Producing a Uniform Random Permutation

Def: **A uniform random permutation is one in which each of the $n!$ possible permutations are equally likely.**

RANDOMIZE-IN-PLACE(**A**)

```
n ← length[A]
for i ← 1 to n
        do swap A[i] ↔ A[RANDOM(i, n)]
```

Lemma **Procedure** RANDOMIZE-IN-PLACE **computes a uniform random permutation.**

Def **Given a set of n elements, a k-permutation is a sequence containing k of the n elements.**

There are $n!/(n-k)!$ possible k-permutations of n elements

Proof via Loop invariant

We use the following loop invariant:

Just prior to the ith iteration of the for loop of lines 2– 3, for each possible $(i-1)$-permutation, the subarray $A[1\,..\,i-1]$ contains this $(i-1)$-permutation with probability $(n-i+1)!/n!$.

Initialization

RANDOMIZE-IN-PLACE(**A**)

1 $n \leftarrow length[A]$
2 **for** $i \leftarrow 1$ **to** n
3 **do swap** $A[i] \leftrightarrow A[\text{RANDOM}(i, n)]$

Just prior to the ith iteration of the for loop of lines 2– 3, for each possible $(i-1)$-permutation, the subarray $A[1\,..\,i-1]$ contains this $(i-1)$-permutation with probability $(n-i+1)!/n!$.

Initialization **Consider the situation just before the first loop iteration, so that $i = 1$. The loop invariant says that for each possible 0-permutation, the subarray $A[1\,..\,0]$ contains this 0-permutation with probability $(n-i+1)!/n! = n!/n! = 1$. The subarray $A[1\,..\,0]$ is an empty subarray, and a 0-permutation has no elements. Thus, $A[1\,..\,0]$ contains any 0-permutation with probability 1, and the loop invariant holds prior to the first iteration.**

Maintenance

RANDOMIZE-IN-PLACE(**A**)

```
1  n ← length[A]
2  for i ← 1 to n
3        do swap A[i] ↔ A[RANDOM(i, n)]
```

Just prior to the ith iteration of the for loop of lines 2– 3, for each possible $(i-1)$-permutation, the subarray $A[1 \mathinner{..} i-1]$ contains this $(i-1)$-permutation with probability $(n-i+1)!/n!$.

Maintenance **We assume that just before the $(i-1)$st iteration, each possible $(i-1)$-permutation appears in the subarray $A[1 \mathinner{..} i-1]$ with probability $(n-i+1)!/n!$, and we will show that after the ith iteration, each possible i-permutation appears in the subarray $A[1 \mathinner{..} i]$ with probability $(n-i)!/n!$. Incrementing i for the next iteration will then maintain the loop invariant.**

Let us examine the ith iteration. Consider a particular i-permutation, and denote the elements in it by $< x_1, x_2, \ldots, x_i >$. This permutation consists of an $(i-1)$-permutation $< x_1, \ldots, x_{i-1} >$ followed by the value x_i that the algorithm places in $A[i]$. Let E_1 denote the event in which the first $i-1$ iterations have created the particular $(i-1)$-permutation $< x_1, \ldots, x_{i-1} >$ in $A[1 \mathinner{..} i-1]$. By the loop invariant, $\mathbf{Pr}(E_1) = (n-i+1)!/n!$. Let E_2 be the event that ith iteration puts x_i in position $A[i]$. The i-permutation $< x_1, \ldots, x_i >$ is formed in $A[1 \mathinner{..} i]$ precisely when both E_1 and E_2 occur, and so we wish to compute $\mathbf{Pr}(E_2 \cap E_1)$. Using equation ??, we have

$$\mathbf{Pr}(E_2 \cap E_1) = \mathbf{Pr}(E_2 \mid E_1)\mathbf{Pr}(E_1) \ .$$

The probability $\mathbf{Pr}(E_2 \mid E_1)$ equals $1/(n-i+1)$ because in line 3 the algorithm chooses x_i randomly from the $n-i+1$ values in positions $A[i \mathinner{..} n]$. Thus, we have

$$\begin{aligned} \mathbf{Pr}(E_2 \cap E_1) &= \mathbf{Pr}(E_2 \mid E_1)\mathbf{Pr}(E_1) \\ &= \frac{1}{n-i+1} \cdot \frac{(n-i+1)!}{n!} \\ &= \frac{(n-i)!}{n!} \ . \end{aligned}$$

Termination

RANDOMIZE-IN-PLACE(**A**)

```
1  n ← length[A]
2  for i ← 1 to n
3        do swap A[i] ↔ A[RANDOM(i, n)]
```

Just prior to the ith iteration of the for loop of lines 2– 3, for each possible $(i-1)$-permutation, the subarray $A[1 \,..\, i-1]$ contains this $(i-1)$-permutation with probability $(n-i+1)!/n!$.

Termination **At termination, $i = n+1$, and we have that the subarray $A[1 \,..\, n]$ is a given n-permutation with probability $(n-n)!/n! = 1/n!$.**