Homework 9 sample solution

Due 10/05/16

September 30, 2016

1. Describe a divide-and-conquer algorithm that accepts a positive integer n and computes $\lfloor \lg n \rfloor$ (that is, the largest integer x such that $2^x \leq n$). Your algorithm should take $O(\lg(\lg n))$ time.

   Hint: you may wish to base your approach on one-sided binary search, which starts at 1, doubles the value until it becomes too large, then performs binary search between the last value that worked and the first value that failed. (Divide-and-conquer algorithms aren't required to be recursive.) You may assume that the square root function takes $\Theta(1)$ time, though there is an $O(\lg \lg n)$ algorithm that does not use the square root.

   Answer: Algorithm that uses $\sqrt{n}$:

```
   Input: n: positive integer
   Output: ⌊lg n⌋
 1 Algorithm: LogSearch
 2 x = 1
 3 ceil = 2
   /* Loop invariant:  ceil = 2^x                                   */
 4 while ceil ≤ n do
 5 |   ceil = ceil^2
 6 |   x = 2x
 7 end
 8 floor = √ceil
 9 delta = √floor
10 x = x/2
11 d = x/2
   /* Main idea:  x = low, x + 2d = high, x + d = mid, and divide d
      by 2 each iteration                                           */
   /* Loop invariants:  floor = 2^x, delta = 2^d, and 2^x ≤ n < 2^(x+2d)
      */
12 while d ≥ 1 do
13 |   if floor · delta ≤ n then
14 |   |   floor = floor · delta
15 |   |   x = x + d
16 |   end
17 |   delta = √delta
18 |   d = d/2
19 end
20 return x
```

Algorithm that doesn't use $\sqrt{n}$:

```
   Input: n: positive integer
   Output: ⌊lg n⌋
 1 Algorithm: NoRootLogSearch
   /* Main idea:  use squares in reverse order instead of square
      root                                                        */
 2 pow = Stack()
 3 ceil = 2
 4 x = 1
 5 while ceil ≤ n do
 6 |   pow.push(ceil)
 7 |   ceil = ceil^2
 8 |   x = 2x
 9 end
10 floor = pow.pop()
11 delta = pow.pop()
12 x = x/2
13 d = x/2
14 while d ≥ 1 do
15 |   if floor · delta ≤ n then
16 |   |   floor = floor · delta
17 |   |   x = x + d
18 |   end
19 |   if ¬pow.empty() then
20 |   |   delta = pow.pop()
21 |   end
22 |   d = d/2
23 end
24 return x
```