Chapter 4 Exercises

Termanteus

2019-08-06

Contents

1. R-4.1

Statement Describe a recursive algorithm for finding the maximum element in a sequence, S, of n elements. What is your running time and space usage?

Solution Both running time and space usage are O(n)

Algorithm 1: Recursive algorithm for finding maximum

```
function findMax (S, n);
Input  : Array of numbers, n
Output: Maximum value from 0 to n in the array S
if n == 1 then
    return S[n − 1];
end
currentMax ⟵ findMax(S, n − 1);
if currentMax > S[n − 1] then
    return currentMax;
else
    return S[n − 1];
end
```

2. R-4.2

Statement Describe a recursive algorithm for finding the maximum element in a sequence, S, of n elements. What is your running time and space usage?

Solution Picture

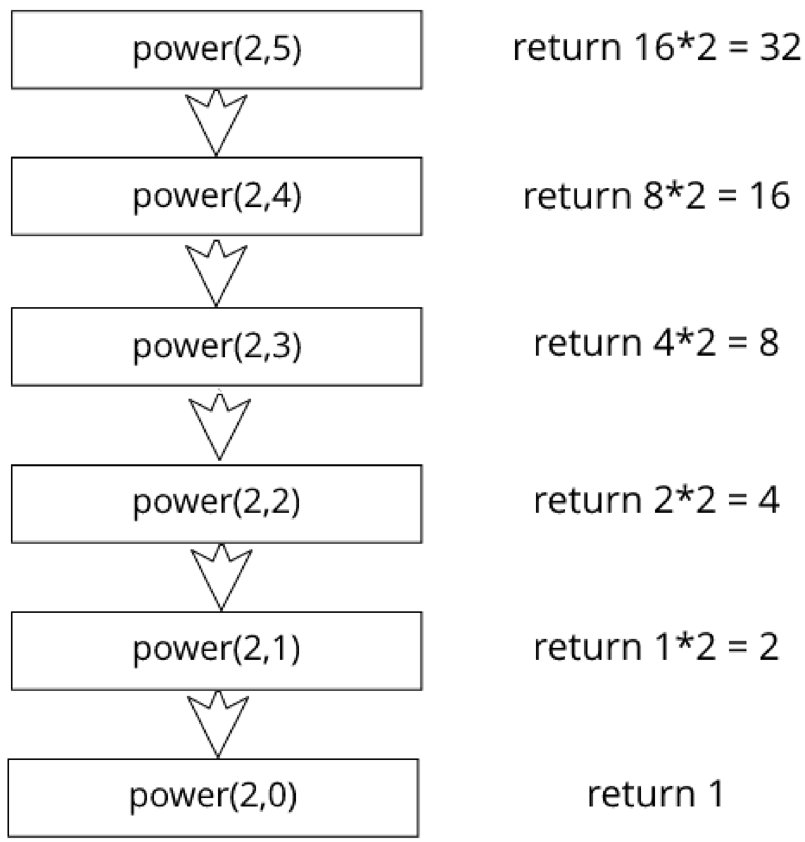

Figure 1: Power(2,5) of Code Fragment 4.11

3. R-4.3

Skip because of the drawing...

4. R-4.4

Skip because of the drawing...

5. R-4.5

Skip because of the drawing...

6. R-4.6

Statement Describe a recursive function for computing the n^{th} Harmonic number $H_n = \sum_{i=1}^{n} \frac{1}{n}$.

Solution Pseudocode

Algorithm 2: Compute the n-th **Harmonic number**

function harmonicNumber (n);
Input : Index n
Output: The n-th value in Harmonic Number
if $n == 1$ **then**
  return 1;
end
return $\frac{1}{n} + harmonicNumber(n-1)$;

7. R-4.7

Statement Describe a recursive function for converting a string of digits into the integer it represents. For example, 13531 represents the integer 13,531.

Solution Pseudocode

Algorithm 3: Convert a string of digits into the integer it represents

function toInt ($string, i$);
Input : String of digits, current index
Output: Number of the string represents until character i
if $i == 1$ **then** **return** *int(string[i])* ;
if $i == 0$ **then** **return**;
return $toInt(string, i-1) * 10 + int(string[i])$;

8. R-4.8

Statement Isabel has an interesting way of summing up the values in a sequence A of n integers, where n is a power of two. She creates a new sequence B of half the size of A and sets $B[i] = A[2i] + A[2i+1]$, for $i = 0, 1, ..., (n/2) - 1$. If B has size 1, then she outputs $B[0]$. Otherwise, she replaces A with B, and repeats the process. What is the running time of her algorithm?

Solution **Total Running time:** O(n)

There're total of $log_2 n$ recursive calls. But in the first run, this algorithm has already had to make n/2 = O(n) sums, that also the cost for creating sequence B or re-set A = B.

9. C-4.13

Statement In Section 4.2 we prove by induction that the number of lines printed by a call to draw_interval(c) is $2^c - 1$. Another interesting question is how many dashes are printed during that process. Prove by induction that the number of dashes printed by draw_interval(c) is $2^{c+1} - c - 2$.

Solution P(c): The number of dashes printed by draw_interval(c) is $2^{c+1} - c - 2$.

Base case: P(1): 1 dash is drawn. Correct.

Inductive step: Assume P(n) is True. Prove P(n+1) true.

draw_interval(n+1) will call draw_interval 2 times and draw_line(n+1) 1 time.

For each draw_interval(n): $2^{n+1} - n - 2$ dashes is drawn.

For draw_line(n+1): $n + 1$ dashes is drawn.

Total: $2^{n+1} - n - 2 + 2^{n+1} - n - 2 + n + 1 = 2^{c+2} - c - 3$. P(n+1) holds.

Proved.