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

RWTH AACHEN UNIVERSITY

Date: December 5, 2022

# Exercise Sheet with solutions 06

Due date: next tutorial session, preferably in groups

## Tutorial Exercise T6.1

Consider the following algorithm that searches an element $x$ in a sorted array $a$ of length $n = km + 1$:

```
i:= 1;
while a[i]<=x
  if a[i]=x then return i;
  i:=i+m;
  if i>n return 0;
for j=i-1 downto max(1,i-(m-1))
  if a[j]=x then return j;
if a[j]<x then return 0;
return 0;
```

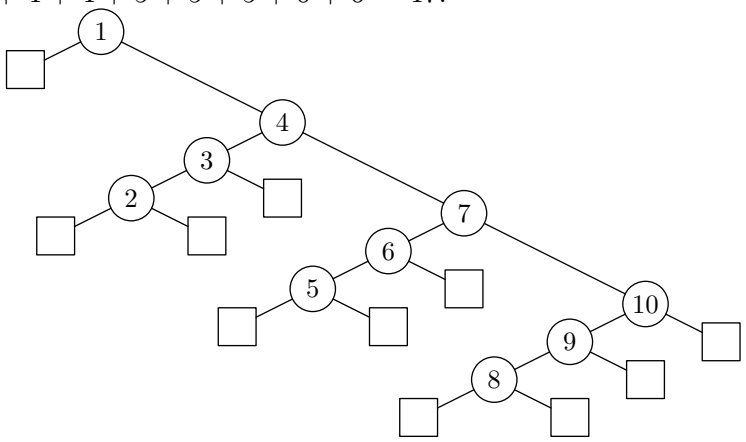
a) Draw the search tree and compute the internal and external path length for $n = 10$ and $m = 3$.

b) Determine $C^+$ and $C^-$ for arbitrary $m$, $k$.

c) What is, for given $n$, the best choice for $m$ w.r.t. the running time?

## Solution

a) The path lengths are $\pi(T) = 0 + 1 + 2 + 2 + 3 + 3 + 3 + 4 + 4 + 5 = 27$ and $\xi(T) = 1 + 3 + 4 + 4 + 4 + 4 + 5 + 5 + 5 + 6 + 6 = 47$:




b) It is sufficient to compute $\pi$. We then obtain

$$C^+ = \frac{\pi(T)}{n} + 1 \text{ and } C^- = \frac{\pi(T) + 2n}{n+1}.$$

Thus,

$$\begin{aligned}
\pi(T) &= \sum_{i=1}^{k} \sum_{j=i}^{i+m-1} j \\
&= \sum_{i=1}^{k} \frac{(i+m)(i+m-1)}{2} - \frac{i(i-1)}{2} \\
&= \frac{1}{2} \sum_{i=1}^{k} (2mi + m(m-1)) \\
&= \frac{mk(k+1) + km(m-1)}{2} \\
&= \frac{km^2 + mk^2}{2}.
\end{aligned}$$

Testing this for the example a) yields: $\pi(T) = (3 \cdot 9 + 3 \cdot 9)/2 = 27$.

c) In both cases, the search depths (in the average case) depends linear on $\pi(T)$. Hence we need to minimize this value. We express $\pi$ using $n' := n - 1 = km$, thereby ignoring the constant.

$$\frac{km^2 + mk^2}{2} = \frac{n'^2/k + n'k}{2}$$

Deriving by $k$ yields $n' - n'^2/k^2 = 0$. Hence, we have $k = \sqrt{n'}$. If $n = k^2 + 1$, this is the optimal value. Otherwise, we simply round to the closest integer, which is optimal because of symmetry.

**Homework Exercise H6.1**

We continue to look at the binary words defined in H5.3. Élisabeth Philippe Marie Hélène de Bourbon wants to write a program that generates such words. Let $W_n$ be the set of all well-formed words of length $n$. The program should output one of the words randomly—such that every word in $W_n$ is output with the same probability. Daniel's method from H5.3 turned out to be too slow for large $n$.

Invent a method to generate such a word in time $O(n^2)$ and implement it. Do not forget that just adding two $n$-bit numbers takes time $\Theta(n)$.

**Solution**

First we compute the probability that a word of length $n$ starts with a prefix 01, 10, or 11. We compute these probabilities as rational numbers by dividing the number of words with a specific prefix by the total number of words of length $n$. By using dynamic programming and storing the results for all lengths up to $n$ this takes $O(n)$ time.

After this preprocessing, a random word can easily be generated from left to right: If we already generated the prefix, say, 011010, we look at the last two digits, here 10. The we can continue with either 10 or 01. We just choose among the two possibilities with the right probability, which we get from the tables. Generating two digits take $O(n)$ time in this way, making a total running time of $O(n^2)$.

The program in Figure 1 does the job for even $n$.

**Homework Exercise H6.2**

Use summation factors to solve the following recurrence:

```
import random

cached = { }

def numwords(prefix, n):
  if(prefix, n) in cached : return cached[(prefix, n)]
  if n < 2 : return 0
  if n == 2 : return 1
  if n == 3 :
    if prefix == "01" : return 2
    else : return 1
  if prefix == "" :
    return numwords("01", n) + numwords("10", n) + numwords("11", n)
  if prefix == "11" : return numwords("01", n - 2)
  if prefix == "01" : pre1, pre2 = "10", "01"
  if prefix == "10" : pre1, pre2 = "10", "11"
  num = numwords(pre1, n - 2) + numwords(pre2, n - 2)
  cached[(prefix, n)] = num
  return num

def randword(prefix, n):
  word = prefix
  while n > 2 :
    if prefix == "11" :
      prefix = "01"
      word += "01"
      n = n - 2
      continue
    if prefix == "01" : pre1, pre2 = "10", "01"
    if prefix == "10" : pre1, pre2 = "10", "11"
    n1 = numwords(pre1, n - 2)
    n2 = numwords(pre2, n - 2)
    if random.randint(1, n1 + n2) ≤ n1 : prefix, word = pre1, word + pre1
    else : prefix, word = pre2, word + pre2
    n = n - 2
  if n == 1 :
    if prefix == "11" : word += "0"
    elif prefix == "10" : word += "1"
    elif prefix == "01" : word += random.select(["0", "1"])
  return word

def randomword(n):
  n1 = numwords("01", n)
  n2 = numwords("11", n)
  n3 = numwords("10", n)
  z = random.randint(1, n1 + n2 + n3)
  if z ≤ n1 : return randword("01", n)
  if z ≤ n1 + n2 : return randword("11", n)
  else : return randword("10", n)
```

Abb. 1

$$
\begin{aligned}
a_0 &= 0 \\
a_n &= \frac{a_{n-1}}{n} + \frac{1}{(n-1)!} \qquad \text{for } n \geq 1
\end{aligned}
$$

**Solution**

Plugging $y_n = 1/(n-1)!$ and $x_n = 1/n$ into the formula known from the lecture yields:

$$
\begin{aligned}
a_n &= \frac{1}{(n-1)!} + \sum_{j=1}^{n-1} \frac{1}{(j-1)!} \frac{1}{j+1} \cdots \frac{1}{n} \\
&= \frac{1}{(n-1)!} + \sum_{j=1}^{n-1} \frac{j}{n!} \\
&= \frac{1}{(n-1)!} + \frac{1}{n!} \frac{n(n-1)}{2}
\end{aligned}
$$