**Lab W1D6**

**Question 1**

**a) “Wooden Block Toys” for two colors**

**Algorithm** toySort(S)

**Input** sequence S of colored toys

**Output** Sequence S in sorted form

start **<-** 0

stop **<-** S.length-1

pivotToy **<-** S[start]

i **<-** 0

j **<-** stop-1

**while** i <= j **do**

**while** i < stop **and** A[i] = pivotToy **do**

i++

**while** j => start and A[j] != pivotToy **do**

j--

**if** i < j **then**

swap(S, i, j)

i++;

j--;

swap(S, i, stop)

**return** A

Is your algorithm in place? **Yes**

What is the time complexity? **O(n)**

**b & c) “Wooden Block Toys” for three or four colors**

**Algorithm** toySort(S, start, numOfColors)

**Input** sequence S of colored toys, index to start sorting, numOfColors

**Output** Sequence S in sorted form

**if** numOfColors < 2 **then return** null

stop **<-** S.length-1

pivotToy **<-** S[start]

i **<-** 0

j **<-** stop-1

**while** i <= j **do**

**while** i < stop **and** A[i] = pivotToy **do**

i++

**while** j => start and A[j] != pivotToy **do**

j--

**if** i < j **then**

swap(S, i, j)

i++;

j--;

swap(S, i, stop)

toySort(S, ++i, --numOfColors)

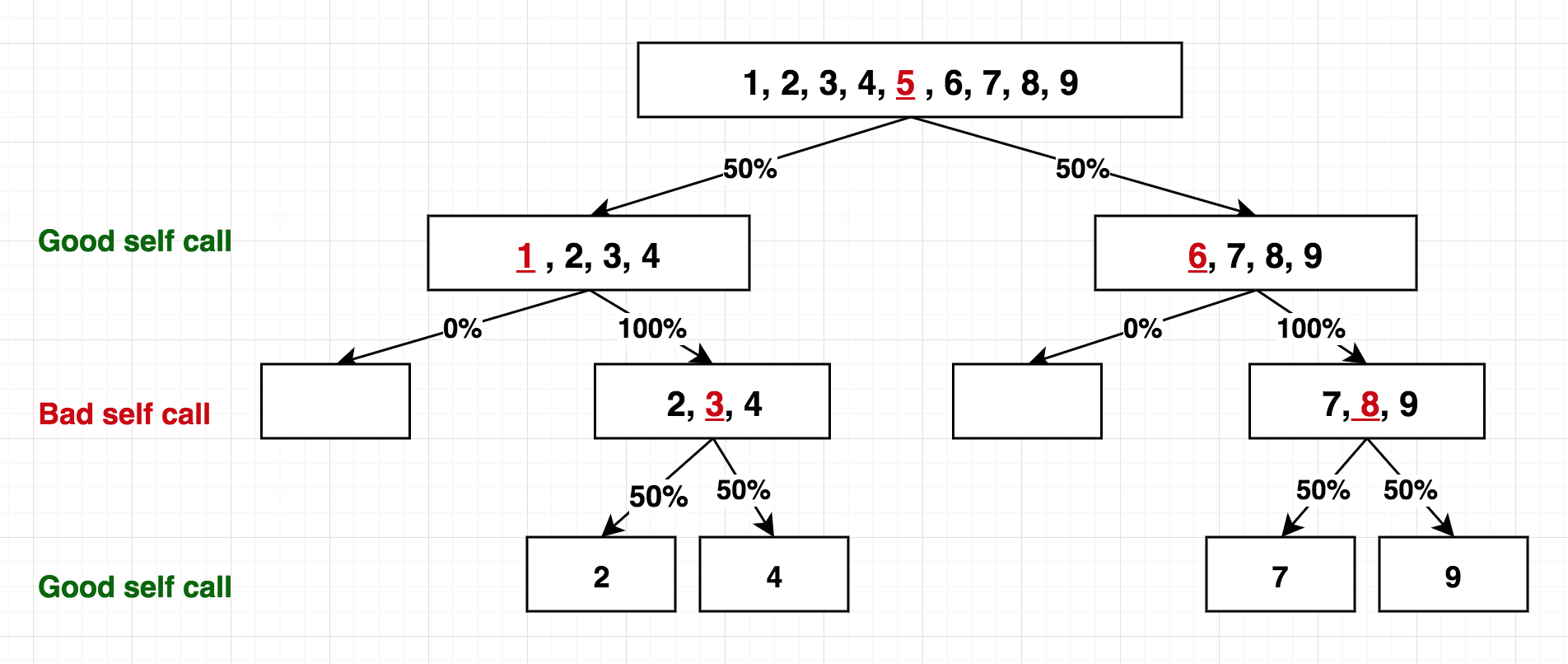
**return** S

Is your algorithm in place? **Yes**

What is the time complexity? **O(nlogn)**

**Question 2**

1. {1, 2, 3, 4, 5, 6, 7, 8, 9}



1. {8, 7, 6, 5, 4, 3, 2, 1, 9}

Diagram

Description automatically generated

1. {9, 1, 8, 2, 7, 3, 6, 4, 5}

Diagram

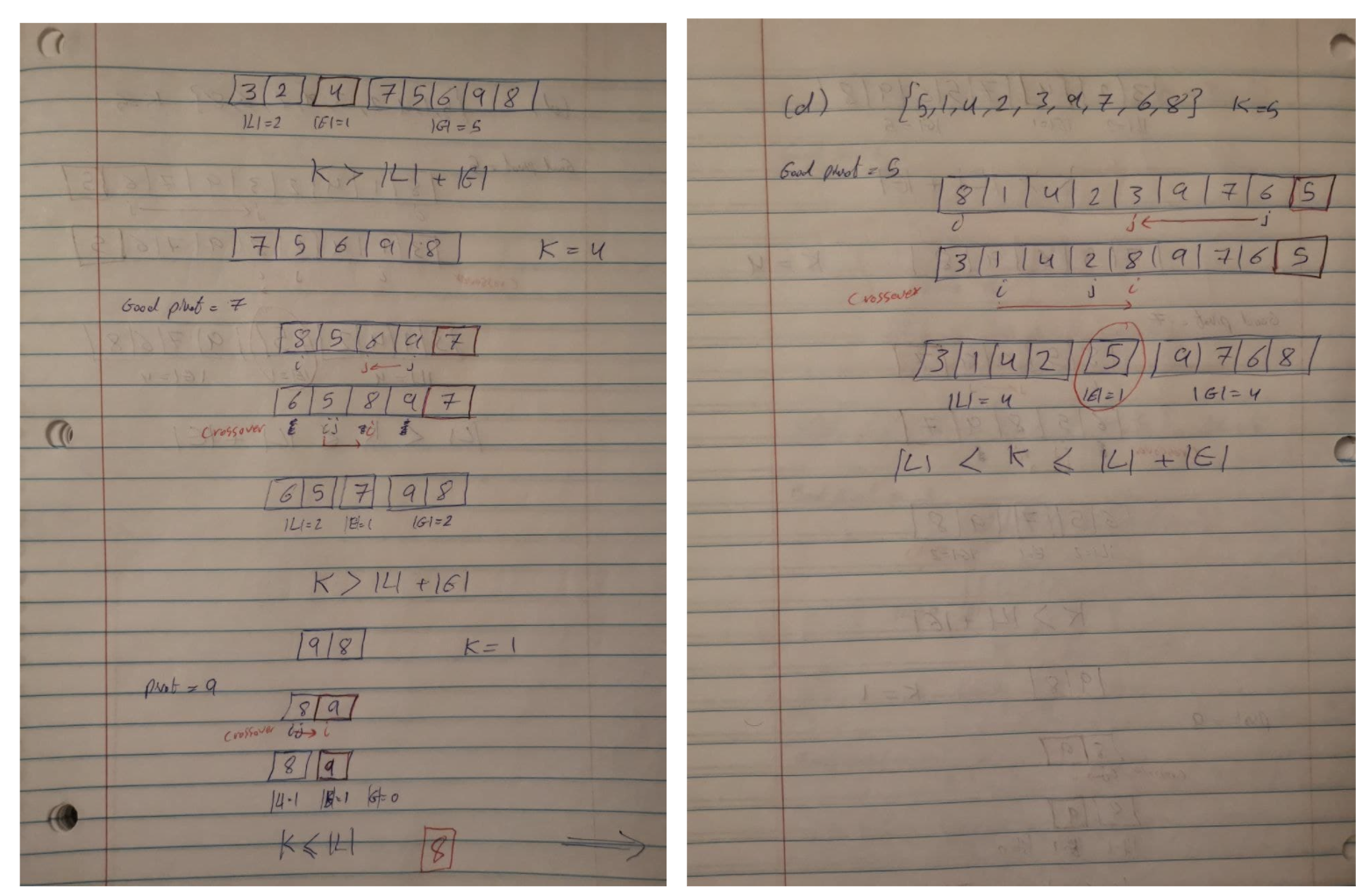
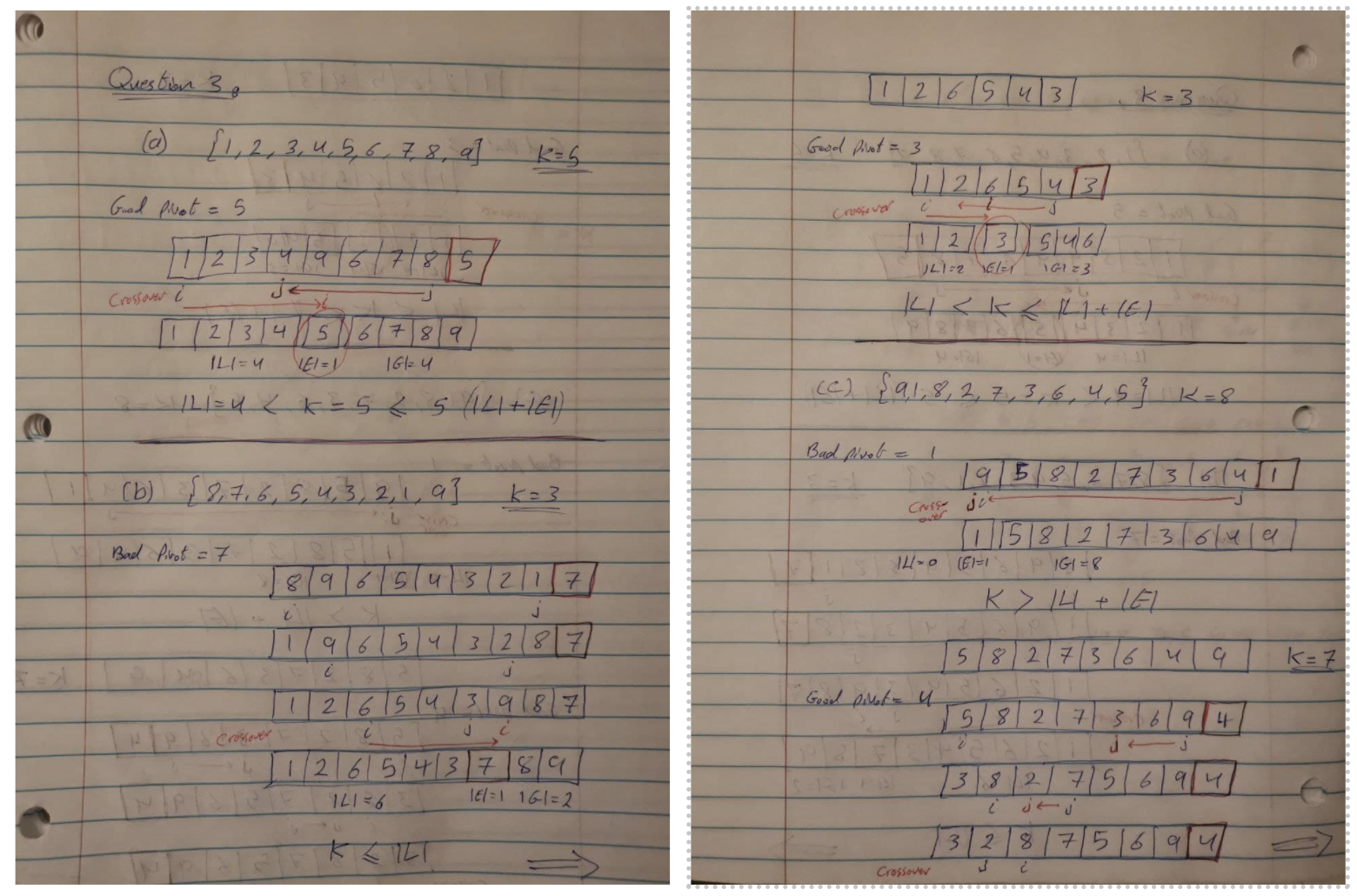
Description automatically generated

1. {5, 1, 4, 2, 3, 9, 7, 6, 8}

Diagram

Description automatically generated

**Question 3.**



**Question 4**

Let us redefine “Good Self Call” and “Bad Self Call”

**Good self-call:** the sizes of L and G are each less than **2n/3** (normal division)

**Bad self-call:** one of L and G has size greater than or equal to **2n/3**.

(a) Repeat the calculations shown in Slides 15, 16 and 17. (You need not draw pictures).

* Self-call is good with a probability of 1/3. At least 1/3 of the calls are good.
* The height of recursion tree is one less than the length of the descending sequence
* (2/3)n, (2/3)2n, . . ., 1, 0.
* Using Master’s theorem, a=1, b=3/2, k=1, a < bk Therefore, running time = **O(n)**
* The length is ≤ **1 + log 3/2 n**
* At each level of the recursion tree, total processing time is **O(n)**The total running time in the good case is **O(nlog n)**

(b) Are you able to derive the same results in Slides 16 and 17? If not, why?

Yes