COM2003 - MOCK ASSIGNMENT

JAMES MARSHALL

1. Empirical Performance Analysis

1.1. **Profiling.** Using the functions and lists defined in `Sorting.hs` collect runtimes for the functions `iSort`, `mergeSort` and `qSort`, for inputs from the lists `sortList`(1-3), `naturalNumbers` and `negativeNaturalNumbers`, on input sizes that seem reasonable to demonstrate how their runtime scales with input size.

Tips:

(1) To take the first 10 items from the list of natural numbers, for example, type `take 10 naturalNumbers`
(2) To collect timings type `:set +s` at the `ghci` prompt.

1.2. **Graphing.** Produce graphs of runtimes for the different algorithms on different sizes of different kinds of input.

Tips:

(1) To allow for random variation due to the algorithm or due to extrinsic factors you may want to run three replicates of the same algorithm on the same input type (such as the three different randomly-generated lists, for example), then take the average execution time
(2) Consider using a semilog plot for the more efficient algorithms (or for all of them), with the x-axis being logarithmic
(3) Generate your graphs using your preferred graphing environment, such as Matlab

2. Asymptotic Complexities

Using mathematical reasoning, or the complexity visualiser, answer the following questions.

(a) Put the asymptotic complexity hierarchy

$$O(n \log n) \subset O(n) \subset O(n^2) \subset O(2^n) \subset O(1) \subset O(\log n)$$

into the correct order

(b) Put the asymptotic complexity hierarchy

$$\Omega(n \log n) \subset \Omega(n) \subset \Omega(n^2) \subset \Omega(2^n) \subset \Omega(1) \subset \Omega(\log n)$$

into the correct order

Date: February 12, 2015

Submission deadline: 17:00 Wednesday 18th March 2015 at DCS Reception.

(c) Is $\Theta(n^2) \subset \Theta(2^n)$? Explain your answer

(d) Determine an appropriate asymptotic complexity for an algorithm with the following running time (in seconds). Explain your answer. How would this algorithm perform on large inputs?

$$\sum_{i=1}^{n} 4^i$$

(e) Determine an appropriate asymptotic complexity for an algorithm with the following running time (in seconds). Explain your answer. How would this algorithm perform on large inputs?

$$\sum_{i=1}^{n} (7 + 3i)$$

3. Recurrences and the Master Theorem

Apply the Master Theorem to determine the asymptotic complexities of the algorithms whose running time is given by the following recurrences. If the Master Theorem is not applicable, state why.

(a) $T(n) = T\left(\frac{2n}{3}\right) + 1$

(b) $T(n) = 9T\left(\frac{n}{3}\right) + n$

(c) $T(n) = 3T\left(\frac{n}{4}\right) + n \log_2 n$

(d) $T(n) = 4T\left(\frac{n}{2}\right) + n^2$

(e) $T(n) = 2T(n - 1) + 5$

(f) $T(n) = 4T\left(\frac{n}{2}\right) + n$

(g) $T(n) = 2T\left(\frac{n}{2}\right) + n \log_2 n$

(h) $T(n) = 4T\left(\frac{n}{2}\right) + n^3$

(i) $T(n) = T(n - 1) + 1$

4. Inductive Proofs

In the following proofs identify the *inductive hypothesis*, *base case* and *inductive step*. In proving each refer to the definitions (indicated by bracketed labels at the end of the lines) as you use them.

(a) Prove that `expsum z n == (1 - z ^ (n + 1)) / (1 - z)`:

```
expsum _ 0 = 1                                   --(expsum1)
expsum z k = z ^ k + expsum z (k - 1)            --(expsum2)
```

(b) Prove that `reverse (xs ++ ys) == reverse ys ++ reverse xs`[1]:

[1] from Thompson, chapter 8; you may assume associativity of the `++` operator

```
[] ++ ys = ys                                        --(++1)
(x:xs) ++ ys = x:(xs ++ ys)                          --(++2)

reverse [] = []                                      --(reverse1)
reverse (x:xs) = reverse xs ++ [x]                   --(reverse2)
```

(c) Prove that `foldr (*) 1 (map (2^) xs) == (2^) (foldr (+) 0 xs)`[2]:

```
foldr f s [] = s                                     --(foldr1)
foldr f s (x:xs) = f x (foldr f s xs)                --(foldr2)

map f [] = []                                        --(map1)
map f (x:xs) = f x: map f xs                         --(map2)
```

[2] assume `xs` is of type `[Integer]`