CS 205 Final Question 2: Detecting Perfect Powers

16:198:205

Consider the problem of factoring - decomposing a number N into its composite factors, (e.g., $2881 = 67 * 43$). This is generally understood to be a hard problem (in that the difficulty increases rapidly with N). For some N however, factoring can actually be quite straightforward (consider numbers that end in a digit 5, or numbers that are the difference of two squares). One special case of easily factorable numbers: N that are perfect powers, i.e., $N = a^k$ for some integer $a, k > 1$. The purpose of this question is to analyze the following problem: ***i)*** *given an integer N, what is the complexity of determining if N is a perfect power, and* ***ii)*** *if N is a perfect power, what the values of a and k are such that $N = a^k$?* The main observation is that while computing roots (how do you compute a square root?) is difficult, computing powers and exponentiating can actually be quite easy.

In each of the following: when we ask for a big-O bound, we are interested in as tight an upper bound as you can justify.

1) Give a big-O bound on the number of bits needed to represent a number N. **(2 points)**

2) Give a big-O bound on the complexity of multiplying two integers between 1 and N? *Hint: what is the worst case?* Note, the bound should be simply in terms of N. **(3 points)**

3) Given N, k, what is the big-O complexity of computing N^k? *Hint: Consider fast exponentiation.* Note, the bound should be in terms of N and k. **(5 points)**

Consider the problem first of determining whether or not N is a perfect square:

4) Given that $1 \leq \sqrt{N} \leq N$, consider sequentially taking each number $a \in \{1, 2, 3, \ldots, N-1, N\}$ and computing a^2. Stop when either i) $a^2 = N$ and you have found the square root, or ii) $a^2 > N$ and N does not have an integer square root. Give a big-O bound on the overall complexity of this search in terms of N. **(4 points)**

5) As an alternate approach: we are effectively searching the set $\{1, 2, 3, \ldots, N\}$ for the value of $\sqrt{N}$ if it is there. For a given a, we can't compare a to $\sqrt{N}$ since $\sqrt{N}$ is not known. However, we can compute and compare a^2 to N. Use this idea as the basis of a *binary search*-type algorithm. Give a big-O bound on the overall complexity of this search, in terms of N. How does it compare to the previous? **(6 points)**

Generalize this idea then to determining whether or not N is a perfect k-th power for some given k (take k to be known):

6) Given that $1 \leq N^{1/k} \leq N$, consider sequentially taking each number $a \in \{1, 2, 3, \ldots, N-1, N\}$ and computing a^k. Stop when either i) $a^k = N$ and you have found the k-th root, or ii) $a^k > N$ and N does not have an integer k-th root. Give a big-O bound on the overall complexity of this search in terms of N and k. **(4 points)**

7) As an alternate approach: we are effectively searching the set $\{1, 2, 3, \ldots, N\}$ for the value of $N^{1/k}$ if it is there. For a given a, we can't compare a to $N^{1/k}$ since $N^{1/k}$ is not known. However, we can compute and compare a^k to N. Use this idea as the basis of a *binary search*-type algorithm. Give a big-O bound on the overall complexity of this search, in terms of N and k. How does it compare to the previous? **(6 points)**

However, we may not know what exponent k to look for - k may be unknown, and need to be determined:

8) For a given value of N, what is the smallest value of k that may need to be considered? What is the largest value of k? Give a big-O bound on the largest possible k. **(5 points)**

9) Consider extending the algorithm in Question 7 in the following way: for every possible k over the range above, use the algorithm in Question 7 to determine if N is a perfect k-power. If it is found that $N = a^k$ for some a, k, report them, otherwise report that N is not a perfect power. For clarity, write out the pseudocode of this algorithm given an input N. Give a big-O bound on the overall complexity of this search, in terms of N. Is this efficient? **(15 points)**

- *Bonus: What if we wanted to know if N was a perfect power* ***mod some prime*** *P. Could this algorithm still be used?*
- *Bonus 2: The above suggested a linear search over possible values of k. Could this be improved, and if so, how? In the case of a linear search, would it be better to try k smallest to largest, or largest to smallest?*