RWTH Aachen
Theoretical Computer Science
Rossmanith–Gehnen

WS 2025
Exercise 6
4.12.2025

Exercise for Analysis of Algorithms

Exercise 21

We want to compare the following two programs for a search in a sorted array:

```
int binsearch(double v) {
  int l,r,m;
  l=1; r=N;
  while (l<=r) {
    m=(r+l)/2;
    if (v==a[m]) return 1;
    if (v<a[m]) r=m-1; else l=m+1;
  }
  return 0;
}
```

```
int binsearch2(double v) {
  int l,r,m;
  l=1; r=N;
  while (r-l>1) {
    m=(r+l)/2;
    if (v<a[m]) r=m-1; else l=m;
  }
  if (a[l]==v) return 1;
  if (a[r]==v) return 1;
  return 0;
}
```

Analyse how many **if**-instructions are executed by the programs in case of a successful or unsuccessful search for v. Find an exact solution for the first program and an estimate of the form $f(n) + O(1)$ for the second one. Make the usual assumptions about v.

Solution:

The first program has already been handled, and we know the sizes $C^- = \lfloor \log(n+1) \rfloor + 2 - 2^{1-\{\log(n+1)\}}$ and $C^+ = \ldots$.

At an unsuccessful search, there are $2C^-$ **if**-instructions because the body of the **while**-loop is executed C^- times, containing two **if**-instructions. In a successful search, however, we have $2C^+ - 1$ **if**-instructions because the second **if**-instruction is not executed in the last iteration.

Now, let's examine *binsearch2*. Let B_n indicate the number of executed **if**-instructions when the observed part of the array still contains $n = r - l + 1$ elements. If fewer than three elements are observed, the **while**-loop is skipped immediately. Thus, for an unsuccessful search, $B_1 = B_2 = 2$. For larger n, the body of the **while**-loop is entered, and an **if**-instruction is executed. If $a[m] > v$, we continue searching on the left; otherwise, on the right. The partial array to be examined becomes smaller, leading to a size of either $\lfloor n/2 \rfloor$ or $\lceil n/2 \rceil$. We establish the following two recursion equations:

$$\overline{B}_n = \overline{B}_{\lceil n/2 \rceil} + 1$$
$$\underline{B}_n = \underline{B}_{\lfloor n/2 \rfloor} + 1$$

for $n > 2$ and $\overline{B}_n = \underline{B}_n = 2$ for $n \leq 2$. Then

$$\underline{B}_n \leq B_n \leq \overline{B}_n$$

certainly holds. We can easily find compact forms for $\underline{B}$ and $\overline{B}$. For $k > 0$, $\underline{B}_{2^k} = k+1$ definitely holds and generally $\underline{B}_n = \lfloor \log(n) \rfloor + 1$ for $n > 2$, $\underline{B}_{2^k+r} = \underline{B}_{2^k}$ for $r < 2^k$.
In the same way $\overline{B}_{2^k} = k+1$ holds for $k > 0$. Here, however, $\overline{B}_{2^k+r} = \overline{B}_{2^k} + 1$ for $0 < r < 2^k$. Therefore $\overline{B}_n = \lceil \log(n) \rceil + 1$ for $n > 2$.
Altogether, we get $B_n = \log(n) + O(1)$.

Exercise 22

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;
```

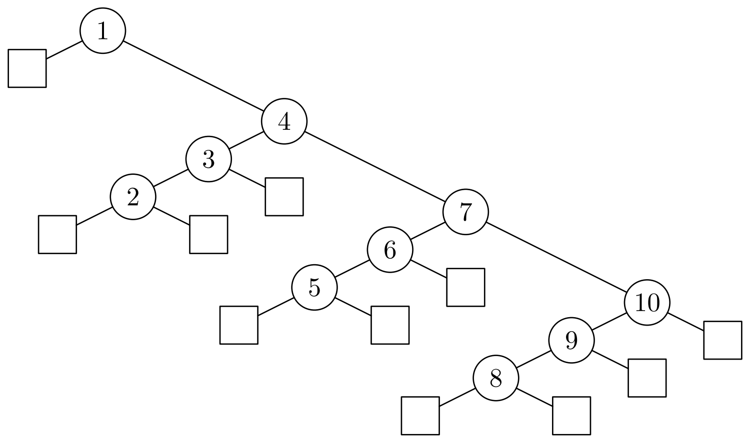
a) Draw the search tree and compute the internal and external path length for $n = 10$ and $m = 3$.

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 π. 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 π 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.

Exercise 23

Compute the generating functions of the following series:

1.	$a_n = 2^n + 3^n$	2.	$b_n = (n+1)2^{n+1}$	3.	$c_n = \alpha^n \binom{k}{n}$
4.	$d_n = n - 1$	5.	$e_n = (n+1)^2$		

Solution:

1. The generating function of (α^n) is $\sum_{n \geq 0} \alpha^n z^n$, which yields $\frac{1}{1-\alpha z}$ in closed form. The generating function of $a_n = 2^n + 3^n$ is thus simply $\frac{1}{1-2z} + \frac{1}{1-3z}$.

2. We start with (2^n) and $\frac{1}{1-2z}$. Derivating yields $b_n = (n+1)2^{n+1}$ with generating function $\frac{2}{(1-2z)^2}$.

3. The series $\binom{k}{n}$ has the generating function $(1+z)^k$. Scaling with α results in $c_n = \alpha^n \binom{k}{n}$ with corresponding generating function $(1 + \alpha z)^k$.

4. We already know that the series $(n+1) = 1, 2, 3, 4, \ldots$ belongs to the generating function $\frac{1}{(1-z)^2}$. In order to obtain $d_n = -1, 0, 1, 2, 3, \ldots$, we first shift this twice to the right. This yields $0, 0, 1, 2, 3, 4, \ldots$ with generating function $\frac{z^2}{(1-z)^2}$. Now we subtract $1, 0, 0, \ldots$ and obtain d_n with generating function $\frac{z^2}{(1-z)^2} - 1$.

5. Recall that $(n+1) = 1, 2, 3, 4, \ldots$ has the generating function $\frac{1}{(1-z)^2}$. We shift to the right and obtain (n) as well as $\frac{z}{(1-z)^2}$. Derivating yields the desired series $e_n = (n+1)^2$ with generating function $\frac{z+1}{(1-z)^3}$.