CMPSC 465 Data Structures & Algorithms
Fall 2021 Antonio Blanca and David Koslicki

HW 2

Due September 13, 10:00 pm

Instructions: You are encouraged to solve the problem sets on your own, or in groups of three to five people, but you must write your solutions strictly by yourself. You must explicitly acknowledge in your write-up all your collaborators, as well as any books, papers, web pages, etc. you got ideas from.

Formatting: Each part of each problem should begin on a new page. Each page should be clearly labeled with the problem number and the problem part. The pages of your homework submissions must be in order. You risk receiving no credit for it if you do not adhere to these guidelines.

Late homework will not be accepted. Please, do not ask for extensions since we will provide solutions shortly after the due date. Remember that we will drop your lowest two scores.

This homework is due Monday, September 13, at 10:00 pm electronically. You need to submit it via Gradescope (Class code 6PERPR). Please ask on Canvas about any details concerning Gradescope.

1. **(20 pts.) Big-O notation.** Assume you have functions f and g such that $f(n)$ is $O(g(n))$. For each of the following statements, decide whether it is true or false. Give a proof for the true statements and a counterexample for the false ones. (Hint: Think carefully before committing to an answer. Sometimes the answer that seems "obvious" might be wrong.)

   (a) $2^{f(n)}$ is $O(2^{g(n)})$
   (b) $f(n)^2$ is $O(g(n)^2)$
   (c) $2^{f(n)}$ is $O(2^{O(g(n))})$
   (d) $\log_2 f(n)$ is $O(\log_2 g(n))$
   (e) $f(n)\log_2 f(n)$ is $O(g(n)\log_2 g(n))$

2. **(20 pts.) Polynomial computation analysis.** Suppose we want to evaluate the polynomial

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

at some point x_0.

   (a) Consider the algorithm that computes first x_0^i for each i, using the fast exponentiation algorithm from lectures. Then, it computes $a_1 x_0$, $a_2 x_0^2$, $a_3 x_0^3, \ldots, a_n x_0^n$ (independently) and, finally, it adds all of these numbers to a_0 to obtain $p(x_0)$. How many sums and how many multiplications are involved in this algorithms? Please provide short explanations. (You can give your answer in O-notation.)
   (b) We show next how to do better using *Horner's rule*. Prove that the following algorithm outputs $p(x_0)$.

Algorithm 1 Horner's rule

```
z = a_n
for i = n − 1 to 0 do:
    z = z · x_0 + a_i
end for
return z
```

(c) How many additions and multiplications does this algorithm use as a function of n? Please provide short explanations. (You can give your answer in O-notation.)

3. (20 pts.) Exponentiation for large numbers. Consider the following algorithm for doing exponentiation.

Algorithm 2 Naive exponentiation algorithm

Input: integers x and y (n-bit binary numbers)
$z \leftarrow 1$, $i \leftarrow 0$
while $i < y$ **do**:
  $z = z \cdot x$
  $i = i + 1$
end while
Output: z

(a) Compute the running time of this algorithm assuming that it takes $O(mn)$ to multiply a n-bit by a m-bit number.

(b) Compute the running time of this algorithm assuming that it takes $O(n \log n)$ to multiply an n-bit by an m-bit number provided $n \geq m$.

4. (20 pts.) Fibonacci numbers. There is an alternative way of computing Fibonacci numbers involving matrices. Note that we have:

$$\begin{pmatrix} F_n \\ F_{n+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} F_{n-1} \\ F_n \end{pmatrix}.$$

If we write the latter equation recursively, we can get $\begin{pmatrix} F_n \\ F_{n+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$. Let $X = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$.

(a) Show that two 2×2 matrices can be multiplied using 4 additions and 8 multiplications.

(b) Show that for all $i < n$, all entries of X^i have $O(n)$ bits.

(c) The following recursive algorithm can be used to efficiently compute X^n.

Algorithm 3 `matrix`(X,n)

Input: X , n
if $n = 1$ **then**
  return X
end if
if n is even **then**
  $Z = \texttt{matrix}(X, \frac{n}{2})$
  return $Z \cdot Z$
end if
if y is odd **then**
  $Z = \texttt{matrix}(X, \frac{n-1}{2})$
  return $Z \cdot Z \cdot X$
end if
Output: `matrix(X,n)`

Show that the running time of this algorithm is $O(M(n) \log n)$, where $M(n)$ is the time it takes to multiply two n-bit number. (Hint: first show that there are $O(\log n)$ recursive calls, and then show each call takes at most $O(M(n))$, you may use the part b to show the latter).

5. (20 pts.) Heap and Heap Sort.

(a) What are the minimum and maximum numbers of nodes in a heap of height h?

(b) Is the array with values $\{10, 14, 19, 35, 31, 42, 27, 44, 26, 33\}$ a Min heap?

(c) Show that in the worst-case Heapify-UP could make $\Omega(\log n)$ swaps on a heap with n elements. (Hint: For a heap with n nodes, give node values that would cause Heapify-UP to be called recursively at every node on a simple path from the root down to a leaf.)

(d) What is the running time of HEAPSORT on an array A of length n that is already sorted in increasing order? What about in decreasing order?