Algorithmics Problem Sheet 1

Jan Kurkofka, Will Turner

Summer 2024

This problem sheet is due on Monday 15th April. The tutorial will be 14:30-16:00 in LAM-2090. Please have a go at as many problems as you can. Each question solved gives you the number of points specified. A grade of 100% can be achieved by getting $10 = 16 - 6$ points. Additional points *do not* carry over to the next problem sheet.

Problem 1 — 1 point

Answer in one or two sentences: Other than speed (time efficiency), what other measures of efficiency might you need to consider in a real-world setting?

Problem 2 — 1 point

Answer in one or two sentences: Suggest a real-world problem in which only the best solution will do. Then come up with one for which an approximate solution is good enough.

Problem 3 — 2 points

Answer the following: How could we modify any sorting algorithm so that it always has a good **best-case** running time?

Problem 4 — 3 points

Consider the following method for sorting the numbers in an array $A[1 : n]$: Find the smallest element of $A[1 : n]$ and exchange it with $A[1]$. Then find the smallest element of $A[2 : n]$ and exchange it with $A[2]$. Then exchange the smallest element in $A[3 : n]$ with $A[3]$. Continue in this way until the array is sorted. This algorithm is known as *selection sort*.

Complete the following: Write the pseudocode for selection sort. Give the best-case and the worst-case running times in Θ-notation.

Problem 5 — 3 points

Consider the following computational problem: Input an integer n and two n-bit binary numbers, stored as two arrays $A[0 : n-1]$ and $B[0 : n-1]$. Output the sum of the two

numbers as an $(n+1)$-bit binary number, stored as an array $C[0:n]$.

COMPLETE THE FOLLOWING: Write an algorithm in pseudocode that solves the above problem. Work out its worst-case running time.

Problem 6 — 3 points

Consider the following algorithm called sum-array, which computes the sum of all of the numbers in the array.

Require: integer $n \geqslant 1$, array $A[1:n]$

1: sum $\leftarrow 0$
2: **for** $i = 1 \ldots n$ **do**
3: sum $\leftarrow$ sum $+A[i]$
4: **return** sum

COMPLETE THE FOLLOWING: State a loop invariant for this procedure. Use it to show that the procedure does what it's meant to.

Problem 7 — 3 points

CODE! Implement insertion sort in Python (or a language of your choice).