RWTH Aachen **SS 2014**
Lehrgebiet Theoretische Informatik Problem Set 1
Reidl—Rossmanith—Sanchez—Sikdar 14.4.2014

Analysis of Algorithms

Problem 1-1

Let x_n, $n \geq 1$ be a solution to the recurrence relation $x_n = \sum_{k=1}^{n} x_k/k$. Show that all solutions to this recurrence form a vector subspace of the space of real sequences. What is the dimension of this subspace 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.

```
1 int maxElem(int a[], int N) {
    int i, max;
3   max = -1;
    for (i = 0; i < N; ++i){
5     if (a[i] > max)
        max = a[i];
7   }
    return max;
9 }
```

---

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:

- The test: `if (a[i] > max)` is executed N times. Since `max` is negative at the beginning, it must be overwritten in the line: `max = a[i]` at least once. The best case is when the largest element is at the beginning, and this assignment is performed exactly once. In the worst case, the array is sorted and increasing, the condition `a[i] > max` is always true and the line `max = a[i]` 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.

- Since the line `if (a[i] > max)` is executed N times, we only have to estimate the average number of times the line `max = a[i]` is executed. This line is executed if and only if `a[i]` is larger than any of its $i-1$ predecessors. The probability that this happens is:

$$\frac{\binom{N}{i} \cdot (i-1)! \cdot (N-i)!}{N!}.$$

  This can be explained as follows: we first select i that constitute our array `a[0], ..., a[i-1]` in $\binom{N}{i}$ ways; place the maximum in position `a[i-1]` and permute the remaining elements in $(i-1)! \cdot (N-i)!$ ways. Since there are a total of $N!$ permutations, the expression for the probability is as written. This just simplifies to $1/i$. Therefore the expected number of times this line is executed is:

$$\sum_{i=0}^{N-1} \frac{1}{i+1} = \sum_{k=1}^{N} \frac{1}{k} = H_N \approx \ln N.$$

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?

```
1 count = 0;
  i = 1;
3 while (i <= N) {
    if (a[i] % 2 == 1)
5     ++count;
    ++i;
7 }
  return count;
```

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==0)` as a function of N. Assume that the constant 150000 is replaced by N.

Son

```
  #include <iostream>
2
  int e[150000];
4 int realdiv(int a) {
    int n=0;
6   for(int i=1; i+i<=a; i++)
      if(a%i==0) n+=i;
8   e[a] = n;
    return n;
10 }

12 main() {
    for(int i=0; i<150000; i++) {
14    int a = realdiv(i);
      if(a >= i) continue;
16    if(e[a]==i) std::cout << i
        <<     << a <<     ;
18  }
  }
```

Father

```
1 #include <stdio.h>
  #define N 1000000
3 int factorsum[N];
  int main() {
5   int i;
    for(i=1; i<N; i++) {
7     int p=i;
      while(p<N) {
9       factorsum[p] += i;
        p += i;
11    }
    }
13  for(i=1; i<N; i++) {
      int a = factorsum[i]-i;
15    if(a<i && i==factorsum[a]-a)
        printf(         , a, i);
17  }
    return 0;
19 }
```
