Assignment 1 - review questions
SOLUTIONS

1. You are given an array A of n distinct integers.

   (a) You have to determine if there exists a number (not necessarily in A) which can be written as a sum of squares of two distinct numbers from A in two different ways (note: m^2+k^2 and k^2+m^2 counts as a single way) and which runs in time $n^2 \log n$ in the **worst case** performance. Note that the brute force algorithm would examine all quadruples of elements in A and there are $\binom{n}{4} = O(n^4)$ such quadruples. (10 points)

   (b) Solve the same problem but with an algorithm which runs in the **expected time** of $O(n^2)$. (10 points)

   Hint: Go through all of $\binom{n}{2} = \frac{n(n-1)}{2}$ pairs $(A[k], A[m])$, $k < m$, of distinct integers in A; compute the sums $A[k]^2 + A[m]^2$ for all $1 \leq k < m \leq n$ and put them in an array B of size $n(n-1)/2$. What can you tell about the numbers in B if there exists a number (not necessarily in A) which can be written as a sum of squares of two distinct numbers from A in two different ways?

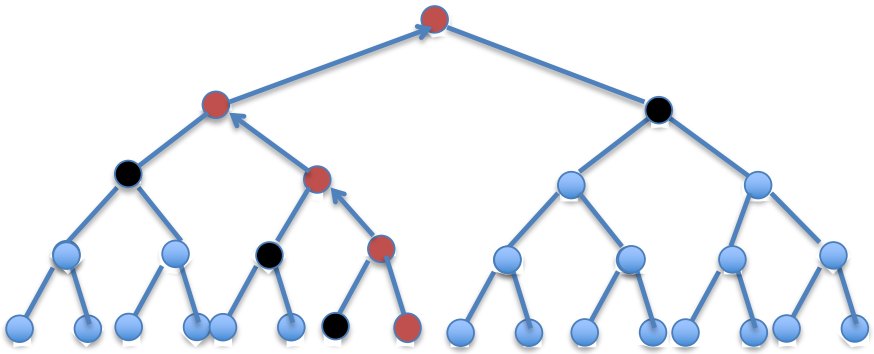
2. Suppose that you bought a bag of n bolts with nuts screwed on them. Your 5 year old nephew unscrewed all the nuts from the bolts and put both the nuts and the bolts back into the bag. The bolts are all of similar quite large size but are actually of many different diameters, differing only by at most a few millimetres, so the only way to see if a nut fits a bolt is to try to screw it on and determine if the nut is too small, if it fits or if it is too large. Design an algorithm for matching each bolt with a nut of a fitting size which runs in the **expected** time $O(n \log n)$. (20 points)

   Hint: Design a "double quick sort" by using a bolt to split the nuts into too small, fitting and too large and then use a matching nut to split all the bolts. Iterate.

3. You are given 1024 apples, all of similar but different sizes and a small pan balance which can accommodate only one apple on each side. Your task is to find the heaviest and the second heaviest apple while making at most 1032 weighings. (20 points)

   Hint: See the figure below; compare apples pairwise, staring with apples as leaves in a complete binary tree, advancing the bigger apple

to the next level. Among which apples on such a tree the second largest apple could be?



4. You are in an orchard which has a quadratic shape of size $4n$ by $4n$ with equally spaced trees. You purchased apples from n^2 trees which also form a square, but the owner is allowing to choose such a square anywhere in the orchard. You have a map with the number of apples on each tree. Your task is to chose such a square which contains the largest total number of apples and which runs in time $O(n^2)$. Note that the brute force algorithm would run in time $\Theta(n^4)$. (20 points)

   Hint: Assume you are given a sequence $A[i]$ of numbers of length $4n$, such as the price of a stock. To detect the trend of the price of such a stock, we often compute a moving average of every n consecutive elements of A, for example the sequence

$$a[i] = \frac{A[i] + A[i+1] + \ldots + A[i+n-1]}{n}, \quad 1 \leq n \leq 3n+1$$

   How would you compute the sequence $a[i]$ efficiently? Note that the brute force would take $(3n+1) \times n = O(n^2)$ many additions. You could do by brute force $a[1] = A[1] + A[2] + \ldots + A[n]$ using n additions, but how can you now obtain $a[2] = A[2] + A[3] + \ldots + A[n] + A[n+1]$ cheaply having already obtained $a[1]$?

5. Determine if $f(n) = O(g(n))$ or $g(n) = O(f(n)$ or both (i.e., $f(n) = \Theta(g(n))$) or neither of the two, for the following pairs of functions

   (a) $f(n) = (\log_2(n))^2$; $\quad g(n) = \log_2(n^{\log_2 n})^2$; $\quad$ (6 points)

(b) $f(n) = n^{10}$; $\quad g(n) = 2^{\sqrt[10]{n}}$; $\quad$ (6 points)

(c) $f(n) = n^{1+(-1)^n}$; $\quad g(n) = n$. $\quad$ (8 points)

Hint:

(a) Use the basic properties of the log function.

(b) Note that $f(n) \leq C\, g(n)$ just in case $\log f(n) \leq \log C + \log g(n)$. You might find useful the L'Hopital's rule which allows you to replace $\lim_{x\to\infty} F[x]/G[x]$ with $\lim_{x\to\infty} F'[x]/G'[x]$.

(c) It is NOT true that for every two functions $f(n)$ and $g(n)$ either $f(n) = O(g(n))$ or $g(n) = O(f(n))$!!