CS3130 Homework 1

Phillip Janowski (pajmc2@mail.umsl.edu)

September 5, 2018

Question 1.

(a) Find $gcd(213486, 5423)$

$213486/5243 = 39R1989$
$213486 = 5423 * 39 + 1989$
$213486/1989 = 107R663$
$213486 = 1989 * 107 + 663$
$213486/663 = 332R0$

$\therefore gcd(213486, 5423) = 663$

(b) Estimate approximately how many times faster it will be to find $gcd(213486, 5423)$ with the help of the EuclidâĂŹs algorithm compared with the algorithm based on checking consecutive integers from minm, n down to gcd(m, n) (see the algorithm #2 from the handout). You may only count the number of modulus divisions of the largest integer by different divisors.

$min(213486, 5423) = 5423$
$m/t \neq 0$ where $m = 213486$ and $t = 5423$ $t - 1 = 5422$
$m - gcd(213486, 5423) = 4760$ steps
4760 steps /4 steps = 1190 times faster

Question 2.

2.1) Prove the formula on which Euclid's algorithm is based: $gcd(m, n) = gcd(n, m \bmod n)$ for every pair of positive integers m and n

2.2) 2.1-4
Given two array's A and B, length n, are added together and stored in an $(n+1)$ length array, C

```
carried = 0
for i = n-1 down to 0
     C[i+1] = {A[i] + B[i] + carried} mod 2
     carried = {A[i] + B[i] + carried} / 2
C[0] = carried
```

2.3) 2.2-2

(a)
```
for i = 0 to n - 1
     smallest = 0
     smallestElement = 0
     for 0 = i to n -1
          if A[j]< A[j+1]
               smallest = A[j]
               smallestElement = j
     A[smallestElement] = j
     A[i] = smallest
```

(b) The loop is invariant because at the beginning of each for loop, the sorted part of the array changes in size with each iteration and only have the i-1 smallest elements

(c) The loop only needs to run for n-1 times because the last element will already be sorted

(d) The run time of this loop is $\theta(n^2)$, because the inner loop runs each of its lines n times and the out loop runs its lines, including the inner loop, n times. $\therefore \theta(n^2)$

Question 3.

Modify the psuedo code on p.40 to count swaps for the bubble sort

```
i = 0
swap = 1
while swap != 0 and i < A.length
     swap = 0
          for j = A.length down to i + 1
               if A[j] < A[j-1]
                    exchange A[j] < A[j-1]
                    swap++
                    i++
```

Question 4.

Handwritten and Stapled to the packet

Question 5.

(a)
```
i = 0
     while i < 3n
          print"CS3130"
          i = i + 3
```

The answer is n times though I came up with 9n+3

(b)
```
for i = 0 to n
     for j = 0 to n
          print"CS3130"
```

$= \sum_{i=0}^{n} \sum_{j=0}^{n} 1 = \sum_{i=0}^{n} (n - 0 + 1 - 1) = n \sum_{i=0}^{n} 1 = n(n - 0 + 1 - 1)$

$$= n^2$$

(c)
```
for i = 0 to n
    for j = i to n
        print"CS3130"
```

$$= \sum_{i=0}^{n} \sum_{j=i}^{n} 1 = \sum_{i=0}^{n} (n - i + 1 - 1) = n \sum_{i=0}^{n} 1 - \sum_{i=0}^{n} i$$
$$= n(n - 0 + 1) - \frac{(0 + n)}{2}(n - 0 + 1) = n^2 + n - \frac{n^2 + n}{2}$$
$$= \frac{n^2 + n}{2}$$

Question 6.

2.1-3 Find v in Array A[ ]

```
i = 0
while i < A.length and v != A[i]
    i++
if A[i] != v
    i = null
```

This loop is invariant because at the beginning we have $i = 0$ at ininitialization. Maintenance, because it maintains while $i < length$ of A or $v \neq A[i]$ and is true before i and i + 1. And Termination, because it only terminates when v is found in A or when we run out of elements in A to check

Question 7.

The average performance of the linear search is $O(\frac{n}{2})$. The worst performance would happen from v being in the $A.length$th position of the array and the best would happen from v being in the 1st position of the array. In a set of completely random runs the average number of comparisons would eventually converge on $\frac{n}{2}$.