CSE331: Introduction to Algorithms
Notes on Lecture 11: The Hiring Problem

Antoine Vigneron

October 12, 2020

1 Proof that PermuteBySorting fails with probability $< 1/n$

Here we prove that the probability that PermuteBySorting produces two equal keys is less than $1/n$. In other words, it produces n distinct keys with probability more than $1 - 1/n$.

Let E_{ij}, $i < j$ denote the event that $P[i] = P[j]$ after the loop is executed, and before we call Merge sort. As there are n^3 equally likely choices for $P[j]$, the probability that it is equal to $P[i]$ is $1/n^3$. So

$$\Pr[E_{ij}] = \frac{1}{n^3}$$

Let E be the probability that (at least) two keys are equal. Then we have

$$E = \bigcup_{i<j} E_{ij}$$

and thus

$$\begin{aligned}
\Pr[E] &= \Pr\left[\bigcup_{i<j} E_{ij}\right] \\
&\leqslant \sum_{i<j} \Pr[E_{ij}] \\
&= \binom{n}{2} \cdot \frac{1}{n^3} = \frac{n+1}{2n^2} \\
&< \frac{1}{n}.
\end{aligned}$$

The first inequality follows from the fact that the probability of a union of events is not larger than the sum of the probabilities of the events. □

2 Proof of correctness of RandomizeInPlace

In the following, we prove that RandomizeInplace produces a permutation of the input chosen uniformly at random.

Without loss of generality, assume that the input is $A[1 \dots n] = (1, 2, \dots, n)$, so we want to prove that the output is a permutation of $(1, \dots, n)$ chosen uniformly at random.

At the ith iteration of the loop, the algorithm generates a random number n_i between i and n. So the output is determined by the n-tuple of integers $(n_1, n_2, \ldots, n_n)$ where $n_i \in i, \ldots, n$. There are $n \times (n-1) \times \cdots \times 2 \times 1 = n!$ such tuples, which are equally likely. We will prove that each permutation corresponds to exactly one such tuple, and thus each permutation has probability exactly $1/n!$ of being computed by this algorithm.

We first illustrate this fact by an example. For instance, suppose that we start from $[1, 2, 3, 4, 5]$ and the output permutation is $[3, 2, 4, 1, 5]$. Then necessarily $n_1 = 3$, and after the first swapping the array was $[3, 2, 1, 4, 5]$. Then in order to obtain 2 at the second position, we must have $n_2 = 2$. After swapping 2 with itself, the array is still $[3, 2, 1, 4, 5]$. As 4 appears in 3rd position, $n_3 = 4$, and after swapping 1 with 4, we obtain the array $[3, 2, 4, 1, 5]$. After this, we must have $n_4 = 4$ and $n_5 = 5$ to keep the last two elements in order. So the only 5-tuple that generates $[3, 2, 4, 1, 5]$ is $(n_1, \ldots, n_5) = (3, 2, 4, 4, 5)$.

This generalizes to any value of n and any output permutation $\sigma = (\sigma_1, \ldots, \sigma_n)$. We must have $n_1 = \sigma_1$, then n_2 is the index of σ_2 in the array obtained after swapping $A[1]$ with $A[n_1]$, and n_3 is the index of σ_3 in the array obtained after swapping $A[2]$ with $A[n_2] \ldots$

This shows that each of the $n!$ possible permutation is obtained from exactly one of the $n!$ possible tuple, hence each permutation is generated with probability $1/n!$.