位置。
3.交换第m段与a[0..r-1]，使得原来第m-1段的前r个元素到位。
4.递归地，第0段内部右移k-r位。

Algorithm 1 rightShift(a, n, k)

Require: $a[0..n-1], n \geq 1, 0 \leq k \leq n$
Ensure: shift a to $a[n-k], a[n-k+1], ..., a[0], a[1], ..., a[n-k-1]$

```
if k == 0 then
    return
end if
r = nmodk, m = ndivm
for i = 1..m − 1 do
    for j = 0..k − 1 do
        swap(a[j], a[i ∗ k + j])
    end for
end for
if r == 0 then
    return
end if
for i = 0..r − 1 do
    swap(a[i], a[m ∗ k + i])
end for
rightShift(a, k, k − r)
return
```

交换次数：$F(n) = (m-1) * k + r + F(k)$, $F(1) = O(1)$ 归纳地，得：$F(n) = m * k - k + O(k) + n - m * k = O(n)$

4 判断单向链表L是否有环

描述：
1.令N=1
2.从L的头出发移动N-1步，设该点为p，若遇到尾部则返回False

3.从p出发，移动N步，每步都判断是否回到p，是则返回True，遇到尾部返回False
4.N+=1，转到1
时间复杂度：
设n为L中的点数，运算次数：$F(n) = \Sigma_{N=1}^{m} 2 * N = O(n^2)$, m为表头到环的距离与环的长度的较大值。当环的长度为$n/2$，表头距离环的距离也为n/2，则$F(n) = \Sigma_{N=1}^{\frac{n}{2}} 2 * N = \Omega(n^2)$ 因此时间复杂度为$O(n^2)$

5 求链表A和B的交

Algorithm 2 intersect(a,b)

```
p=a,q=b
newLink = NULL
while p!=NULL and q!=NULL do
  while p!=NULL and (q==NULL or p.value < q.value) do
    p = p.next
  end while
  while q!=NULL and (p==NULL or q.value < p.value) do
    q = q.next
  end while
  if p!=NULL and q!=NULL and p.value == q.value then
    //找到一个交点
    将p的信息复制到newNode
    newNode.next=NULL
    if newLink==NULL then
      将newLink设置为newNode
    else
      将newNode插到newLink尾部
    end if
    p = p.next, q = q.next
  end if
end while
return newLink
```

复杂度：恰好遍历a,b，所以时间复杂度为$\Theta(n)$。因为复制出了新的链表，所以空间复杂为$O(n)$。

6 用栈S1和S2模拟一个队列

思想：S1中保存队列的元素，元素在队列中的顺序是从头到尾，在S1中是从栈底到栈顶。 enqueue直接PUSH到S1的栈顶。 dequeue需将S1中的元素逆序转移到S2中，使队列头的元素处在S2的栈顶，从而POP。队列空等价于S1空。

Algorithm 3 enqueue(x)

```
PUSH(S1,x)
```

Algorithm 4 dequeue

```
//将S1的元素转移到S2，顺序变反
while not Sempty(S1) do
  POP(S1,x)
  PUSH(S2,x)
end while
//此时S1的栈底元素在S2的栈顶
POP(S2,x)
while not Sempty(S2) do
  POP(S2,x)
  PUSH(S1,x)
end while
//原来的S1的栈底元素已被删除，即出队
```

Algorithm 5 queue_empty

```
return Sempty(S1)
```

7

7.1 火车进站台问题

设F(n)为n辆火车进出站台后的顺序数。有递推式：

$$F(0) = F(1) = 1$$
$$F(n) = \sum_{k=1}^{k=n} F(k-1) * F(n-k)$$

证明如下：考虑序列1,2,...,n。首先1入栈，考虑1第几个出栈。
设1第$k(1 \leq k \leq n)$个出栈，1出栈时需保证栈中除1外没有其它元素，所以序列2,...,n进行了k-1次入栈和k-1次出栈操作。可用数学归纳法证明，这些出入栈操作只涉及2,...,k这k-1个元素，不涉及k+1及以后的元素。 2,...,k进栈、出栈的方案数为F(k-1)，然后1出栈，剩下n-k个元素再进栈、出栈，方案数为F(n-k)。
所以F(n) = F(0)*F(n-1) + ... + F(k-1)*F(n-k) + ... + F(n-1)*F(0)

7.2 关于栈的性质的证明

$p_1, ..., p_n$是合法的出栈序列 $\iff$ 不存在$i < j < k$,s.t. $p_j < p_k < p_i$。
首先有：$p_i < p_j \iff p_i$比p_j先进栈
先证明充分性：要证明p_1,p_2,...,p_n可以依次出栈
首先将比p_1小的元素$p_{i_1}, ..., p_{i_k}$入栈。然后将p_1入栈、出栈。得到序列p_1。考虑此时p_2可能的位置：

1. 不在栈中：继续将小于p_2的元素入栈，然后将p_2入栈、出栈，得到序列p_1,p_2。
2. 在栈顶：出栈。得到p_1,p_2。
3. 在栈中，不在栈顶：由入栈的顺序知，栈顶元素p_{i_k}满足：$p_2 < p_{i_k} < p_1$，但$1 < 2 < i_k$，矛盾。这种情况不会出现。

因此，总可以得到p_1, p_2。归纳地，若已得到序列$p_1, ..., p_i$，此时p_{i+1}还未入栈或在栈顶，若在栈中不在栈顶则出现矛盾，于是可以使p_{i+1}紧接着出栈。从而得到序列$p_1, ..., p_i, p_{i+1}$。因此p_1,p_2,...,p_n可以依次出栈。
再证明必要性：已知p_1,p_2,...,p_n是合法出栈序列。假设存在$i < j < k$，使

表 1: 转化为后缀式

符号	栈	后缀式
12		12
*	*	12
(	*(	12
8	*(	12 8
*	*(*	12 8
9	*(*	12 8 9
-	*(-	12 8 9 *
10	*(-	12 8 9 * 10
)	*	12 8 9 * 10 -
-	-	12 8 9 * 10 - *
11	-	12 8 9 * 10 - * 11
		12 8 9 * 10 - * 11 -

得$p_j < p_k < p_i$。
p_i比p_j和p_k先出栈，但后入栈。说明p_i出栈时p_j和p_k都在栈中。$p_j < p_k$说明p_j比p_k先入栈，从而后出栈，因此$j > k$，矛盾。
因此不存在这样的i,j,k。

8

8.1 把12*(8*9-10)-11转化为后缀式

见表(1)

8.2 利用栈计算上一题中得到的后缀表达式

见表(2)

表 2: 计算12 8 9 * 10 - * 11 -

符号	栈
12	12
8	12 8
9	12 8 9
*	12 72
10	12 72 10
-	12 62
*	744
11	744 11
-	733（答案）