Gads aflevering 1

Thomas Vinther & Jens Kristian Refsgaard Nielsen

07-09-2018

Assume we have an infinite long sorted sequence of integers $x_1 < x_2 < x_3 < \cdots < x_{d-1} < x_d < x_{d+1} < x_{d+2} < \cdots$ and we want to find the position of an integer y in this list, i.e. we want to find the index d, such that $y = x_d$ if y is contained in the list, or otherwise if y is not in the list the successor of y in the list, e.g. $x_{d-1} < y \leq x_d$ (here we assume $x_0 = -\infty$ if $y < x_1$).

Subtask 1

In the original transcript of the excercise it wasn't specified that we were considering an integer sequence, so we went a bit overboard. Let
binSearch(double x, ArrayList<Double> T, int p, int r)
denote binary search of the array T for the element x in the interval [p,r] in $\log_2(r-p)$ time, returning the index m such that $T[m-1] < x \leq T[m]$.
Now let $X = \{x_j \in \mathbb{R} : j \in \mathbb{N} \setminus \{0\}\} \cup \{x_0 = -\infty\}$ be our infinite sequence of reals, with negative infinity. Consider then the following algorithms.

```
01. public int infSearchR(double y, ArrayList<Double> X) {
02.     if (y ≤ X.get(1)){
03.         return 1;
04.     }
05.     int n = Math.max((int) Math.abs(y)+1, 1000);
06.     int i = 0;
07.     while (true){
08.         if (X.get(i * n + 1) ≤ y && y ≤ X.get(n * (i + 1))){
09.             return binSearch(y,X,i*n+1,n*(i+1));
10.         }
11.         i++;
12.     }
13.}
14.
15. public int infSearchZ(double y, ArrayList<Integer> X) {
16.     if (y ≤ X.get(1)){
17.         return 1;
18.     }
19.     int n = (int) Math.abs(y)+1;
20.     int m = binSearch(0,X,1,Math.abs(X.get(1))); \\ finds the index m such that x_m ≥ 0
```

```
21.    if (y ≤ 0){
22.        return binSearch(y,X,1,m);
23.    }
24.    else {
25.        return binSearch(y,X,m,m+n)
26.    }
27.}
```

Subtask 2

The infSearchR algorithm works whenever $X \subset \mathbb{R}$ and the sequence is unbounded, because if it is bounded, for example $x_m = \sum_{j=1}^{m} \frac{1}{j^2}$ then $\lim_{m\to\infty} x_m = \sum_{j=1}^{\infty} \frac{1}{j^2} = \frac{\pi^2}{6} \approx 1.6$ it is clearly an increasing sequence since $x_m = x_{m-1} + \frac{1}{m^2}$ however if we call infSearchR(2,X) it will never terminate.
If X is unbounded then there will be an index $d < \infty$ that solves our problem, now a way to find d is to search for y in the intervals [in+1,n(i+1)] for $i = 1..t$, where $t = \frac{d}{n}$ which can be very big, however our somewhat smart choice of n ensures that if y is very large, the intervals we consider will also be very large hopefully shortening the run time considerably. When we have found the proper interval we do a binary search inside it, and we know from the lectures that binary search will find the proper index d, which we then return.

Now if $X \subset \mathbb{Z}$ the condition that it is sorted and strictly increasing ensures that it is also unbounded, there is nothing wrong with using infSearchR in this case aswell, but infSearchZ is probably faster, depending on the sequence. A sequence of strictly increasing integers will grow by atleast 1 every step, so say x_1 is -100 we clearly get that $x_{100} \geq 0$. We use this fact in line 20. to find the first m such that $x_m \geq 0$, now if y is negative and greater than x_1 it simply must be found within the interval $[x_1, x_m]$. If y is not found there we look in the interval $[x_m, x_{m+n}] \subset \mathbb{N}$ now we can use a similar argument to see that $y \leq x_n = x_{\lceil |y| \rceil}$ so we will find the index we are searching for in line 25. This concludes all the possible cases.

Subtask 3

We wish to consider the run times.
infSearchR makes 1 comparison in line 02. and $\lceil \frac{d}{n} \rceil$ run throughs of the while loop in line 07. resulting in $\lceil \frac{d}{n} \rceil$ comparisons in line 08., giving at worst $\lceil \frac{d}{n} \rceil + \log_2(n)$ comparisons, where the $\log_2(n)$ is from binary search.
infSearchZ also makes the initial comparison in line 16. then makes $\log_2(x_1)$ comparisons in line 20. to find m, then either $\log_2(m)$ or $\log_2(n)$ comparisons in lines 21. through 26. in total the worst case runtime will be

$$wR(\text{infSearchR}) = 1 + \left\lceil \frac{d}{n} \right\rceil + \log_2(n) \tag{0.1}$$

$$wR(\text{infSearchZ}) = 1 + \log_2(m) + \max\{\log_2(m), \log_2(n)\} \tag{0.2}$$