Homework: Algorithms

1. Use the Master Theorem (if possible) to solve the following recurrences.

    1. $T(n) = 4T(n/3) + \log(n)n.$
    2. $T(n) = 3T(n/3) + \sqrt{n}.$
    3. $T(n) = 2T(n/2) + n\log(n)$
    4. $T(n) = 16T(n/4) + n$

2. Given the following divide and conquer algorithms, describe their run times with a recurrence relation.

```
def find(array):
   """array is an array of floating point numbers"""
   if len(array) = 1:
      return array[0]
   for i in range(0,len(array),2):
      if array[i]<array[i+1]:
           array[i], array[i+1] = array[i+1],array[i]
   return min(array[0:len(array):2]), max(array[1:len(array):2])
```

```
def find(array, lo, hi):
"""array is an array of integers.
   lo and ho are indices and count is a function
   and count(array, lo, hi, ele) takes c*n time"""
   if lo>hi: return 0;
   elif lo==hi: return array[lo]
   else:
      mid = (lo+hi)//2
      x = find(A, lo, mid)
      y = find(A, mid+1, lo)
      if x==y: return x
      if x>0:
          if count(A, lo, hi, x) > (hi-lo+1)//2:
              return x
      if y>0:
          if count(A, lo, hi, y) > (hi-lo+1)//2:
               return y
```

3. You are given a chess-board of size $2^n \times 2^n$ with one field marked. Find a divide-and-conquer algorithm that covers the whole chess-board with the exception of the marked elements with pieces that look like the one below:

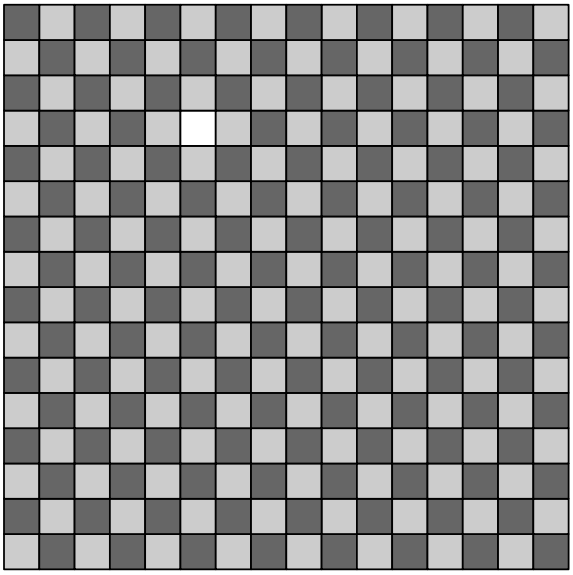

Figure 1: Chess board with one marked field

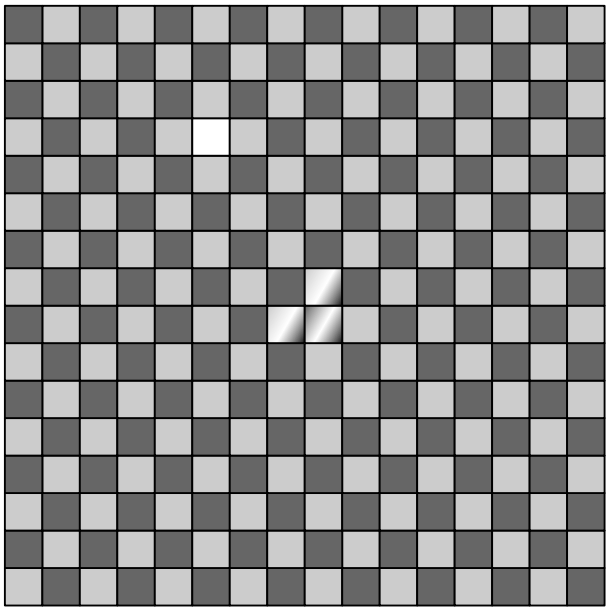

Figure 2: Chess board with a piece in the middle. This constitutes a hint.