Analysis of Algorithms, WS 2020
Prof. Dr. P. Rossmanith
Dr. E. Burjons, H. Lotze, D. Mock

Date: 25 November 2020

Exercise Sheet with solutions 05

Problem T11

Let $x \in \mathbf{R}^+$. Is $\lceil \sqrt{x} \rceil = \left\lceil \sqrt{\lceil x \rceil} \right\rceil$?

Solution

Let $i \in \mathbf{N}$, s.t. $i^2 < x \leq (i+1)^2$. Then

$$i^2 < x \leq \lceil x \rceil \leq (i+1)^2.$$

Using monotonicity of $\sqrt{\cdot}$ we get

$$i = \sqrt{i^2} < \sqrt{x} \leq \sqrt{\lceil x \rceil} \leq \sqrt{(i+1)^2} = i+1$$

This means that both $\sqrt{x}$ and $\sqrt{\lceil x \rceil}$ are strictly greater than i and smaller than $i+1$. Since $i \in \mathbf{N}$ we can conclude that

$$\lceil \sqrt{x} \rceil = \left\lceil \sqrt{\lceil x \rceil} \right\rceil = i+1.$$

Problem T12

Use summation factors to solve the following recurrence:

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

Solution

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

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

Problem T13

Consider the following program:

```
int sel_sort ( int a[], int n ) {
  for ( int i = 0; i < n; ++i ) {
    int min = i;
    for ( int j = i; j < n; ++j ) {
      if ( a[j] < a[min] ) {
        min = j;
      }
    }
    int temp = a[i];
    a[i] = a[min];
    a[min] = temp;
  }
}
```

The input to this program is an array $a[0, \ldots, n-1]$ that contains n pairwise distinct integer keys in random order.

a) Explain how this program sorts the given array.

b) Analyse how often each instruction of the program is executed on average depending on n.

c) There is only one instruction whose analysis is not trivial. Which one is it?

Make a table for small values of n by hand that lists the results for this instruction. Compare the table entries with the results from your closed formula that you obtained in b).

Solution

The algorithm is, as indicated by the name, selection sort: it finds the minimal element of the yet-to-be-sorted part and appends it to the sorted part by exchange.

For part b), the outer loop is execute n times, therefore each statement not in the inner loop is executed that often. The if-statement is executed exactly $\frac{n(n+1)}{2}$ times, which leaves the min $= j$ statement.

The first comparison ($j = i$ to i) always fails, therefore the expected number of executions (cf. Problem T1) then is

$$\sum_{i=0}^{n-1} \sum_{j=i+1}^{n-1} \frac{1}{j-i+1} = \sum_{i=0}^{n-1} \sum_{j=1}^{n-i-1} \frac{1}{j+1} = \sum_{i=0}^{n-1} (H_{n-i} - 1) = (n+1)H_n - 2n.$$

Problem H10 (10 credits)

Solve the following recurrence relation by order reduction:

$$a_0 = 0 \qquad a_1 = 1 \qquad a_{n+2} + a_{n+1} - n^2 a_n = n!$$

This is the same recurrence relation as in T10, but the initial conditions are different. That exercise should be solved by order reduction. Solve this exercise by whatever method you like.

Solution

Noch keine Lösung vorhanden!!!!!!!!

Problem H11 (15 credits)
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;
```

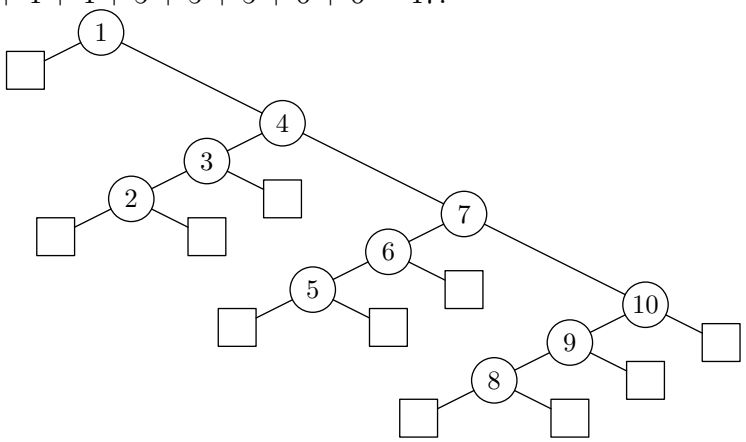
a) Draw the search tree and compute the internal and external path length for $n = 10$ and $m = 3$.

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.

Problem H12 (15 credits)
Generalize the question in T11 to functions that are monotonically increasing, continuous, and do not map non-integers to integers.

Solution
Let $x \in \mathbf{R}$. Given a function f that is monotonically increasing, continuous and does not map nonintegers to integers we want to know if

$$\lceil f(x) \rceil = \left\lceil f\big(\lceil x \rceil\big) \right\rceil .$$

Consider the largest integer i such that $i < f(x) \leq i + 1$. Taking into consideration that f does not map non-integers to integers, we know that there exist two integers k_1 and k_2 such that $f(k_1) = i$ and $f(k_2) = i + 1$. Because f is monotonically increasing we also know that $k_1 < x \leq k_2$ and in particular, $k_1 < x \leq \lceil x \rceil \leq k_2$, which means that if we apply f to every member of this inequality, the inequality is respected by the monotonicity and continuity of f and we see that

$$i = f(k_1) < f(x) \leq f\big(\lceil x \rceil\big) \leq f(k_2) = i + 1 .$$

So both $f(x)$ and $f(\lceil x \rceil)$ are strictly bigger than i but smaller than $i + 1$, which means that

$$\lceil f(x) \rceil = \left\lceil f\big(\lceil x \rceil\big) \right\rceil .$$