1.

Give an $O(nlgk)$-time algorithm to merge k sorted lists into one sorted list, where n is the total number of elements in all the input lists.

To merge k sorted lists into a single sorted list, we can use a heap-based algorithm known as the k-way merge. The basic idea is to take the first element of each list, add them to a min-heap, and then repeatedly remove the smallest element from the heap and add it to the output list until the heap is empty.

Here's the algorithm:

```
merge_k_sorted_lists(lists):
   heap = empty heap
   for i = 1 to k:
      if lists[i] is not empty:
         add (lists[i][0], i) to heap
         # add first element of each list to heap

   result = empty list

   while heap is not empty:
      val, i = remove smallest element from heap
      append val to result
      if lists[i] has more elements:
         add (lists[i][0], i) to heap
         # add next element from corresponding list to heap
         remove first element from lists[i]

   return result
```

The key to the efficiency of this algorithm is the use of the min-heap to keep track of the smallest element from each list. The time complexity of the algorithm is $O(n\ lg\ k)$, where n is the total number of elements in all the input lists, and k is the number of input lists. This is because each element is added to and removed from the heap once, and the size of the heap is bounded by k. Hence, given the heap operations (insert/pop) are $O(lg\ k)$ and will only perform once for each element, the total time complexity of the algorithm is $n * O(lg\ k) = O(n\ lg\ k)$.

2. ***Suppose that instead of swapping element*** $A[i]$ ***with a random element from the subarray*** $A[i... n]$***, we swapped it with a random element from anywhere in the array.***

No, this code does not produce a uniform random permutation.

Given an array with an size n, this algorithm generate n^n uniform distributed events with its operations while the result will only contains $n!$ permutations. Hence, it is impossible to guarantee all the permutations can be uniformly generated for all n.

For example, consider the case where $n = 3$. There are $3! = 6$ possible permutations of the $array[\, A[1],\ A[2],\ A[3]\,]$. However, for example, the permutation $[\, A[1],\ A[2],\ A[3]\,]$ can be generated in four ways:

- By swapping the first element with itself, then swapping the second element with itself, and the third element with itself.
- By swapping the first element with the second element, then swapping the second element with the first element , and the third element with itself.
- By swapping the first element with the third element, then swapping the second element with itself, and the third element with the first element.
- By swapping the first element with itself, then swapping the second element with the third element, and the third element with the second element.

Therefore, the probability of generating the permutation $[\, A[1],\ A[2],\ A[3]\,]$ is $4/(3^3) = 4/27$, Thus, $[\, A[1],\ A[2],\ A[3]\,]$ is generated with a lower probability than $1/6$, which violates the requirement of a uniform random permutation.

3.

(a)

To calculate the expected number of incoming links to node v_j, we can consider the probability that any given node v_i $(i \neq j)$ in the network is connected to node v_j. For each node v_i, the probability that it is connected to v_j is $1/(i - 1)$, since there are $i - 1$ nodes in the network before v_i joins, and it selects one of them uniformly at random. Therefore, the expected number of incoming links to node v_j is:

$$E[incoming\ links\ to\ v_j]$$

$$= \sum_{i=j+1}^{n} P(v_i\ is\ connected\ to\ v_j)$$

$$= \sum_{i=j+1}^{n} 1/(i - 1)$$

$$= \sum_{i=1}^{n-1} 1/i - \sum_{i=1}^{j-1} 1/i$$

$$\equiv \Theta(ln\, n) - \Theta(ln\, j) = \Theta(ln\, n - ln\, j) = \Theta(ln\, \frac{n}{j})$$

(b)

In this model, the first node has no incoming links by definition. For any subsequent node j that joins, the probability that an existing node does not link to the new node is $(j - 2)/(j - 1)$. Thus, the probability that an existing node i $(when\ i \geq 2)$ has no incoming link is $\frac{i-1}{i} \times \frac{i}{i+1} \times\ ...\ \times \frac{n-2}{n-1} = \frac{i-1}{n-1}$.

Let X be the number of nodes with no incoming links in the final network with n nodes. Each node i has a probability of $\frac{i}{n-1}$ of having no incoming links. Since the nodes join the network independently, we can use the linearity of expectation to find the expected value of X:

$$E[X] = \sum_{i=2}^{n} \frac{i-1}{n-1} = \sum_{i=1}^{n-1} \frac{i}{n-1} = \frac{n(n-1)}{2(n-1)} = \frac{n}{2}$$