# Prob1:

# possible outcome of 4-elements is 4!

In the worst case: # comparisons = height of decision tree that >= ceil(log(4!)) = 5

# Prob2:

Algorithm **arrange**(S)

Input sequence S with n integers

Output sequence S sorted in the weird way

S ← mergeSort(S)

R ← new Array

i ← 0

j ← n-1

k ← 0

while i<j do

R[k++] ← S[i++]

R[k++] ← S[j--]

if i=j then

R[k] ← S[i]

return R

a. Running time: O(nlogn) + O(n)

b. As professor proved in lecture, can’t be better than O(nlogn)

# Prob3:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| r[] | 27,72,27 | 1,64 |  |  |  |  |  | 34,16 | 80,8 |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|  |  |  |  |  |  |  |  |  |  |
| q[] | 1,8 | 16 |  | 27,27,34 |  |  |  | 64 | 72,80 |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

# Prob4:

static Integer find(int[] a) {

int[] bucket = new int[3 \* a.length];

for (int i = 0; i < bucket.length; i++) {

bucket[i] = 0;

}//O(3n)

for (int i = 0; i < a.length; i++) {

bucket[a[i]]++;

}//O(n)

for (int i = 0; i < a.length; i++) {

if (bucket[a[i]] == 1)

return a[i];

}//O(n)

return null;

}

Running time: O(3n + n + n) = O(n)