Problem 1

(a) T(N) = T(N/3) +T(2N/3) + O(N). In level K, (N/3 + 2N/3)^K. In case of (N/3)^K, T(N) = Ω(n log n), in case of (2N/3)^K, T(N) = O(n log n). So T(N) = Θ(n log n).

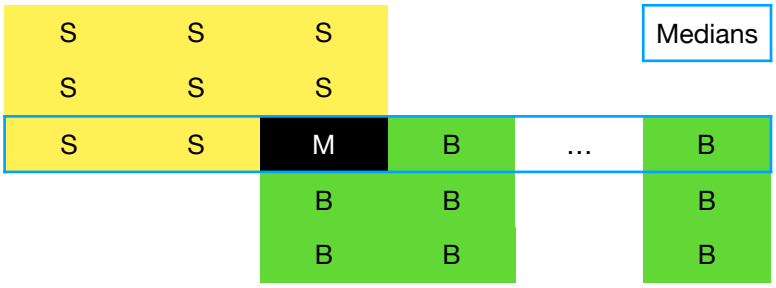
(b) T(N) = T(N/7) +T(6N/7) + O(N). In level K, (N/7 + 6N/7)^K. In case of (N/7)^K, T(N) = Ω(n log n), in case of (6N/7)^K, T(N) = O(n^4).

Problem 2

(a) The pivot group includes at least 3n/10 - 3 elements, and when M is included, it includes 3n/10 - 2 element. A non-pivot group (i.e., a left partial array) includes n-(3n/10-2), that is, up to 7n/10 + 2 elements. Therefore, in the worst case, the division ratio becomes 7n/10 + 2 : 3n/10 - 3, showing a ratio of approximately 7:3.

(b) The median value M of m_1, m_2, ..., m_ceiling (n/5) is recursively obtained. If the total number of elements is odd, the median is one, so there is no problem. If the total number of elements is even, one of the two median values is randomly selected. size of input of recursion is n/5. Select the appropriate side of the divided group and repeat recursively. Time complexity is Θ(n).

(c) If repeatedly entering a large group of elements to be found while the division is not balanced,



the performance goes beyond the linear shape. On the other hand, if a constant constant ratio is kept and divided continuously, the complexity is always O(n). Time Complexity: T(n) ≤ T(ceil(n/5)) + T(7n/10+2) + O(n), T(n) ≤T(n/5+1) + T(7n/10+2) + O(n) -> T(n) <= cn.

Problem 3

(a) a = 4, b = 2, n^2 = n^2, Θ(N^2)
(b) a = 2, b = 2, n > log n, Θ(n)
(c) a = 0.2, b = 2, n^-1 < n, Θ(log n)
(d) a = 8, b = 2, n^3 < 2^n, Θ(2^n)
(e) a = 2, b = 2, n > n / log n, Θ(n)

Problem 4

(a)
```
void kth_smallest_numbers(vector<int>&v ,int k) {
    vector<int> ans;
    map<int,int> m;

    for(auto i : v){
        m[i]++;
    }

    for(auto i : m){
        ans.push_back(i.first);
        k--;
        if(k == 0)
            break;
    }

    cout<<"All kth smallest element -: ";
        for(auto i : ans){
```

```
        cout<<i<<" ";
    }
}
```

(b) In for(auto i : m) in (a),

```
for(auto i : m){
    if(l != 0)
        k--; l--;
    else if (k == 0)
        break;
    else {
        ans.push_back(i.first);
        k—;
    }
}
```