# Homework 1. Due April 16, 6PM.

CS180: Algorithms and Complexity
Spring 2018

GUIDELINES:

- Upload your assignments to Gradescope by 6:00 PM.
- Follow the instructions mentioned on the course webpage for uploading to Gradescope very carefully (including starting each problem on a new page and matching the pages with the assignments); this makes it easy and smooth for everyone. As the guidelines are simple enough, bad uploads will not be graded.
- You may use results proved in class without proofs as long as you state them clearly.
- Most importantly, make sure you adhere to the policies for academic honesty set out on the course webpage. The policies will be enforced strictly. Homework is a stepping stone for exams; keep in mind that reasonable partial credit will be awarded and trying the problems will help you a lot for the exams.

1. Arrange the following in increasing order of asymptotic growth rate (i.e., $O(\ )$). For full credit it is enough to just give the order. [.75 points]
   (a) $f_1(n) = n^3$
   (b) $f_2(n) = 1000n^5/2$
   (c) $f_4(n) = n(\log n)^{1000}$
   (d) $f_5(n) = 2^{n \log n}$
   (e) $f_3(n) = 2^{3\sqrt{n}}$
   (f) $f_6(n) = 2^{(\log n)^{0.9}}$
2. Use Karatsuba's algorithm to multiply the following two binary integers: 10110100 and 10111101. Your entire calculations should be in binary and show all your work. [.75 points]
3. Solve the following recurrences by the Master theorem: [.75 points]
   (a) $T(n) = 4T(n/5) + n$.
   (b) $T(n) = 6T(n/3) + n$.
   (c) $T(n) = 16T(n/4) + n^2$.

4* We have a list $A$ of $n$ integers, for some $n = 2^k - 1$, each written in binary. Every number in the range 0 to $n$ is in the list exactly once, except for one. However, we cannot directly access the value of integer $A[i]$ (for any $i$); instead, we can only access the $j$'th bit of $i$: $A[i][j]$. Our goal is to find the missing number.

Give an algorithm to find the missing integer that uses $O(n)$ bit accesses. Prove that your algorithm runs in $O(n)$ time. Note that a brute force solution that accesses every bit will take time $\Theta(n \log n)$. [.75 points]

ADDITIONAL PROBLEMS. DO NOT turn in answers for the following problems - they are meant for your curiosity and understanding.

1. Call an array of $n$ numbers $B[0], B[1], \ldots, B[n-1]$ *$L$-regular* if for any index $i \in \{0, \ldots, n-1\}$, $B[i]$ occurs within $L$ places of its actual location in the sorted ordering. For example, any array of size $n$ is $n$-regular and the array $[1, 3, 2, 5, 4]$ is 1-regular.

   Give an algorithm which takes as input $L$ and a $L$-regular array $B$ of size $n$ and sorts the elements of $B$ in $O(n(\log L))$-time. You don't need to analyze the running time or prove correctness of your algorithm (but for full-credit, your algorithm must be correct and run in time $O(n(\log L))$.

   [Hint: One way is to adapt merge sort. Beware that proving correctness for this problem is quite tricky and there are several tempting 'wrong' proofs!]

2. Problems 2.8, 5.1, 5.4 from text.