YSC2229: Introductory Data Structures and Algorithms

Week 03 Homework: Theory

This assignment consists of several problems dedicated to reasoning about asymptotic complexity of algorithms. Please, typeset your answers in LaTeX and submit the soultion as a single PDF file.

Problem 1. Assume that each of the expressions below gives the time demand $T(n)$ of an algorithm for solving a problem of size n. Specify the complexity of each algorithm using big O-notation.

1. $500n + 100n^{1.5} + 50n \log_{10} n$
2. $n \log_3 n + n \log_2 n$
3. $n^2 \log_2 n + n(\log_2 n)^2$
4. $0.5n + 6n^{1.5} + 2.5 \cdot n^{1.75}$

Problem 2. The following statements provide some "properties" of the big O-notation for the functions $f(n)$, $g(n)$ etc. State whether each statement is `TRUE` or `FALSE`. If it is true, provide a proof sketch using the properties of the O-notation. If it is false, provide a counter-example, and if you a way to fix it (*e.g.*, by changing the boundary on the right), provide a "correct" formulation a proof sketch while it holds.

1. $5n + 10n^2 + 100n^3 \in O(n^4)$
2. $5n + 10n^2 + 100n^3 \in O(n^2 \log n)$
3. Rule of products: $g_1(n) \in O(f_1(n))$ and $g_2(n) \in O(f_2(n))$, then $g_1(n) \cdot g_2(n) \in O(f_1(n) \cdot f_2(n))$.
4. Prove that $T_n = c_0 + c_1 n + c_2 n^2 + c_3 n^3 \in O(n^3)$ using the formal definition of the big O notation.

Problem 3. One of the two software packages, **A** or **B**, should be chosen to process data collections, containing each up to 10^{12} records. Average processing time of the package **A** is $T_A(n) = 0.1 \cdot n \cdot \log_2 n$ nanoseconds and the average processing time of the package **B** is $T_B(n) = 5 \cdot n$ nanoseconds. Which algorithm has better performance in the big O sense? Work out exact conditions when these packages outperform each other.

Problem 4. Express the complexity of Selection Sort using big-O notation. Justify your answer.

Problem 5. Find closed forms (explicit expressions) for the following recurrence relations on $f(n)$.

1. $f(0) = 4$ and for $n \geq 1$, $f(n) = f(n-1) + 5$
2. $f(0) = 3$ and for $n \geq 1$, $f(n) = 5f(n-1) - 2$
3. $f(1) = 1$ and for $n \geq 2$, $f(n) = n^2 f(n-1) + n \cdot (n!)^2$

Problem 6. Recall the definition of a matrix determinant by Laplace expansion:

$$|M| = \sum_{i=0}^{n-1} (-1)^i M_{0,i} \cdot |M^{0,i}|$$

where $M^{0,i}$ is the corresponding minor of the matrix M of size n, with indexing starting from 0. This definition can be translated to OCaml as follows:

```
let rec detLaplace m n =
  if n = 1 then m.(0).(0)
  else
    let det = ref 0 in
    for i = 0 to n - 1 do
      let min = minor m 0 i in
      let detMin =  detLaplace min (n - 1) in
      det := !det + (power (-1) i) * m.(0).(i) * detMin
    done;
    !det
```

Out of the explanations and the code above, estimate (in terms of big-O notation) the time complexity $t(n)$ of the recursive determinant computation. Start by writing down a recurrence relation on $t(n)$. Consider the complexity of returning an element of an array to be 0 (*i.e.*, $t(1) = 0$). For $n > 1$, consider the time cost of computing the `minor` of a matrix, `power`, addition, multiplication and other primitive operations to be constants and approximate all of them by a single constant c.