RWTH Aachen **SS 2012**
Lehrgebiet Theoretische Informatik Problem Set 1
Langer—Reidl—Rossmanith—Sikdar 13.4.2012

Analysis of Algorithms — Tutorial

Problem 1-1

Let x_n, $n \geq 1$ be a solution to the recurrence relation $x_n = \sum_{k=1}^{n} x_k/k$. All solutions form a subvector space of $\mathbf{R}^{\mathbf{N}}$, the space of real sequences. What is the dimension of this subvector space and how does a general solution look like?

Problem 1-2

Consider the following algorithm that computes the maximum element in an array of non-negative integers. We assume that all elements are pairwise different and that each permutation occurs with equal probability.

```
int maxElem(int a[ ], int N) {
  int i, max;

  max = −1;
  for (i = 0; i < N; i++)
    if (a[i] > max)
      max = a[i];
  return max;
}
```

a) How often are the instructions $a[i] > max$ and $max = a[i]$ executed in the *worst case* and in the *best case*?

b) What is the probability that this worst case occurs?

c) How often are the two instructions executed in the *average case*?

Solution:

- Line 3 is executed N times. Since `max` is negative at the beginning, it must be overwritten in line 4 at least once even in the best case, when the largest element is at the beginning. In the worst case, the array is sorted and increasing, the condition in line 3 is always true and line 4 is executed N times.
- The best case occurs if the first element is the maximum element. The probability for this event is $1/N$. The worst case happens with a probability of only $1/N!$, since among the $N!$ permutations of the array only one is the increasing sequence.

- Line 3 is executed N times. The estimation for line 4 is more complicated, since the probability that line 4 is executed for $a[i]$ depends on the previous $a[j]$, $j < i$, and is thus not independent for all i. We can tackle this problem with the linearity of the expected value.
  The probability that the kth element is larger than its $k-1$ predecessors is $1/k$, since among the $k!$ permutations of the first k elements there are exactly $(k-1)!$, s.t. the largest element is at position k. The expected number of executions of line 4 is therefore
  $$\sum_{i=0}^{N-1} \frac{1}{i+1} = \sum_{k=1}^{N} \frac{1}{k} = H_N = \ln N + \gamma + o(1),$$
  where H_N is the Nth harmonic number and $\gamma \approx 0.78$.

Problem 1-3

Let w be a random word in $\{a,b\}^n$ chosen independently and with uniform probability. What is the expected number of iterations of the **while**-loop in the following algorithm? The function *is_palindrome* checks if the given word is a palindrome, i.e., if the word and its reverse are identical.

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

Solution: We w.l.o.g. assume w starts with a. If $w_2 = a$, too, which happens with a probability of $\frac{1}{2}$, then we already found a palindrome after one iteration. Otherwise we obtain the prefix ab. In the following iteration, we get a palindrome if $w_3 = a$, which again happens with probability $\frac{1}{2}$. Otherwise, we get the prefix abb. This observeration carried forward, we easily see that we find a palindrome once $w_i = a$ is read.
The expected number of iterations therefore is

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

where the last summand stems from the fact, that for $w = ab^{n-1}$ the `while`-loop is executed exactly $n-1$ times. Using the formula for the geometric progression, we therefore obtain

$$2 - \frac{2(n+1)}{2^n} + \frac{1}{2^{n-1}}(n-1) = 2 - \frac{1}{2^{n-2}}.$$

Homework Assignment 1-1 (10 Points)

Let a be an array of length N, whose entries are random numbers chosen from $\{1,\ldots,N\}$ independently and with uniform probability (i.e., repetitions are possible and likely). What is the expected number of executions of each line of the following algorithm?

```
count = 0;
i = 1;
while (i ≤ N)
    if (a[i]%2 ≡ 1)
        count++;
    i++;
return count;
```

Solution: Lines 1, 2, and 7 are executed once. Line 3 is executed $N + 1$ times. Lines 4 and 6 are therefore executed N times. For each execution of line 4, line 5 is executed with a probability of $\lceil N/2 \rceil / N$, which is $\frac{1}{2}$ for even N and $\frac{1}{2} + 1/2N$ for odd N. The expected number of iterations for line 5 therefore is $\lceil N/2 \rceil$.

Homework Assignment 1-2 (10 Points)

Two natural numbers $m \neq n$ are called *amicable*, if the sum of all proper factors of m equals n — and the other way around. A son and a father wrote the following programs that compute amicable numbers. What is the running time of the son's program? Find an exact formula for the number of executions of the instruction **if**(a % $i \equiv 0$) as a function of N. Assume that the constant 150000 is replaced by N.

Son

```
#include <iostream>

int e[150000];
int realdiv(int a) {
  int n = 0;
  for(int i = 1; i + i ≤ a; i++)
    if(a%i ≡ 0) n += i;
  e[a] = n;
  return n;
}

main() {
  for(int i = 0; i < 150000; i++) {
    int a = realdiv(i);
    if(a ≥ i) continue;
    if(e[a] ≡ i) std :: cout << i
      << "  " << a << "\n";
  }
}
```

Father

```
#include <stdio.h>
#define N 1000000
int factorsum[N];
int main() {
  int i;
  for(i = 1; i < N; i++) {
    int p = i;
    while(p < N) {
      factorsum[p] += i;
      p += i;
    }
  }
  for(i = 1; i < N; i++) {
    int a = factorsum[i] − i;
    if(a < i && i ≡ factorsum[a] − a)
      printf("%d %d\n", a, i);
  }
  return 0;
}
```

Solution:

The line in question is executed $\sum_{i=0}^{N} \lfloor \frac{i}{2} \rfloor$ times, from which we can already see that the running time will be $\Theta(N^2)$. Let us look at the precise number:

Assuming N is even, we get that

$$\sum_{i=0}^{N} \lfloor \frac{i}{2} \rfloor \quad = \quad \sum_{i \text{ even}} \frac{i}{2} + \sum_{i \text{ odd}} \frac{i-1}{2}$$

$$
\begin{aligned}
&= \sum_{i=0}^{N/2} i + \sum_{i=0}^{N/2-1} i \\
&= \frac{1}{2}\left(\frac{N}{2}(\frac{N}{2}+1) + (\frac{N}{2}-1)\frac{N}{2}\right) \\
&= \frac{N}{2}\left(\frac{N}{2}+1+\frac{N}{2}-1\right) \\
&= \frac{N^2}{4}
\end{aligned}
$$

Similarly, if N is odd, we obtain

$$
\begin{aligned}
\sum_{i=0}^{N} \lfloor \frac{i}{2} \rfloor &= \sum_{i \text{ even}} \frac{i}{2} + \sum_{i \text{ odd}} \frac{i-1}{2} \\
&= \sum_{i=0}^{(N-1)/2} i + \sum_{i=0}^{(N-1)/2} i \\
&= \frac{N-1}{2}\frac{N+1}{2} \\
&= \frac{N^2-1}{4}
\end{aligned}
$$