Complexity: Problems for Week 5

Exercise 1 *Show, for sufficiently large values of n, that $5n^4 + 2n^3 \leq 5.01n^4 - 100n^3$.*

Exercise 2 *Given the running times of programs/algorithms, evaluate the corresponding complexities.*

(a) The running time of my program, on an argument of size n, is $3n^2 + 9n + 8$ seconds. Is this $O(n^2)$? Is it $O(n)$? Is it $O(n^3)$?

(b) The running time of my program, on an argument of size n, is 5^n seconds for $n < 1000$, and $3n^2 + 9n + 8$ seconds for $n \geqslant 1000$. Is this $O(n^2)$? Is it $O(n)$? Is it $O(n^3)$?

(c) On an argument of size n, I first run a program whose running time is in $O(n^2)$, and then run a program whose running time is in $O(n^3)$. Show that the total running time is in $O(n^3)$.

(d) Suppose you have two algorithms to solve a given problem. The first algorithm has a running time of $3n^2 + 2n + 33$ while the second algorithm has a running time of $2^n - 5n + 5$. Which one will you prefer and why?

Exercise 3 *The following program operates on an array of characters that are all* `a` *or* `b`.

```
void f (char[] p) {
  elapse(1 second);
  for (nat i = 0; i<p.length(), i++) {
    if (p[i]=='a'){
      elapse(1 second);
    } else {
      elapse(2 seconds);
    }
    elapse (1 second);
  }
}
```

What is the average time taken to process an array of length 4, assuming that the character in position i (starting from 0) has probability 2^{-i} of being `a`*, and that the characters are independent? Also, what would be the worst case?*

Exercise 4 *(a) My program takes 2^{2^n} steps on every input of size $n < 100000$, and $5n^3 + 3n + 8$ steps on every input of size $n \geqslant 100000$. Show that the running time is in $O(n^3)$.*

(b) Show that if $f \in O(g)$ and $g \in O(h)$ then $f \in O(h)$.

(c) Show that $2^n \in O(n!)$

Exercise 5 *(a) A sorting method has Big-O complexity $O(n \log n)$. For $n > 1$, assume that time $T(n)$ of sorting n items is directly proportional to $n \log n$ – that is, $T(n) = Cn \log n$. Derive a formula for $T(n)$, given the time in milliseconds $T(1000) = 1$, and use this to estimate how long the method will take to sort $n = 1000000$ items.*

(b) One of the two software packages, A or B, should be chosen to process large databases that contain up to 10^{16} records. Given a number of records $n > 1$, the average processing time of Package A is $T_A(n) = 0.1n \log_2 n$ microseconds, and the average processing time of Package B is $T_B(n) = 6n$ microseconds.
For processing large databases as described, which algorithm is more efficient? Work out the exact conditions when these packages outperform each other.

Exercise 6 *Compute the time complexity (with respect to N) of the following functions. Give an informal justification. (Complexity proof is not required.)*

(a) The function A() is doing some processing on a string:

```
void A(String str){
  nat N = str.length();
  for(nat i = 0; i < N; i = i+1){
    // p seconds elapse
  }
  for(nat i = 0; i < N; i = i+1){
    for(nat j = 0; j < N; j = j+1){
      // q seconds elapse
    }
  }
  for(nat i = 0; i < N; i = i+1){
    for(nat j = 0; j < N; j = j+1){
      // r seconds elapse
    }
  }
}
```

(b) The function B() is doing some processing on a string:

```
void B(String str){
  nat N = str.length();
  for(nat j = 2 * N; j > 0; j = j-1){
    for(nat i = N; i > 0; i = i/2){
      // p seconds elapse
    }
  }
}
```

(c) The function C() is doing some processing on a string:

```
void C(String str){
  nat N = str.length();
  nat i = 1000;
  nat k = 0;
  // p seconds elapse
  while(i > 1){
    for(nat j = 1; j < N*N; j = j+1){
      // q seconds elapse
      if (j < N)
        k += 1; // r seconds elapse
    }
    i = i - 1; // s seconds elapse
  }
}
```

(d) The function D() is processing the number N, using recursion:

```
nat D(nat N){
  if (N == 1){
```

```
    return 1; // p seconds elapse
  }
  else{
    D(N-1);  // q seconds elapse
    D(N-1);  // q seconds elapse
  }
}
```

Exercise 7 *Callum writes a program that operates on an array of* `a`*'s and* `b`*'s. The time taken is* $5A^2 + 2B^3$*, where* A *is the number of* `a`*'s and* B *the number of* `b`*'s. If* n *is the length of the array, show that, in the worst case, the time taken is* $O(n^3)$*.*