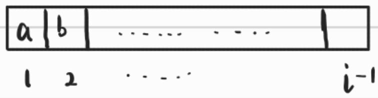
1.(a) The question can be simplified into the problem that insert a number into an array and how much is the probability of the number ahead of the second largest number.



since there are i places to be inserted. and the probability of any case is same as others.

so, the probability is $\frac{2}{i}$

(b) For part 1 : 1 comparsion.

For part 2 : the left n-2 number all need to be compared with b. → n-2 comparsions.

As for comparsion with a. only these number greater than b need comparsion.

so the number is $\sum_{i=3}^{n} \frac{2}{i} \cdot 1$

so the sum of comparsions is $n + \sum_{i=3}^{n} \frac{2}{i} - 1$

since $\int_{m-1}^{m} \frac{1}{x} dx > \sum_{i=m-1}^{m} \frac{1}{i+1} \Rightarrow 2\int_{3}^{n} \frac{1}{x-1} dx > \sum_{i=3}^{n} \frac{2}{i} \Rightarrow 2(\ln n-1 - \ln 2) > \sum_{i=3}^{n} \frac{2}{i+1}$

so the upper bound is $n + O(\ln n)$

2. 2.1> Prove: B is a sorted array.

In step 3. count[i] is the number of times that element $\leq i$ appear in the array

So in step 4. i must locate in [count[i-1]+1, count[i]]

Since all the elements from A are in $\{1, 2, \cdots k\}$, so we have count[1] $\leq$ count[2] $\leq \cdots \leq$ count[k]

For different numbers such as x, y, and suppose $x<y$. Then we have count[x] < count[y]

And x is in [count[x-1]+1, count[x]], y is in [count[y-1]+1, count[y]]

Therefore, x is in front of y in the array B.

2.2>. Prove the sorting is stable.

Just need to analyse the step 4.

Suppose $A[j_1] = A[j_2]$ and $j_1 < j_2$.

First, $j = j_2 \Rightarrow$ B[count[A[j_2]]] = A[j_2]. Then the count[A[j_2]] minus 1

. Then when $j = j_1$ B[count[A[j_1]]] = A[j_1]. And count[A[j_1]] is smaller than the orignal value at the time when $j = j_2$. so $j_1' =$ count[A[j_1]] < $j_2' =$ count[A[j_2]]

2.3> The time complexity is $O(n+k)$

Assume each loop of step 1. 2. 3. 4. take the time: C_1 C_2 C_3 C_4.

Step 1 takes $C_1 \cdot k$

Step 2 takes. $C_2 \cdot n$.

step 3 take $C_3 \cdot (k-1)$

step 4 take $C_4 (n)$

The sum of time is $(C_2+C_4)n + (C_1+C_3)k - C_3$

So the time complexity is $O(n+k)$

3. The algorithm shows as below:

Finding the Heavy: (Input the balls group)

Begin.

divide the balls into three groups: A. B. C.

Put A. B on the balance.

if A. and B weigh equally.

return Finding the Heavy (C).

else:

Put B. C on the balance,

if. B and C weigh equally.

return Finding the Heavy (A)

else

return Finding the Heavy (B)

As for the complexity, each time, the probability of only needing one weighing is 1/3.

So
$$\begin{cases} 10 & \text{the worst case} \\ \frac{25}{3} & \text{the average case} \\ 5 & \text{the best case} \end{cases}$$