Random Permutation

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong
Earlier we extended the RAM model with one more atomic operator: $\text{RANDOM}(x, y)$. This operator allows us to design algorithms with randomization.

Today we will discuss a randomized algorithm for permuting the elements of an array.
The Random Permutation Problem

We have an array \( A \) of \( n \) distinct integers, say, 1, 2, ..., \( n \). We want to design an algorithm to randomly permute these integers. Namely, when our algorithm finishes, \( A \) should be storing a sequence which can be any of the \( n! \) permutations with the same chance.
**Example**

Suppose that \( A = (1, 2, 3) \).

We must generate each of the following sequences with probability \( 1/6 \):  
- \((1, 2, 3)\)
- \((1, 3, 2)\)
- \((2, 1, 3)\)
- \((2, 3, 1)\)
- \((3, 1, 2)\)
- \((3, 2, 1)\)
The Algorithm

This problem can be solved in $O(n)$ worst case time by a beautiful 3-line algorithm:

1. for $i = 1$ to $n$
2. $x = \text{RANDOM}(1, i)$
3. swap $A[x]$ with $A[i]$
Example

Consider again $A = (1, 2, 3)$. We will demonstrate the execution of the algorithm by enumerating all of its possible outcomes.

Notice that the algorithm generates two integers: say $a$ for $i = 2$ and $b$ for $i = 3$. Specifically, $a$ can take 1 or 2 with the same probability, while $b$ can take 1, 2, or 3 with the same probability.

So there are 6 possibilities for the $(a, b)$ combination. Each possibility happens with probability precisely $1/6$.

The next slide shows the outcome of each possibility.
Example

- $(a, b) = (1, 1)$. Outcome: $A = (3, 1, 2)$.
- $(a, b) = (1, 2)$. Outcome: $A = (2, 3, 1)$.
- $(a, b) = (1, 3)$. Outcome: $A = (2, 1, 3)$.
- $(a, b) = (2, 1)$. Outcome: $A = (3, 2, 1)$.
- $(a, b) = (2, 2)$. Outcome: $A = (1, 3, 2)$.
- $(a, b) = (2, 3)$. Outcome: $A = (1, 2, 3)$.

Indeed, $A$ has been randomly permuted — each of the 6 permutations happens with probability $1/6$. 
Proof of Correctness

Remark: The proof will not be tested in exams.

We will prove that the algorithm is correct for any value of $n$ by induction. Correctness for $n = 1$ is obvious.

Assuming that the algorithm is correct for permuting $n - 1$ elements, next we prove that it is also correct for permuting $n$ elements.
Consider the for-loop with $i = n$. By the inductive assumption, now the first $n - 1$ positions of $A$ are storing a random permutation of $1, 2, \ldots, n - 1$.

That is, at this moment, each of the $(n - 1)!$ permutations is $(A[1], A[2], \ldots, A[n - 1])$ with probability exactly $1/(n - 1)!$. 

Yufei Tao

Random Permutation
Due to symmetry, consider any of those \((n - 1)\) permutations 
\((A[1], A[2], ..., A[n - 1])\). The for-loop with \(i = n\) will generate 
each of the following \(n\) permutations with the same probability \(1/n\):

- \((n, A[2], ..., A[n - 1], A[1])\)
- ...
- \((A[1], ..., A[i - 1], n, A[i + 1]..., A[n - 1], A[i])\)
- ...
- \((A[1], ..., A[n - 1], n)\)

It now follows that each of the \(n!\) permutations of \((1, 2, ..., n)\) is 
generated with probability precisely \(1/n!\).