COP-4534: Algorithm Techniques

Homework 2

DUE: Sunday February 18 at 11:55 PM

• Please remember that all submissions must be typeset. Handwritten submissions will NOT be accepted. These must be uploaded to SCIS moodle in PDF format only.

• Please remember to type your name on top of your submission.

1. (20 points) An array $A$ of $n$ distinct numbers are said to be unimodal if there exists an index $k$, $1 \leq k \leq n$, such that $A[1] < A[2] < \cdots < A[k-1] < A[k]$ and $A[k] > A[k+1] > \cdots > A[n]$. In other words, $A$ has a unique peak and are monotone on both sides of the peak. Design an efficient algorithm that finds the peak in a unimodal array of size $n$.

2. (20 points) A patient has $n$ pills to take. In each day, he can take either one pill or two pills until all pills are gone. Let $T(n)$ denote the number of different ways the patient can take all $n$ pills. Give a closed form for $T(n)$. (Note that – for example – the two sequences $(1, 2, 2)$ and $(2, 1, 2)$ are considered as two different ways of taking 5 pills.)

3. (20 points) There are 7 jars of pills, each contains 20,000 identical pills. Each good pill weighs 10g, but bad pills can be 1 or 2 grams heavier or lighter than good pills (that is, possible weights of bad pills are 8, 9, 11 and 12 gram). If you are allowed to use a very accurate scale only once, how do you determine which jars contain good pills and which jars contain bad pills, as well as what are the weights of bad pills?

4. (20 points) Your friend claims that it is asymptotically faster to square an $n$-bit integer than to multiply two $n$-bit integers. Should you believe your friend? Back up your statement with arguments.

5. (20 points) Recall that in MERGESORT algorithm, we used a linear-time subroutine called MERGE which merges two sorted lists into a single sorted list. Now suppose there are $k$ sorted lists and there are $n$ elements in total in these $k$ lists. Design an efficient algorithm that merges these $k$ sorted lists into one sorted list. The running time of your algorithm should be a function of both $n$ and $k$. 