Homework 1

Lei Zhang

February 9, 2016

1 Exercises 1.1 - 5

Algorithm 1 Finding common elements in two sorted lists

INPUT:
An array $A_1[0..m-1]$ for m sorted numbers;
Another sorted array $A_2[0..n-1]$ with n numbers;
OUTPUT: The list of common elements in two sorted lists $C\{1, ...n\}$;

```
i ← 0
while (i < m)and(j < n) do
    if A1[i] == A2[i] then
        Add A[i] to the list C
        i ← i + 1
        j ← j + 1
    else if A1[i] > A2[i] then
        j ← j + 1
    else
        i ← i + 1
    end if
end while
return C
```

2 Exercises 1.1 - 6

a.

m = 31415, n = 14142
Step 1: $r = (31415 \mod 14142) = 3131, m \leftarrow 14142, n \leftarrow 3131$
Step 2: $r = (14142 \mod 3131) = 1618, m \leftarrow 3131, n \leftarrow 1618$
Step 3: $r = (3131 \mod 1618) = 1513, m \leftarrow 1618, n \leftarrow 1513$
Step 4: $r = (1618 \mod 1513) = 105, m \leftarrow 1513, n \leftarrow 105$
Step 5: $r = (1513 \mod 105) = 43, m \leftarrow 105, n \leftarrow 43$
Step 6: $r = (105 \mod 43) = 19, m \leftarrow 43, n \leftarrow 19$

Step 7: $r = (43 \mod 19) = 5, m \leftarrow 19, n \leftarrow 5$
Step 8: $r = (19 \mod 5) = 4, m \leftarrow 5, n \leftarrow 4$
Step 9: $r = (5 \mod 4) = 1, m \leftarrow 4, n \leftarrow 1$
Step 10: $r = (4 \mod 1) = 0, m \leftarrow 1, n \leftarrow 0$
Step 11: return 1

b.

Using consecutive integer checking algorithm, every time the value of t is decreased, it takes two divisions. There are 28284 divisions in total.

Using Euclid's algorithm, there are 10 divisions.

Euclid's algorithm is 28284/10 = 2828.4 times faster in terms of divisions.

3 Exercises 1.1 - 7

k, r are integers. Assume $r = (m \mod n)$. m can be expressed as $m = k * n + r$

Assume d is a common divisor of m and n: $d|m$, $d|n$. As $r = m - k * n$, $d|r$, d is also a common divisor of n and r.

Assume d is a common divisor of n and r: $d|n$, $d|r$. As $m = k * n + r$, $d|m$, d is also a common divisor of m and n.

Now $\{m, n\}$ and $\{n, r\}$ have the same common divisors. Thus, $gcd(m, n) = gcd(n, m \mod n)$

4 Exercises 1.2 - 4

Algorithm 2 finding real roots

Require: arbitrary real coefficients a, b, and c

1: $d \leftarrow b^2 - 4 * a * c$
2: **if** $d < 0$ **then return** $false$
3: **else if** $d == 0$ **then**
4: $x_1 \leftarrow (-b + sqrt(d))/(2 * a)$
5: **return** x_1
6: **else if** $d > 0$ **then**
7: $x_1 \leftarrow (-b + sqrt(d))/(2 * a)$
8: $x_2 \leftarrow (-b - sqrt(d))/(2 * a)$
9: **return** x_1, x_2
10: **end if**

5 Exercises 1.2 - 9

Algorithm 3 MinDistance

INPUT: Array A[0..n-1] of numbers
OUTPUT: Minimum distance between two of its elements

1: $A \leftarrow Merge(A)$
2: $dmin \leftarrow \infty$
3: **for** $i \leftarrow 0$ to $n-2$ **do**
4: **if** $|A[i] - A[i+1]| < dmin$ **then**
5: $dmin \leftarrow |A[i] - A[i+1]|$
6: **end if**
7: **end for**
8: **return** dim

The algorithm of merge sort describes as follow:

Algorithm 4 Merge Sort

INPUT: An array $A[0..n-1]$ of orderable elements
OUTPUT: Array $A[0..n-1]$ sorted in nondecreasing order

```
1: function MERGE(A, p, q, r)
2:     n_1 = q − p + 1
3:     n_2 = r − q
4:     Let L[1 ... n_1 + 1] and R[1 ... n_2 + 1] be new arrays
5:     for i ← 1 to n_1 do
6:         L[i] = A[p + i − 1]
7:     end for
8:     for j = 1 to n_2 do
9:         R[i] = A[q + j]
10:    end for
11:    L[n_1 + 1] = ∞
12:    R[n_2 + 1] = ∞
13:    i = 1
14:    j = 1
15:    for k = p to r do
16:        if L[i] < R[j] then
17:            A[k] = L[i]
18:            i = i + 1
19:        else if L[i] > R[j] then
20:            A[k] = R[j]
21:            j = j + 1
22:        else
23:            A[k] = −∞
24:            j = j + 1
25:        end if
26:    end for
27: end function
28:
29: function MERGESORT(A)
30:    if n == 1 then return A
31:    end if
32: end function
```

6 Exercises 1.3 - 1

a.

- outer loop 1: $Count[0]+=1, Count[2]+=1, Count[3]+=1, Count[0]+=1, Count[0]+=1$
- outer loop 2: $Count[2]+=1, Count[3]+=1, Count[1]+=1, Count[5]+=1$

- outer loop 3: $Count[3]+=1, Count[2]+=1, Count[2]+=1$
- outer loop 4: $Count[3]+=1, Count[3]+=1$
- outer loop 5: $Count[5]+=1$

For now, $Count[0]=3, Count[1]=1, Count[2]=4, Count[3]=5, Count[4]=0, Count[5]=2$

Thus, $S[3]=60, S[1]=35, S[4]=81, S[5]=98, S[0]=14, S[2]=47$

Result: $S=[14,35,47,60,81,98]$

b.

It's not stable.

c.

It's not an in-place algorithm as it uses extra storage space in the array Count.