Analysis of Algorithms WS 2022
Prof. Dr. P. Rossmanith
M. Gehnen, H. Lotze, D. Mock

Date: November 7, 2022

Exercise Sheet with solutions 02

Due date: next tutorial session

Task T2.1

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, $M < k$ and $M < N - 1 - k$, or equivalently $M + 1 \leq k \leq N - M - 2$.

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

This equation is only valid for $N \geq 2M + 2$ otherwise $S_N = 0$.
From the lecture we know that for $N > M + 1$,

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

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

$$f_N = \begin{cases} \frac{N-2M-2}{N} & \text{for } N \geq 2M + 2, \\ 0 & \text{otherwise.} \end{cases}$$

Now we can get a closed formula for 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

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

Task T2.2

The next program is presented in x86 assembler language: Again the array `ds[0]...ds[2*N-2]` contains N pairwise distince natural numbers. Each permutation occurs with the same probability. How often is each instruction of this program executed on average?

`maxElem:`	`mov`	`ax, 0xFFFF`	$A\ ax \leftarrow -1;$
	`xor`	`dx, dx`	$A\ dx \leftarrow 0;$
`next:`	`cmp`	`dx, N`	$B\ i < N$?
	`jae`	`done`	B jump if above or equal ($i \geq N$)
	`mov`	`bx, ds:[2*dx]`	$C\ bx \leftarrow a[dx]$
	`cmp`	`bx, max`	$C\ bx > max$?
	`jna`	`skip`	C jump if not above ($bx \leq N$)
	`mov`	`ax, bx`	$D\ ax \leftarrow bx$
`skip:`	`add`	`dx, 0x0002`	$E\ ax \leftarrow ax + 1;$
	`jmp`	`next`	E jump
`done:`	`push`	`ax`	F push the maximum on the stack

Solution

Die Blöcke `A` und `F` werden offensichtlich jeweils einmal durchlaufen. Der Block `B` wird für jedes i von 0 bis N ausgeführt, also $N+1$-mal. Da der Sprung nach `done` nur für $i = N$ erfolgt, wird der Block `C` genau N-mal betreten. Die Anzahl der Durchläufe von `D` ergibt sich gemäß des letzten Teils der ersten Tutoraufgabe als H_N. Der Block `E` schließlich wird genauso oft betreten wie Block `C` (wobei `D` entweder durchlaufen oder übersprungen wird).

Block	Durchläufe	Kosten	Blockkosten	kum. Gesamtkosten
`A`	1	2	2	2
`B`	$N+1$	2	$2N+2$	$2N+4$
`C`	N	3	$3N$	$5N+4$
`D`	H_N	1	H_N	$5N+H_N+4$
`E`	N	2	$2N$	$7N+H_N+4$
`F`	1	1	1	$7N+H_N+5$

Im *average case* werden also $7N + H_N + 5$ Instruktionen ausgeführt; das sind $7N + \ln N + 5.78 + o(1)$ viele. Zur Einschätzung: der natürliche Logarithmus von einer Milliarde liegt unter 21.

Task H2.1 (10 pts)

We already analyzed C_n, the *total* expected number of comparisons in the two innermost `while`-loops of the quicksort algorithm (see the program fragment below).

What is the expected number of executions of the single comparison `a[i] < k`?

```
[...]
    i = l - 1; j = r ; k = a[j];
    do{
        do{i++;} while ( a[i] < k );
        do{j--;} while ( k < a[j] );
        t = a[i]; a[i] = a[j]; a[j] = t;
        } while ( i < j );
[...]
```

Solution

Let L_n be the expected number of times that the instruction `a[i] < k` is executed. If the pivot goes to position $a[k]$, then the number of comparisons (not taking into account the recursive steps) will be exactly k. Now the probability that the pivot goes to $a[k]$ is $1/n$. Therefore the number of comparisions in the non-recursive step is

$$\frac{1}{n}\sum_{k=1}^{n} k = \frac{n+1}{2}.$$

Now in the recursive step, the left subarray has size $k-1$ and the right subarray has size $n-k$. Therefore the expected number of comparisions in the recursive step is:

$$\frac{1}{n}\sum_{k=1}^{n}(L_{k-1}+L_{n-k}) = \frac{2}{n}\sum_{k=0}^{n-1} L_k.$$

The recurrence for the expected number of comparisions is therefore:

$$L_n = \frac{n+1}{2} + \frac{2}{n}\sum_{k=0}^{n-1} L_k,$$

the solution for which works out to be $L_n = C_n/2$.

Task H2.2 (10 pts)

We consider the following Algorithm. The array `a` contains a random permutation of 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.