Sample Solution for Problem Set 1

Data Structures and Algorithms, Fall 2022

September 23, 2022

Contents

1 Problem 1

- Loop invariant: After the i-th loop, subarray $A[1...i]$ is sorted and $A[i]$ is the smallest element of A$[i...n]$.
- Proof:
  - Initialization: There is only one element, A[1], when $i = 1$.
  - Maintain: $A[i-1]$ is the smallest element of $A[i-1...n]$. After exchange, $A[i]$ is the smallest one of $A[i...n]$ and $A[1...i]$ is still sorted, when $i \leftarrow i + 1$.
  - Termination: $A[1...n]$ is sorted, when $i = n$.
- Correctness: Elements are exchanged only and $A[1...n]$ is sorted.

2 Problem 2

Without loss of generality, we assume $x \leq y$.

Algorithm

Input : two non-negative integers $x \leq y$
Output: $\gcd(x, y)$
if $x = 0$ **then**
  return y;
else
  return $\gcd(y \mod x, x)$;
end

Algorithm 1: $\gcd(x, y)$

Correctness

Note that x strictly decreases. Therefore, the process will terminate in finite time. By using $\gcd(x, y) = \gcd(y \mod x, x)$, we can prove the correctness of our algorithm by induction on x.

Challenge

In fact, this algorithm is efficient. It only uses $O(\log \max(x, y))$ times of modulo operation!

3 Problem 3

(a)

Counterexample: let $f(n) = 2^n$, when $c > 1$

$$\lim_{n \to \infty} \frac{f(cn)}{f(n)} = 2^{(c-1)n} \to \infty$$

(b)

Counterexample: let

$$g(n) = \begin{cases} f(n) & n \text{ is odd} \\ f(n) \cdot e^{-n} & n \text{ is even} \end{cases}$$

It's easy to prove that $g \in O(f), g \notin \Theta(f)$ but $g \notin o(f)$

4 Problem 4

$$
\begin{aligned}
1 = n^{1/\lg n} &\ll \\
\lg(\lg^* n) &\ll \\
\lg^* n = \lg^*(\lg n) &\ll \\
2^{\lg^* n} &\ll \\
\sqrt{\lg\lg n} &\ll \\
\ln\ln n &\ll \\
\ln n &\ll \\
\lg^2 n &\ll \\
2^{\sqrt{2\lg n}} &\ll \\
(\sqrt{3})^{\lg n} &\ll \\
n = 2^{\lg n} &\ll \\
n\lg n = \lg(n!) &\ll \\
n^2 = 4^{\lg n} &\ll \\
n^3 &\ll \\
(\lg n)! &\ll \\
(\lg n)^{\lg n} = n^{\lg\lg n} &\ll \\
(9/8)^n &\ll \\
2^n &\ll \\
n \cdot 2^n &\ll \\
e^n &\ll \\
n! &\ll \\
(n+1)! &\ll \\
2^{2^n} &\ll \\
2^{2^{n+1}} &
\end{aligned} \tag{1}
$$

5 Problem 5

Overview

Suppose there are two stacks S_1 and S_2.

- $ENQUEUE(x)$: directly push element x into stack S_1.
- $DEQUEUE()$: pop all the elements in S_1 and push them into S_2. Then, the element at the top of S_2 is the element we have to dequeue. Pop all the elements in S_1 and push them into S_2 (for further operations).

Pseudocode

```
function ENQUEUE(x):
    S1.push(x)

function DEQUEUE():
    while not S1.empty() : // for i from 1 to S1.size() is incorrect.
        S2.push(S1.pop())
    res = S2.pop()
    while not S2.empty() :
        S1.push(S2.pop())
    return res
```

5.1 Complexity

Obviously, each $ENQUEUE$ operation takes $\Theta(1)$ time.

Consider the $DEQUEUE$ operation. Suppose there are at most n elements in the stack simultaneously, then the $DEQUEUE$ operation takes $O(n)$ time.

6 Problem 6

Overview

We can maintain two stacks S_1 and S_2 which have equal size. S_1 stores the original elements. S_2 stores prefix maximum of S_1.

Pseudocode

```
function PUSH(x):
    S1.push(x)
    if S2.empty() or x < S2.top():
        S2.push(x);
    else:
        S2.push(S2.top())

function POP():
    S2.pop()
    return S1.pop()

function MIN():
    return S2.top()
```

Complexity

Obviously, each operation takes $\Theta(1)$ time.
Suppose there are at most n elements in the stack simultaneously, we use two stacks of size n. Therefore, space complexity is $\Theta(n)$ in the whole process.

Alternate Solution

There are many different implementations, here's another example. We can maintain a stack S_1 for original elements and a non-strictly decreasing stack $S2$ for possible minimums.

```
function PUSH(x):
    S1.push(x)
    if S2.empty() or x <= S2.top():   // x < S2.top() is incorrect!(why?)
        S2.push(x);

function POP():
    if S1.top() == S2.top():
        S2.pop()
    return S1.pop()

function MIN():
    return S2.top()
```

Remark

- Pseudocode should be precise and concise.
- Pay attention to the interface of data structure. Generally, $top(), pop(), empty()$ are common stack operations.