Assignment #2
Due date: Wednesday, April 15th, at 11am
Each question is worth 10 points

Submitting the second assignment: from your **<u>CSE</u>** account run

give cs3121 Assignment1 2_studendID.pdf

or

give cs3121 Assignment1 2_studendID.doc

For example:

give cs3121 Assignment1 2_30009.pdf

where "2_studendID.pdf " or "2_studendID.doc" is the file that contains your solutions with your name and studentID on EACH PAGE. "studendID" is your student ID number.

If you do not have a CSE account you can submit your assignment in either .pdf or .doc format via

https://cgi.cse.unsw.edu.au/~give/Student/give.php

Please **DO NOT** email your solutions.

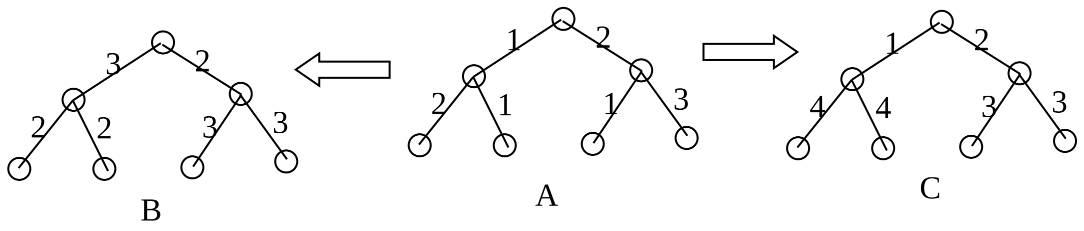
---

1) Improve the result from problem 5 of the previous assignment by showing that for every $e > 0$, given n real numbers $x_1,\dots,x_n$ where each x_i is a real number in the interval $[0, 1]$, there exists an algorithm that runs in ***linear time*** and that will output a permutation of the n numbers, say $y_1, \dots, y_n$, such that
$\sum_{i=2}^{n} |y_i - y_{i-1}| < 1 + e$.

2) To evaluate FFT($a_0,a_1,a_2,a_3,a_4,a_5,a_6,a_7$) we apply recursively FFT and obtain FFT(a_0,a_2,a_4,a_6) and FFT(a_1,a_3,a_5,a_7). Proceeding further with recursion, we obtain FFT(a_0,a_4) and FFT(a_2,a_6) as well as FFT(a_1,a_5) and FFT(a_3,a_7). Thus, from bottom up, FFT($a_0,a_1,a_2,a_3,a_4,a_5,a_6,a_7$) is obtained using permutation ($a_0,a_4,a_2,a_6,a_1,a_5,a_7$) as the leaves of the recursion tree of the original input sequence. Given any input ($a_0,a_1,a_2,\dots a_{2^n-1}$) describe the permutation of the leaves of the recursion tree.

(**Hint:** write indices in binary and see what the relationship is of the bits of the i^{th} element of the original sequence and the i^{th} element of the resulting permutation of elements as they appear on the leaves on the recursion tree.)

3) ***Timing Problem in VLSI chips***. Consider a complete balanced binary tree with $n = 2^k$ leaves. Each edge has an associate positive number that we call *the length* of this edge (see picture below). The *distance* from the root to a leaf is the sum of the lengths of all edges from the root to this leaf. The root sends a clock signal and the signal propagates along the edges and reaches the leaf in time proportional to the distance from the root to this leaf. Design an algorithm which ***increases*** the lengths of some of the edges in the tree in a way that ensures that the signal reaches all the leafs in the same time while the sum of the lengths of all edges is minimal. (For example, on the picture below if the tree A is transformed into trees B and C all leaves of B and C are on the distance 5 from the root and thus receive the clock signal in the same time, but the sum of lengths of edges in C is 17 while sum of lengths in B is only 15.)



4) Along the long, straight road from Loololong to Goolagong houses are scattered quite sparsely, sometimes with long gaps between two consecutive houses. Telstra must provide mobile phone service to people that live alongside the road, and the range of Telstra's cell base station is 5km. Design an algorithm for placing the ***minimal number*** of base stations alongside the road, that is sufficient to cover all houses.

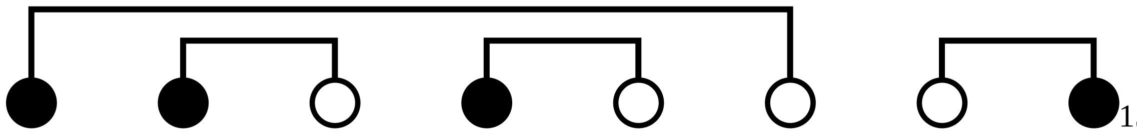
5) Assume you have \$2, \$1, 50c, 20c, 10c and 5c coins to pay for your lunch. Design an algorithm that, given the amount that is a multiple of 5c, pays it with a minimal number of coins and argue that it is optimal.

6) Assume denominations of your coins are $1, c, c^2, c^3, \ldots, c^n$ for some integer $c > 1$. Is the greedy algorithm, which, given any amount, pays it with the largest possible denominations first, optimal i.e. resulting in a minimal number of coins?

7) Give an example of a set of denominations containing the single cent coin for which the greedy algorithm does not always produce an optimal solution.

8) Alice wants to throw a party and is deciding whom to call. She has n people to choose from, and she has made up a list of which pairs of these people know each other. She wants to pick as many people as possible, subject to two constraints: at the party, each person should have at least five other people whom they know and at least five other people whom they do not know. Give an efficient algorithm that takes as input the list of n people and the list of all pairs who know each other and outputs a subset of these n people which satisfies the constraints and which has the largest number of invitees. Argue that your algorithm indeed produces a subset with the largest possible number of invitees.

9) Assume that you are given n white and n black dots, lying on a line, equally spaced. The dots appear in any order of black and white, see the **example** picture below. Design a greedy algorithm which connects each black dot with a (different) white dot, so that the total length of wires used to form such connected pairs is minimal. The length of wire used to connect two dots is equal to their distance along the line.



10) Design an algorithm which multiplies any polynomial of degree 16 with any polynomial of degree 8 using only 25 multiplications in which both operands can be arbitrarily large.

11) Simplify the expression $i\,(\omega_{64})^7\,(\omega_{32})^{15}$ where i is the imaginary unit.

12) Multiply the following three pairs of polynomials using at most the prescribed number of multiplications of large numbers (large numbers are those which depend on the coefficients thus can be arbitrarily large).

(a) $P(x) = a_0 + a_2 x^2 + a_4 x^4 + a_6 x^6$; $Q(x) = b_0 + b_2 x^2 + b_4 x^4 + b_6 x^6$ using at most 7 multiplications of large numbers;

(b) $P(x) = a_0 + a_{17} x^{17} + a_{19} x^{19} + a_{21} x^{21} + a_{23} x^{23}$; $Q(x) = b_0 + b_{17} x^{17} + b_{19} x^{19} + b_{21} x^{21} + b_{23} x^{23}$ using at most 16 multiplications of large numbers;

(c) $P(x) = a_0 + a_{100} x^{100}$ and $Q(x) = b_0 + b_{100} x^{100}$ with at most 3 multiplications of large numbers.