RWTH Aachen
Lehrgebiet Theoretische Informatik
Reidl—Rossmanith—Sanchez—Sikdar

SS 2014
Problem Set 7
02.06.2014

Analysis of Algorithms

Problem 7-1

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 ( int j = i - 1; j >= max( 1, i - (m - 1) ); --j){
  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?

Problem 7-2

Consider the following two programs for searching elements in ordered arrays: Determine the

```
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;
}
```

number of executions of **if**-statements in both problems when searching for an element v, in case of both, the successful and unsuccessful search.

Please give an exact solution for *binsearch* and an estimation of the form $f(n) + O(1)$ for for *binsearch2*.

Prerequisites:

- The array contains n *different* elements.
- For the successful search, each element is searched for with equal probability.
- For the unsuccessful search, v is chosen randomly, s.t., with probability $\frac{1}{n+1}$ is "in" one of the $n+1$ possible gaps.

Solution:
The first program was handled in the lecture. We therefore know the values $C^- = \lfloor \log(n+1) \rfloor + 2 - 2^{1-\{\log(n+1)\}}$ and $C^+ = \ldots$.
In an unsuccessful search, $2C^-$ `if`-instructions are executed, since there are C^- runs and each run contains two `if`-instructions. In an successful search, we require $2C^+ - 1$ `if`-instructions, because the second `if`-instructions in the last run is not reached anymore.
Let us now consider `binsearch2`. For an array with n elements, let B_n denote the average number of `if`-instructions executed. If $n < 3$, the `while` is not entered at all. In the case of an unsuccessful search we therefore obtain $B_1 = B_2 = 2$. For a successful search, $B_1 = 1$, but $B_2 = \frac{1}{2}(1+2) = \frac{3}{2}$.
For $n \geq 3$, the `while`-loop is entered and one `if`-instruction is used per iteration of the `while`. If $a[m] > v$, the algorithm searches on the left of the current element, and otherwise on the right. The remaining array thus becomes shorter, either $\lfloor n/2 \rfloor$ or $\lceil n/2 \rceil$. This gives us two recurrences:

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

for $n \geq 3$.

Homework Assignment 7-1 (10 Points)

Consider the following algorithms for searching an element x in an ordered array a of length n. Here, m, is some fixed, but known integer.

```
int search( int x ) {
  int l, r, i;
  l = 1;
  r = n;
  while( r - l >= m ) {
    i = ( l + r ) / 2;
    if ( a[i] == x ) return 1;
    if ( x < a[i])  r = i - 1; else l = i + 1;
  }
  for( i = l; i <= r; ++i ) {
    if ( a[i] == x ) return 1;
    if ( a[i] < x ) return 0;
  }
  return 0;
}
```

Draw the search tree and compute internal and external path lengths for $n = 17$ and $m = 3$.

Homework Assignment 7-2 (10 Points)

Complete the analysis of the average number of times the `if`-instructions are executed in `binsearch2` for both a successful and an unsuccesful search (see Problem 7-2). First show that for all $k \geq 0$, the following hold:

1. $\left\lfloor \left\lfloor n/2^k \right\rfloor /2 \right\rfloor = \left\lfloor n/2^{k+1} \right\rfloor$.
2. $\left\lceil \left\lceil n/2^k \right\rceil /2 \right\rceil = \left\lceil n/2^{k+1} \right\rceil$.

Next, give a detailed analysis of the recurrences:

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