RWTH Aachen **WS 2017**
Theoretical Computer Science Exercise 2
Rossmanith–Dreier–Kuinke 7.11.2017

Exercise for Analysis of Algorithms

Exercise T2

Let S_N be the expected number of pushs on the stack in the Quicksort program, where the input consists of a random permutation of N distinct keys. Analyze S_N for all values of N and M.

Solution:

We start by giving a recurrence relation for S_N, note that we only push something on the stack if *both* halves are larger than M, that is, $M+1 \leq k \leq N-1-k$.

$$\begin{aligned} S_N &= \frac{1}{N}\sum_{k=0}^{N-1}\big(S_k + S_{N-1-k} + (M+1 \leq k \leq N-1-k)\big) \\ &= \frac{2}{N}\sum_{k=0}^{N-1} S_k + \frac{N-2M-2}{N} \end{aligned}$$

This equation is only valid for $N \geq 2M+2$ otherwise $S_N = 0$.
From the lecture we know that for $N > M$,

$$X_N = \frac{2}{N}\sum_{k=0}^{N-1} X_k + f_N = \frac{N+1}{M+2}X_{M+1} + (N+1)\sum_{k=M+2}^{N}\frac{kf_k - (k-1)f_{k-1}}{k(k+1)}$$

We need to define a function f_N that is valid for smaller values of N and get

$$f_N = \begin{cases} \frac{N-2M-2}{N} & N \geq 2M+2 \\ 0 & \text{else} \end{cases}$$

Now we can solve S_N. Since $S_{M+1} = 0$ and $f_k = 0$ for $k \leq 2M+2$ the expression simplifies a lot.

$$\begin{aligned} S_N &= (N+1)\sum_{k=2M+3}^{N}\frac{1}{k(k+1)} \\ &= (N+1)\Big(\frac{1}{2M+3} - \frac{1}{N+1}\Big) \\ &= \frac{N+1}{2M+3} - 1 \end{aligned}$$

Alternative Solution: It is also possible to solve S_N directly without using the formula for X_N

$$\begin{aligned} NS_N - (N-1)S_{N-1} &= 2S_{N-1} + 1 \\ NS_N &= (N+1)S_{N-1} + 1 \\ \sigma_N &:= \frac{S_N}{N+1} = \sigma_{N-1} + \frac{1}{N(N+1)} \\ \sigma_N &= \sum_{k=2M+4}^{N} \frac{1}{k(k+1)} + \sigma_{2M+3} = \frac{1}{2M+4} - \frac{1}{N+1} + \sigma_{2M+3} \\ S_N &= \frac{N+1}{2M+4} - 1 + \frac{(N+1)}{(2M+3)(2M+4)} = \frac{N+1}{2M+3} - 1 \end{aligned}$$

Exercise T3

Let $w \in \{a,b\}^n$ a word that has been chosen uniformly at random. How often is the body of the `while`-loop executed on average in the following algorithm? The function `is_palindrome` tests whether a word is a palindrome, i.e., the same when read backwards.

```
i = 2;
while (i <= n)
  if (is_palindrome(w[1],...,w[i]))
      return true;
  i++;
return false;
```

Solution:

Without loss of generality let $w[1]$ be a. If $w[2]$ also is a, which happens with probability $\frac{1}{2}$, we have found a palindrome after one round, and are done, otherwise, we have the prefix ab. In the following round we find a palindrome if an a occurs, otherwise, we have the prefix abb. It follows that we find a palindrome at that moment when a second a occurs. For $1 \leq k \leq n-2$, body is executed k times if and only if $w[2], \ldots, w[k] = b$ and $w[k+1] = a$. The body is executed $n-1$ times if and only if $w[2], \ldots, w[n-1] = b$, as the algorithm terminates after the last round no matter if it found a palindrome or not. The expected number of iterations therefore is

$$E = \Big(\sum_{k=1}^{n-2} \frac{1}{2^k} k\Big) + \frac{1}{2^{n-2}}(n-1).$$

We use the fact that

$$\sum_{i=1}^{b} \frac{1}{2^i} = 1 - \frac{1}{2^b}$$

to rewrite the first part of the sum

$$\begin{aligned}
&\sum_{k=1}^{n-2} \frac{1}{2^k} k = \sum_{k=1}^{n-2} \sum_{l=1}^{k} \frac{1}{2^k} \\
&= \sum_{k=1}^{\infty} \sum_{l=1}^{\infty} [1 \le k \le n-2][1 \le l \le k] \frac{1}{2^k} \\
&= \sum_{k=1}^{\infty} \sum_{l=1}^{\infty} [1 \le l \le k \le n-2] \frac{1}{2^k} \\
&= \sum_{l=1}^{n-2} \sum_{k=l}^{n-2} \frac{1}{2^k} \\
&= \sum_{l=1}^{n-2} \frac{1}{2^{l-1}} \sum_{k=1}^{n-l-1} \frac{1}{2^k} \\
&= \sum_{l=1}^{n-2} \frac{1}{2^{l-1}} (1 - \frac{1}{2^{n-l-1}}) \\
&= \frac{n-2}{2^{n-2}} + \sum_{l=1}^{n-2} \frac{1}{2^{l-1}} \\
&= \frac{n-2}{2^{n-2}} + 2 - \frac{1}{2^{n-3}}
\end{aligned}$$

We put both summands toghether to get the final result

$$E = \frac{n-2}{2^{n-2}} + 2 - \frac{1}{2^{n-3}} + \frac{n-1}{2^{n-2}} = 2 - \frac{1}{2^{n-3}} + \frac{1}{2^{n-2}} = 2 - \frac{1}{2^{n-2}}.$$

Exercise H2

We consider the following Algorithm. The array `a` contains a random permutation of the the numbers $1, \ldots, N$.

```
void doSomething(int *a, int N)
{
  int i;

  for (i=0; i<N-1; i++)   /*  1  */
    while (a[i] > a[i+1]) /*  2  */
      a[i]--;             /*  3  */
}
```

How often is line 3 executed on average?

Solution:

We define the random variable

$$Z_k = \max\{0, a[k] - a[k+1]\},$$

which states how often line 3 is executed when the variable i has the value k. In total, line 3 is executed $Z_1 + Z_2 + \cdots + Z_{N-1}$ times. By linearity of expectation, the expected number of executions is the sum of the expected values of these variables.

For two numbers $x, y \in \{1, \ldots, N\}$ mit $x \neq y$ holds $\Pr[x = a[k], y = a[k+1]] = (N-2)!/N! = 1/N(N-1)$, because there are $N!$ posible permutations and $(N-2)!$ permutations where two values are fixed.

For fixed x and y the third line is executed exactly $\max\{0, x-y\}$ times. This yields the expected value

$$E(Z_k) = \sum_{1 \leq y < x \leq N} \frac{x-y}{N(N-1)} = \sum_{x=1}^{N} \sum_{y=x+1}^{N} \frac{x-y}{N(N-1)} = \frac{N+1}{6}.$$

By linearity of expectation we get

$$E(Z_1 + \cdots + Z_{N-1}) = \sum_{k=1}^{N-1} E(Z_k) = \frac{(N-1)(N+1)}{6} = \frac{N^2-1}{6}$$

as the expected number of executions.