1a. $T(n) = T(n/5)+cn, T(1)=1$

$T(n/5)=T(n/25)+c(n/5)$

$T(n/25)=T(n/125)+c(n/25)$

$T(n)=T(n/5^k)+c(n+(n/5)+(n/25)+...+(n/5^(k-1)))$

$c(n+(n/5)+(n/25)+...+(n/5^(k-1)))=cn(1+1/5+1/25+...+1/5^k-1)$

$T(n)=T(1)+cn*5/4=1+5/4cn$

$T(n)=O(n)$

1b. $T(n)=4T(n/3)+cn^2, T(3)=1$

$T(n/3)=4T(n/9)+c(n/3)^2$

$T(n/9)=4T(n/27)+c(n/9)^2$

$T(n)=4^k T(n/3^k)+cn^2(1+4/9+16/81+...+((4^(k-1))/(9^(k-1)))$

$$\sum_{i=0}^{k-1} (4/9)^i=9/5(1-(4/9)k)$$

$T(n)=4^k T(n/3^k)+cn^2 *9/5$

$T(n)=4^k=4^{log3n}$

$T(n) = O(n^{log3n})$

1c. $T(n) = 16T(n/2)+n^3, T(1)=1$

$T(n/2)=16T(n/4)+(n/2)^3$

$T(n/4)=16T(n/8)+(n/4)^3$

$T(n)=16^k T(n/2^k)+n^3(1+1/8+1/64+...+1/8^k-1)$

$$\sum_{i=0}^{k-1} (1/8)^i=8/7(1-(1/8)^k)$$

$T(n)=16^k T(n/2^k)+n^3 *8/7$

$T(n)=16^k=n^4$

$T(n)=O(n^4)$

2.
```
def find_fixed_point(A, low, high):
    if low > high:
        return None

    mid = (low + high) // 2

    if A[mid] == mid:
        return mid
    elif A[mid] > mid:
        return find_fixed_point(A, low, mid - 1)
    else:
        return find_fixed_point(A, mid + 1, high)

def find_fixed_point_main(A):
    return find_fixed_point(A, 0, len(A) - 1)
```

Recurrence relation: T(n)=T(n/2)+O(1)
This equals out to O(logn)

3.def merge(S1_prime, S2_prime):
```
    S_prime = S1_prime + S2_prime
    maximal_set = []

    for point in S_prime:
        dominated = False
        for other in maximal_set:
            if dominates(point, other) or dominates(other, point):
                dominated = True
                break
        if not dominated:
            maximal_set.append(point)

    return maximal_set

def dominates(A, B):
    return A[0] < B[0] and A[1] < B[1]

def maximal_subset(S):
    if len(S) <= 1:
        return S

    S.sort(key=lambda point: point[0])

    mid = len(S) // 2
    S1 = S[:mid]
    S2 = S[mid:]

    S1_prime = maximal_subset(S1)
    S2_prime = maximal_subset(S2)

    return merge(S1_prime, S2_prime)
```

Recurrence Relation: T(n)=2T(n/2)+O(n^2)
This solves out to O(n^2)