HOMEWORK 1: SOLUTIONS

Exercise 3

Consider the algorithm for sorting problem (Algorithm 1) that sorts an array by counting, for each of its elements, the number of smaller elements and then uses this information to put the element in its appropriate position in the sorted array. Apply this algorithm to sorting the list: 60, 35, 81, 98, 14, 47.

Algorithm 1 *ComparisonCountingSort(A[0 . . . n-1])*

```
1: {Sorts an array by comparison counting}
2: Input: Array A[0 . . . n − 1] of values.
3: Output: Array S[0 . . . n − 1] of values.
4: for i=0 to n-1 do
5:    Count[i] = 0
6: end for
7: for i=0 to n-2 do
8:    for j=i+1 to n-1 do
9:       if A[i]¡A[j] then
10:         Count[j] = Count[j] + 1
11:      else
12:         Count[i] = Count[i] + 1
13:      end if
14:   end for
15: end for
16: for i=0 to n-1 do
17:   S[Count[i]] = A[i]
18: end for
19: return S
```

1. a. Sorting 60, 35, 81, 98, 14, 47 by comparison counting will work as follows:

Array $A[0..5]$		60	35	81	98	14	47
Initially	$Count[]$	0	0	0	0	0	0
After pass $i = 0$	$Count[]$	3	0	1	1	0	0
After pass $i = 1$	$Count[]$		1	2	2	0	1
After pass $i = 2$	$Count[]$			4	3	0	1
After pass $i = 3$	$Count[]$				5	0	1
After pass $i = 4$	$Count[]$					0	2
Final state	$Count[]$	3	1	4	5	0	2
Array $S[0..5]$		14	35	47	60	81	98

Exercise 5

List the following functions according to their order of growth from the lowest to the highest:

- $(n-2)!$
- $5 \lg (n+100)^10$
- 2^{2^n}
- $0.001n^4 + 3n^3 + 1$
- $\ln^2 n$
- $\sqrt[3]{n}$
- 3^n

5. $(n-2)! \in \Theta((n-2)!)$, $5\lg(n+100)^{10} = 50\lg(n+100) \in \Theta(\log n)$, $2^{2n} = (2^2)^n \in \Theta(4^n)$, $0.001n^4 + 3n^3 + 1 \in \Theta(n^4)$, $\ln^2 n \in \Theta(\log^2 n)$, $\sqrt[3]{n} \in \Theta(n^{\frac{1}{3}})$, $3^n \in \Theta(3^n)$. The list of these functions ordered in increasing order of growth looks as follows:

$$5\lg(n+100)^{10}, \quad \ln^2 n, \quad \sqrt[3]{n}, \quad 0.001n^4 + 3n^3 + 1, \quad 3^n, \quad 2^{2n}, \quad (n-2)!$$

Exercise 6

Compute the following sums:

- $1 + 3 + 5 + 7 + \ldots + 999$
- $2 + 4 + 8 + 16 + \ldots + 1024$
- $\sum_{i=3}^{n+1} 1$
- $\sum_{i=3}^{n+1} i$
- $\sum_{i=0}^{n-1} i(i+1)$
- $\sum_{j=1}^{n} 3^{j+1}$
- $\sum_{i=1}^{n} \sum_{j=1}^{n} ij$
- $\sum_{i=1}^{n} \frac{1}{i(i+1)}$

1. a. $1+3+5+7+...+999 = \sum_{i=1}^{500}(2i\text{-}1) = \sum_{i=1}^{500} 2i - \sum_{i=1}^{500} 1 = 2\frac{500*501}{2}\text{-}500 = 250,000.$

(Or by using the formula for the sum of odd integers: $\sum_{i=1}^{500}(2i\text{-}1) = 500^2 = 250,000.$

Or by using the formula for the sum of the arithmetic progression with $a_1 = 1$, $a_n = 999$, and $n = 500$: $\frac{(a_1+a_n)n}{2} = \frac{(1+999)500}{2} = 250,000.$)

b. $2+4+8+16+...+1,024 = \sum_{i=1}^{10} 2^i = \sum_{i=0}^{10} 2^i - 1 = (2^{11}-1) - 1 = 2,046.$

(Or by using the formula for the sum of the geometric series with $a = 2$, $q = 2$, and $n = 9$: $a\frac{q^{n+1}-1}{q-1} = 2\frac{2^{10}-1}{2-1} = 2,046.$)

c. $\sum_{i=3}^{n+1} 1 = (n+1) - 3 + 1 = n - 1.$

d. $\sum_{i=3}^{n+1} i = \sum_{i=0}^{n+1} i - \sum_{i=0}^{2} i = \frac{(n+1)(n+2)}{2} - 3 = \frac{n^2+3n-4}{2}.$

e. $\sum_{i=0}^{n-1} i(i+1) = \sum_{i=0}^{n-1}(i^2+i) = \sum_{i=0}^{n-1} i^2 + \sum_{i=0}^{n-1} i = \frac{(n-1)n(2n-1)}{6} + \frac{(n-1)n}{2}$

$= \frac{(n^2-1)n}{3}.$

f. $\sum_{j=1}^{n} 3^{j+1} = 3\sum_{j=1}^{n} 3^j = 3[\sum_{j=0}^{n} 3^j - 1] = 3[\frac{3^{n+1}-1}{3-1} - 1] = \frac{3^{n+2}-9}{2}.$

g. $\sum_{i=1}^{n}\sum_{j=1}^{n} ij = \sum_{i=1}^{n} i \sum_{j=1}^{n} j = \sum_{i=1}^{n} i\frac{n(n+1)}{2} = \frac{n(n+1)}{2}\sum_{i=1}^{n} i = \frac{n(n+1)}{2}\frac{n(n+1)}{2}$

$= \frac{n^2(n+1)^2}{4}.$

h. $\sum_{i=1}^{n} 1/i(i+1) = \sum_{i=1}^{n}(\frac{1}{i} - \frac{1}{i+1})$

$= (\frac{1}{1} - \frac{1}{2}) + (\frac{1}{2} - \frac{1}{3}) + ... + (\frac{1}{n-1} - \frac{1}{n}) + (\frac{1}{n} - \frac{1}{n+1}) = 1 - \frac{1}{n+1} = \frac{n}{n+1}.$

(This is a special case of the so-called *telescoping series*—see Appendix A—$\sum_{i=l}^{u}(a_i - a_{i-1}) = a_u - a_{l-1}.$)

Exercise 7

Find the order of growth of the following sums. Use the $\Theta(g(n))$ with the simplest function $g(n)$ possible.

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

2. a. $\sum_{i=0}^{n-1}(i^2+1)^2 = \sum_{i=0}^{n-1}(i^4+2i^2+1) = \sum_{i=0}^{n-1} i^4 + 2\sum_{i=0}^{n-1} i^2 + \sum_{i=0}^{n-1} 1$

$\in \Theta(n^5)+\Theta(n^3)+\Theta(n) = \Theta(n^5)$ (or just $\sum_{i=0}^{n-1}(i^2+1)^2 \approx \sum_{i=0}^{n-1} i^4 \in \Theta(n^5)$).

b. $\sum_{i=2}^{n-1}\log_2 i^2 = \sum_{i=2}^{n-1} 2\log_2 i = 2\sum_{i=2}^{n-1}\log_2 i = 2\sum_{i=1}^{n}\log_2 i - 2\log_2 n$

$\in 2\Theta(n\log n) - \Theta(\log n) = \Theta(n\log n)$.

c. $\sum_{i=1}^{n}(i+1)2^{i-1} = \sum_{i=1}^{n} i2^{i-1} + \sum_{i=1}^{n} 2^{i-1} = \frac{1}{2}\sum_{i=1}^{n} i2^i + \sum_{j=0}^{n-1} 2^j$

$\in \Theta(n2^n)+\Theta(2^n) = \Theta(n2^n)$ (or $\sum_{i=1}^{n}(i+1)2^{i-1} \approx \frac{1}{2}\sum_{i=1}^{n} i2^i \in \Theta(n2^n)$).

d. $\sum_{i=0}^{n-1}\sum_{j=0}^{i-1}(i+j) = \sum_{i=0}^{n-1}[\sum_{j=0}^{i-1} i + \sum_{j=0}^{i-1} j] = \sum_{i=0}^{n-1}[i^2 + \frac{(i-1)i)}{2}] = \sum_{i=0}^{n-1}[\frac{3}{2}i^2 - \frac{1}{2}i]$

$= \frac{3}{2}\sum_{i=0}^{n-1} i^2 - \frac{1}{2}\sum_{i=0}^{n-1} i \in \Theta(n^3) - \Theta(n^2) = \Theta(n^3)$.

Exercise 8

Consider the following algorithm (Algorithm 2)

Algorithm 2 *Mystery(n)*

1: **Input**: A non-negative integer n.
2: $S = 0$
3: **for** *i=1 to n* **do**
4: $\quad S = S + i \times i$
5: **end for**
6: **return** S

1. What does this algorithm compute ?
2. What is the basic operation ?
3. How many times is the basic operation executed ?
4. What is the efficiency class of this algorithm ?
5. Suggest an improvement, or a better algorithm altogether,a nd indicate its efficiency class.

4. a. Computes $S(n) = \sum_{i=1}^{n} i^2$.

b. Multiplication (or, if multiplication and addition are assumed to take the same amount of time, either of the two).

c. $C(n) = \sum_{i=1}^{n} 1 = n$.

d. $C(n) = n \in \Theta(n)$. Since the number of bits $b = \lfloor \log_2 n \rfloor + 1 \approx \log_2 n$ and hence $n \approx 2^b$, $C(n) \approx 2^b \in \Theta(2^b)$.

e. Use the formula $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$ to compute the sum in $\Theta(1)$ time (which assumes that the time of arithmetic operations stay constant irrespective of the size of the operations' operands).

Exercise 9

Consider the following algorithm (Algorithm 3)

Algorithm 3 *Secret(A[0 . . . n-1])*

```
1: Input: Array A[0...n − 1] of n real numbers.
2: minval = A[0]
3: maxval = A[0]
4: for i=0 to n-1 do
5:    if A[i] < minval then
6:       minval = A[i]
7:    end if
8:    if A[i] > maxval then
9:       maxval = A[i]
10:   end if
11: end for
12: return maxval − minval
```

Answer questions (1)-(5) of exercise 8 about this algorithm.

5. a. Computes the range, i.e., the difference between the array's largest and smallest elements.

   b. An element comparison.

   c. $C(n) = \sum_{i=1}^{n-1} 2 = 2(n-1)$.

   d. $\Theta(n)$.

   e. An obvious improvement for some inputs (but not for the worst case) is to replace the two if-statements by the following one:

   if $A[i] < minval$ $minval \leftarrow A[i]$

else if $A[i] > maxval$ $maxval \leftarrow A[i]$.

Exercise 10

Consider the following algorithm (Algorithm 4)

Algorithm 4 *Enigma(A[0 . . . n-1, 0 . . . n-1])*

```
1: Input: Array A[0 . . . n − 1] of n real numbers.
2: for i=0 to n-2 do
3:    for j=i+1 to n-1 do
4:       if A[i, j] ≠ A[j, i] then
5:          return false
6:       end if
7:    end for
8: end for
9: return true
```

Answer questions (1)-(5) of exercise 8 about this algorithm.

6. a. The algorithm returns "true" if its input matrix is symmetric and "false" if it is not.

   b. Comparison of two matrix elements.

   c. $C_{worst}(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} [(n-1) - (i+1) + 1)$

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

   d. Quadratic: $C_{worst}(n) \in \Theta(n^2)$ (or $C(n) \in O(n^2)$).

   e. The algorithm is optimal because any algorithm that solves this problem must, in the worst case, compare $(n-1)n/2$ elements in the upper-triangular part of the matrix with their symmetric counterparts in the lower-triangular part, which is all this algorithm does.

Exercise 11

Solve the following recurrence relations.

1. $x(n) = x(n-1) + 5$ for $n > 1$, $x(1) = 0$
2. $x(n) = 3x(n-1)$ for $n > 1$, $x(1) = 4$
3. $x(n) = x(n-1) + n$ for $n > 0$, $x(0) = 0$
4. $x(n) = x(n/2) + n$ for $n > 1$, $x(1) = 1$ (solve for $n = 2^k$)
5. $x(n) = x(n/3) + 1$ for $n > 1$, $x(1) = 1$ (solve for $n = 3^k$)

1. a. $x(n) = x(n-1) + 5 \quad \text{for } n > 1, \quad x(1) = 0$

$$\begin{aligned} x(n) &= x(n-1) + 5 \\ &= [x(n-2) + 5] + 5 = x(n-2) + 5 \cdot 2 \\ &= [x(n-3) + 5] + 5 \cdot 2 = x(n-3) + 5 \cdot 3 \\ &= \ldots \\ &= x(n-i) + 5 \cdot i \\ &= \ldots \\ &= x(1) + 5 \cdot (n-1) = 5(n-1). \end{aligned}$$

Note: The solution can also be obtained by using the formula for the n term of the arithmetical progression:

$$x(n) = x(1) + d(n-1) = 0 + 5(n-1) = 5(n-1).$$

b. $x(n) = 3x(n-1) \quad \text{for } n > 1, \quad x(1) = 4$

$$\begin{aligned} x(n) &= 3x(n-1) \\ &= 3[3x(n-2)] = 3^2 x(n-2) \\ &= 3^2[3x(n-3)] = 3^3 x(n-3) \\ &= \ldots \\ &= 3^i x(n-i) \\ &= \ldots \\ &= 3^{n-1} x(1) = 4 \cdot 3^{n-1}. \end{aligned}$$

Note: The solution can also be obtained by using the formula for the n term of the geometric progression:

$$x(n) = x(1) q^{n-1} = 4 \cdot 3^{n-1}.$$

c. $x(n) = x(n-1) + n \quad \text{for } n > 0, \quad x(0) = 0$

$$\begin{aligned} x(n) &= x(n-1) + n \\ &= [x(n-2) + (n-1)] + n = x(n-2) + (n-1) + n \\ &= [x(n-3) + (n-2)] + (n-1) + n = x(n-3) + (n-2) + (n-1) + n \\ &= \ldots \\ &= x(n-i) + (n-i+1) + (n-i+2) + \cdots + n \\ &= \ldots \\ &= x(0) + 1 + 2 + \cdots + n = \frac{n(n+1)}{2}. \end{aligned}$$

d. $x(n) = x(n/2) + n \quad \text{for } n > 1, \quad x(1) = 1 \quad (\text{solve for } n = 2^k)$

$$\begin{aligned}
x(2^k) &= x(2^{k-1}) + 2^k \\
&= [x(2^{k-2}) + 2^{k-1}] + 2^k = x(2^{k-2}) + 2^{k-1} + 2^k \\
&= [x(2^{k-3}) + 2^{k-2}] + 2^{k-1} + 2^k = x(2^{k-3}) + 2^{k-2} + 2^{k-1} + 2^k \\
&= \ldots \\
&= x(2^{k-i}) + 2^{k-i+1} + 2^{k-i+2} + \cdots + 2^k \\
&= \ldots \\
&= x(2^{k-k}) + 2^1 + 2^2 + \cdots + 2^k = 1 + 2^1 + 2^2 + \cdots + 2^k \\
&= 2^{k+1} - 1 = 2 \cdot 2^k - 1 = 2n - 1.
\end{aligned}$$

e. $x(n) = x(n/3) + 1 \quad \text{for } n > 1, \quad x(1) = 1 \quad (\text{solve for } n = 3^k)$

$$\begin{aligned}
x(3^k) &= x(3^{k-1}) + 1 \\
&= [x(3^{k-2}) + 1] + 1 = x(3^{k-2}) + 2 \\
&= [x(3^{k-3}) + 1] + 2 = x(3^{k-3}) + 3 \\
&= \ldots \\
&= x(3^{k-i}) + i \\
&= \ldots \\
&= x(3^{k-k}) + k = x(1) + k = 1 + \log_3 n.
\end{aligned}$$

Exercise 12

Consider the following recursive algorithm (Algorithm 5) for computing the sum of the first n cubes: $S(n) = 1^3 + 2^3 + \ldots + n^3$

1. Set up and solve a recurrence relation for the number of times the algorithm's basic operation is executed.
2. How does this algorithm compare with the straightforward nonrecursive algorithm for computing this sum ?

Algorithm 5 *S(n)*

```
1: Input: A positive integer n.
2: Output: The sum of the first n cubes
3: if n = 1 then
4:     return 1
5:     return false
6: else
7:     S(n − 1) + n × n × n
8: end if
```

3. a. Let $M(n)$ be the number of multiplications made by the algorithm. We have the following recurrence relation for it:

$$M(n) = M(n-1) + 2, \quad M(1) = 0.$$

We can solve it by backward substitutions:

$$\begin{aligned} M(n) &= M(n-1)+2 \\ &= [M(n-2)+2]+2 = M(n-2)+2+2 \\ &= [M(n-3)+2]+2+2 = M(n-3)+2+2+2 \\ &= \ldots \\ &= M(n-i)+2i \\ &= \ldots \\ &= M(1)+2(n-1) = 2(n-1). \end{aligned}$$

b. Here is a pseudocode for the nonrecursive option:

Algorithm *NonrecS*(n)
//Computes the sum of the first n cubes nonrecursively
//Input: A positive integer n
//Output: The sum of the first n cubes.
$S \leftarrow 1$
for $i \leftarrow 2$ **to** n **do**
    $S \leftarrow S + i * i * i$
return S

The number of multiplications made by this algorithm will be

$$\sum_{i=2}^{n} 2 = 2\sum_{i=2}^{n} 1 = 2(n-1).$$

This is exactly the same number as in the recursive version, but the nonrecursive version doesn't carry the time and space overhead associated with the recursion's stack.