Problem Set 1

Data Structures and Algorithms, Fall 2022

2022 年 9 月 17 日

1 P1

证明: 1: 第一次迭代后,A[1] 显然是一个有序的子数组; 2: 假设第 j 次迭代后,A[1...j] 是一个有序的子数组; 3: 现在证明第 j+1 次迭代后,A[1...j+1] 是一个有序的子数组, 在 while 循环中, 每次迭代都把 A[i] 往后挪一个位置, 所以总是保持 A[i...j+1] 的有序性, 当退出 while 循环时, 有两种情况,a. 此时的 i==0, 那么最后一行代码将把 A[1] 的值赋为 key, 此时 A[1...j+1] 有序,b. 此时 A[i]<=key, 最后一行代码使得 A[i+1]=key, 此时 A[i...j+1] 有序, 由假设可知 A[1...i] 有序, 所以 A[1...j+1] 有序; 4. 所以当最外层 for 循环结束时,A[1..N] 整个数组都是有序的.

2 P2

为了计算 GCD(x,y), 可以设计一个递归函数. 伪代码如下:

Algorithm 1 compute GCD of x,y

```
function GCD(x, y)
    if y == 0 then
        return x
    else
        return GCD(y, x%y)
    end if
end function
```

有限性: 根据除法的性质, $x\%y < y$, 所以每次递归第二个参数都会变小直至 0.

正确性: 根据公式 $GCD(x, y) = GCD(y, x\%y)$, 每次递归都保持 GCD(x, y) 的值不变，所以得出的是正确结果.

3 P3

(a) 反例: $f(n) = n!$

(b) 证明: 欲证 $o(f) = O(f) - \Theta(f)$, 只需证 $\forall g(n) \in o(f)$, 有 $g(n) \in O(f)$ 但是 $g(n) \notin \Theta(f)$; 假设 $g(n) \in o(f)$, 则 $\forall c > 0, \exists n_0 > 0, s.t. f(n) < cg(n)$, 当 $n \geq n_0$, 因此 $\exists c_0 > 0, \exists n_1 > 0, s.t. f(n) <= cg(n)$, 当 $n \geq n_1$, $g(n) \in O(f)$, 若 $g(n) \in \Theta(f)$, 则 $\exists c_1 > 0, c_2 > 0, n_0 > 0, s.t. c_1 g(n) \leq f(n) \leq c_2 g(n)$,forall $n \geq n_0$, 即 $\exists c > 0, n > 0, s.t. cg(n) \leq f(n)$, 与 $\forall c > 0, \exists n_0 > 0, s.t. f(n) < cg(n)$ 矛盾, 所以 $g(n) \notin \Theta(f)$.

综上,$o(f) = O(f) - \Theta(f)$.

4 P4

$1 = n^{\frac{1}{\lg n}} << \lg\lg^* n << \lg^* n = \lg^* \lg n << 2^{\lg^* n} << \sqrt{\lg\lg n} << \ln\ln n << \ln n << \lg n! << \lg^2 n << 2^{\sqrt{2\lg n}} << (\sqrt{3})^{\lg n} << 2^{\lg n} = n << n\lg n << n^2 = 4^{\lg n} << n^3 << n^{\lg\lg n} = \lg n^{\lg n} << (\lg n)! << \frac{9}{8}^n << 2^n << e^n << n2^n << n! << (n+1)! << 2^{2^n} << 2^{2^n+1}$

5 P5

假设两个栈分别为 s1,s2.

入队 (Enqueue) 操作每次把新元素压入 s1, 出队 (Dequeue) 操作每次把 s2 中的栈顶元素弹出, 若 s2 为空, 则需要先把 s1 中的所有元素先弹出, 放入 s2 中. 这样设计的话 s1 中的栈顶元素相当于队尾元素,s2 中的栈顶元素相当于队首元素.

伪代码:

Algorithm 2 Enqueue

```
s1.push(key)
```

Algorithm 3 Dequeue

```
if s2.empty() then
    while not s1.empty() do
        s2.push(s1.pop())
    end while
end if
s2.pop()
```

Enqueue: $\Theta(1)$

Dequeue: $\Theta(1)$ to $\Theta(n)$

6 P6

使用两个栈 s1,s2.

s1 中存的是进栈的元素,s2 中存的是对应的最小值. 每次入栈操作时,s1 存的是 x 的值,s2 存的是 min(x, 当前最小值), 出栈直接 s1.pop(), s2.pop() 即可, 求最小值 return s2.top().

伪代码:

Algorithm 4 进栈

```
if s1.empty() then
    s1.push(x), s2.push(x)
else
    s1.push(x), s2.push(min(x, s2.top()))
end if
```

Algorithm 5 出栈

```
s2.pop()
return s1.pop()
```

Algorithm 6 最小值

```
return s2.top()
```

空间复杂度: $\Theta(n)$