Reservoir Sampling

<u>Algorithm()</u>:
•Initially we store the first K items(<==>elements) read from the stream in an data structure. An array(<==>list) of size K(1st item: index 1 , 2nd item: index 2, etc). So the first K elements read will be stored in the data structure we choose with certainty.
•Then for every item with (index number) > K(so we've already read K items and we continue to read from the stream) we produce a random number r between 1 and index(Example: if we've read K+5 items then index=K+5, which is invalid for our array of size K).
If: 1≤ r ≤ K we store the new value-item in the index r, otherwise(if r > K): we have an invalid index and the array stays the same. So every item i (where i>K) should have a probability = **K/i** of being added to our array. Note: i is the current index (or the number of items read so far) of the stream. It works like a counter.
So: • *if r ≤ K: the new item is inserted into the array at index r.*
• *else(r>K): the array remains the same.*
Where r is a rand number from 1 to i.
•We repeat the above process until we've read all the items of the stream.

•<u>Comment 1</u>: Everything happens in 1 pass: time complexity is O(N) and space complexity is O(K).
It is obvious that for every item i in the stream (with i>K), the probability of replacing an element in the array should be computed to ensure **uniformity**.

•<u>Comment 2</u>: I refer to index 1, 2, etc. in my Algorithm, but some programming languages (like C or Python) use 0-based indexing.

===
Simple proof:

• For every item, a random index r is selected between 1 and |item|.
(|item|: the number of items we've read at a certain point in time)
• For r≤K, the item at index r in our array is replaced with the new item we've just read from the stream. Otherwise, the array remains as it is.
• The 1st K items are added to the array(the probability= 1,which means certainty). There is no need to check if a certain condition is satisfied.
• Now for each item i>K, the chance of it being selected into the array is: **K/i** and the items already in the array have a probability of staying in the array is: **(1−K/i)** after each new item is read and processed.
• So at the end, the probability of each item being in the array after reading the stream of items is: **pr = K/N** .

$$\Pr(\text{"i item} \leq K \text{ remains"}) = \prod_{j=K+1}^{N} \left(1 - \frac{K}{j}\right) =$$

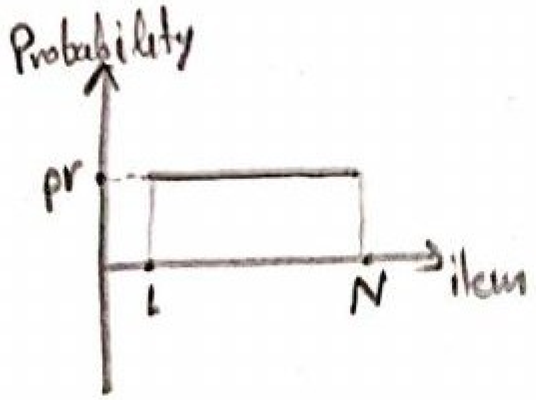
$$= \frac{K}{K+1} \cdot \frac{K+1}{K+2} \cdot \frac{K+2}{K+3} \cdot \ldots \cdot \frac{N-1}{N}$$

$$= \frac{K}{N}$$

$$\Pr(\text{"i item} > K \text{ remains"}) = \prod_{j=i+1}^{N} \left(1 - \frac{K}{j}\right) \cdot \frac{K}{i} =$$

$$= \left(\frac{i}{i+1} \cdot \frac{i+1}{i+2} \cdot \frac{i+2}{i+3} \cdot \ldots \cdot \frac{N-1}{N}\right) \cdot \frac{K}{i} =$$

$$= \frac{K}{i} \cdot \frac{i}{N} = \frac{K}{N}$$



$\Longrightarrow$ $\nexists$ bias για κάποιο item (element)

pr = K/N for $\forall$ item