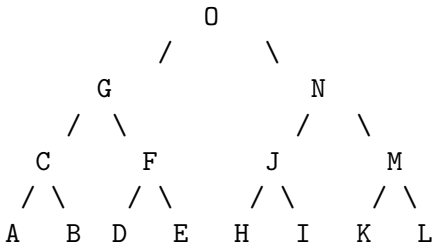
1. (a) This is actually bubble sort. The outer loop iterates $n$ times, the inner loop on average $n/2$ times, so asymptotic compexity $\Theta(n^2)$. It is an in–place sorting algorithm.

   (b) Use the Master Theorem, with $a = 8$ and $b = 2$, so $E = log8/log2 = 3$. $f(n) \in \Theta(n^3)$ holds, so this is condition 2, so $T(n) \in \Theta(n^3 log(n))$.

2. (a) $A$ is already a maxheap, so do nothing: complexity $\Theta(0)$.

   (b) The tree is




   If you traverse this tree in a pre–order way you encounter the letters in the order OGCABFDENJHIMKL.

   (a) If you arrive in square $(i, j)$ then the optimal number number of pearls is $c(i, j)$ plus the maximum of the pearls from where you come from, so the maximum of $P(i-1, j)$ and $P(i, j-1)$. So the right answer is ii.

   (b)
```
def maxpoints(c,n):

    P=[[0 for j in range(n+1)] for i in range(n+1)]

    for i in range(1,n+1):
        for j in range(1,n+1):
            P[i][j] = max(P[i-1][j],P[i][j-1]) + c[i][j]

    return P[n][n]
```
   The complexity of this algorithm is $\Theta(n^2)$.