Algorithms & Data Structures

1. (10 points)

   (a) Consider this sorting algorithm that sorts an array *arr* of integers:

   ```
   def Sort(arr):
       n = len(arr)

       for i in range(n-1):
           for j in range(0, n-i-1):

               if arr[j] > arr[j+1] :
                   arr[j], arr[j+1] = arr[j+1], arr[j]
   ```

   Give the asymptotic time complexity of this algorithm, expressed in the number of comparisons. Is this an in–place algorithm?

   (b) Give the asymptotic order of the solution of the following recurrence equation:

   $$T(n) = 8T(n/2) + n^3 + 4n + 1/n$$

2. (a) (5 points)
   Given an array A sorted in decreasing order. Give an efficient algorithm that turns A into a heap.

   (b) (5 points)
   Given a completely filled binary tree of depth 3, where each node has as a (unique) key one of the letters A, B, . . ., or O, in such a way that the tree is alphabetically sorted post–order.
   What is the order in which you encounter the letters if you traverse this tree in a pre–order way?

3. (10 points)
   A game is played in a garden divided into n times n squares. In each square there is a number of pearls (at least 1 per square). You start in the leftmost lowest square (that has coordinates $(1, 1)$) and you are allowed each time to move upward or to the right. You have to end in the rightmost highest square (that has coordinates (n, n)). You may take the pearls in each square that you pass. The goal is to get as much pearls as possible.

   (a) Let $c(i, j)$ be the number of pearls in square (i, j), and $P(i, j)$ be the optimal total number of pearls you have collected when you arrive in square (i, j). Assume $P(i, j) = 0$ for $i = 0$ or $j = 0$. Motivate which of the following recurrence relations hold for $1 \leq i \leq n$, $1 \leq j \leq n$:

      i. $P(i, j) = max\{P(i - 1, j + c(i, i)), P(i - 1, j)\}$

ii. $P(i,j) = c(i,j) + max\{P(i-1,j), P(i,j-1)\}$

iii. $P(i,j) = c(i,j) + min\{P(i,j-1), P(i-1,j)\}$

iv. $P(i,j) = max\{P(i-1, j+c(i,j)), P(i+c(i,j), j-1)\}$

(b) Give an algorithm to determine the maximal number of pearls you can win. The complexity may not be bigger than $\Theta(n^2)$.