University of Waterloo
CS240 - Winter 2021
Assignment 1

There are written questions and a programming question in this assignment.

- the deadline for **all written questions, from Problems 1 to 6 inclusive,** is Wednesday January 27, 5pm;
- the deadline to submit your file `kWayMergeSort.cpp` **from Problem 7** is Wednesday February 3, 5pm.

The integrity of the grade you receive in this course is very important to you and the University of Waterloo. As part of every assessment in this course you must read and sign an Academic Integrity Declaration (AID) before you start working on the assessment and submit it **before the deadline of January 27** along with your answers to the assignment; i.e. **read, sign and submit A01-AID.txt now or as soon as possible**. The agreement will indicate what you must do to ensure the integrity of your grade. If you are having difficulties with the assignment, course staff are there to help (provided it isn't last minute).

The Academic Integrity Declaration must be signed and submitted on time or the assessment will not be marked.

Please read `http://www.student.cs.uwaterloo.ca/~cs240/w21/guidelines/guidelines.pdf` for guidelines on submission. **Each written question solution must be submitted individually to MarkUs as a PDF** with the corresponding file names: a1q1.pdf, a1q2.pdf, ..., a1q6.pdf.

It is a good idea to submit questions as you go so you aren't trying to create several PDF files at the last minute. **Remember, late assignments will not be marked but can be submitted to MarkUs after the deadline for feedback if you email cs240@uwaterloo.ca and let the ISAs know to look for it.**

Logarithms are in base 2, if not mentioned otherwise.

Problem 1 [3+3+3 marks]

Provide a complete proof of the following statements from first principles (i.e., using the original definitions of order notation).

a) $12n^4 - (\log(n))^4 \in O(n^4)$

b) $12n^4 - (\log(n))^4 \in \Omega(n^4)$

c) $2^n \in o(3^n)$

Problem 2 [3+3+3 marks]

For each pair of the following functions, fill in the correct asymptotic notation among Θ, o, and ω in the statement $f(n) \in \sqcup(g(n))$. Provide a brief justification of your answers. In your justification you may use any relationship or technique that is described in class.

a) $f(n) = (\log(n))^3 + (\log(n))^2$, $g(n) = (\log(n))^3$

b) $f(n) = 3^n$, $g(n) = 3^{n+\log(n)}$

c) $f(n) = n^3(5 + \sin(n))$, $g(n) = n^3$

Problem 3 [3+4+4 marks]

Analyze the following pieces of pseudocode and give a tight (Θ) bound on the running time as a function of n. Show your work and justify your answers (in all cases, n is assumed to be a positive integer).

a)
```
s = 0
for i = 1 to n * n do
   for j = 1 to i * i * i do
      s = s + 1
```

b)
```
p = 1
t = 2
s = 0
for i = 1 to n do
   for j = 1 to p do
      s = s + 1
   p = p * t
   t = t * t
```

c)
```
s = n
while (s > 4)
  if (s is even)
    s = s / 2
  else
    s = s - 1
```

Problem 4 [5+5+5 marks]

Prove or disprove each of the following statements. To prove a statement, you should provide a formal proof that is based on the definitions of the order notations. To disprove a statement, provide a counterexample and explain it.

a) We consider two algorithms, Algo1 and Algo2, that solve the same problem. We suppose that for any input of size n, Algo1 takes time $T_1(n)$ and Algo2 takes time $T_2(n)$. Finally, suppose that $T_1(n)$ is in $\Omega(n^3)$ and $T_2(n)$ is in $O(n^2)$. Does it imply that there exists n_0 such that for $n \geq n_0$, Algo2 runs faster than Algo1 on inputs of size n?

b) We consider two algorithms, Algo1 and Algo2, that solve the same problem. We suppose that for any input of size n, Algo1 takes time $T_1(n)$ and Algo2 takes time $T_2(n)$. Finally, suppose that $T_1(n)$ is in $\Omega(n^2)$ and $T_2(n)$ is in $O(n^3)$. Does it imply that there exists n_0 such that for $n \geq n_0$, Algo2 runs faster than Algo1 on inputs of size n?

c) Consider two functions $f(n)$ and $g(n)$, with $f(n)$ in $\Theta(g(n))$ and both $f(n)$ and $g(n)$ positive for all n. Does it imply that $f(n)^n$ is in $\Theta(g(n)^n)$?

Problem 5 [2+4 marks]

Dr. I. M. Smart has invented a new class of functions, denoted $O'(g)$: A function $f(n)$ is in $O'(g(n))$ if there is a constant $c > 0$ such that $f(n) \leq cg(n)$ for *all* $n > 0$. All functions map positive integers to positive integers.

a) Prove that $f(n) \in O'(g(n))$ implies that $f(n) \in O(g(n))$.

b) Prove that $f(n) \in O(g(n))$ implies that $f(n) \in O'(g(n))$.

Problem 6 [2+6+3+4+4]

We now consider the problem of merging several sorted arrays, and its application to sorting. In all questions, we only count the number of *key comparisons* we do, that is, comparisons *between array elements*; we are only interested in giving worst-case estimates. In all arrays, indices start at 0. Do not forget to justify your claims.

a) For the upcoming extension of Merge, we will need a function ArgMin(M), where M is an array of *pairs* $M[i] = (v_i, e_i)$ of length m. It should return an index ℓ such that v_ℓ is minimum among $v_0, \ldots, v_{m-1}$. If we write $M[i] = (v_i, e_i)$, then we suppose that we can access v_i as $v_i = M[i]$.first and e_i as $e_i = M[i]$.second.

Write the pseudo-code for such a function. For full marks, it should do $m - 1$ comparisons between the elements v_i's (and you should give a brief justification).

b) Consider the following extension of Merge to k-way-Merge($A_1, \ldots, A_k$), which merges k sorted arrays, for $k \geq 2$. Note that even for $k = 2$, the style of pseudo-code is slightly different from Merge as given in class (the algorithm is not in-place, for instance).

k-way-Merge($A_1, \ldots, A_k$)

1 let I be an array of length k initialized to $[0, \ldots, 0]$

2 let $n_1, \ldots, n_k$ be the lengths of $A_1, \ldots, A_k$ and $n = n_1 + \cdots + n_k$

3 let A be an array of length n

4 for $i = 0, \ldots, n-1$

4.1 let M be the array $[(A_j[I[j-1]], j),\ j \in [1, \ldots, k]$ such that $I[j-1] \neq n_j]$

4.2 let $\ell = \text{ArgMin}(M)$

4.3 let $j = M[\ell]$.second

4.4 $A[i] = A_j[I[j-1]]$

4.5 $I[j-1] = I[j-1] + 1$

There was a typo in the assignment as originally posted. Every time we used $I[j]$, it should have been $I[j-1]$, as we now do above.

Briefly justify correctness, by explaining what happens in the main loop (2 marks). Assuming that $\text{ArgMin}(M)$ does exactly $m-1$ comparisons if M has length m, prove that in the worst case, k-way-Merge does $(k-1)(n-k/2)$ comparisons (4 marks). Note that with $k = 2$, this is $n-1$, as expected.

Note: there exist better algorithms for k-way merge.

c) Use k-way-Merge to design an algorithm k-way-MergeSort(A, k) that sorts an array of length n by splitting it into k subarrays, instead of two for MergeSort (the subarrays should have lengths that generalize those used in MergeSort). The result should be written in A. Here, k is an integer in $2, \ldots, n$.

In the pseudo-code (and the implementation, see below), you do not need to use the tricks showed in class (where we avoided creating new arrays whenever possible).

d) For the analysis of k-way-MergeSort, we do the following approximation: instead of $(k-1)(n-k/2)$ comparisons in the worst case, suppose that k-way-Merge *always* does $(k-1)n$ comparisons (the missing terms do not affect the total too much). Under this assumption, give the *exact* number of comparisons in k-way-MergeSort, assuming that n is a power of k.

e) If A has length n, what does k-way-MergeSort(A, n) remind you of? In this case, what is the bound you obtained in the previous question?

Problem 7 Programming question [12 marks]

Submission deadline for this question is Wednesday February 3, 5pm.

Implement k-way-Merge and k-way-MergeSort from the previous problem to sort arrays of integers. For your arrays, you will use C++ vectors; you may use `push_back`.

We give you (on the assignment webpage) a starter file `kWayMergeSort.cpp`. You can compile it (using the `c++17` standard), but notice that the bodies of the functions `kWayMerge`,

`split` and `kWayMergeSort` are empty; do not change their signatures. You should complete this file and submit it once you are done (you can write new functions).

The `main` function reads lines from `cin` until it sees an EOF (you can assume that each line contains a single integer written in base 10). The first integer is k, the others are the numbers to sort. The `main` function prints the array, calls `kWayMergeSort` (which does nothing for the moment), and prints the array after execution. We also provide a sample input / output (you can pipe `input1.txt` into your program; once everything is implemented, you should obtain what is in `output1.txt`). Do not change anything in the main function you submit (in particular, leave the #`ifndef` and #`endif` we put there).