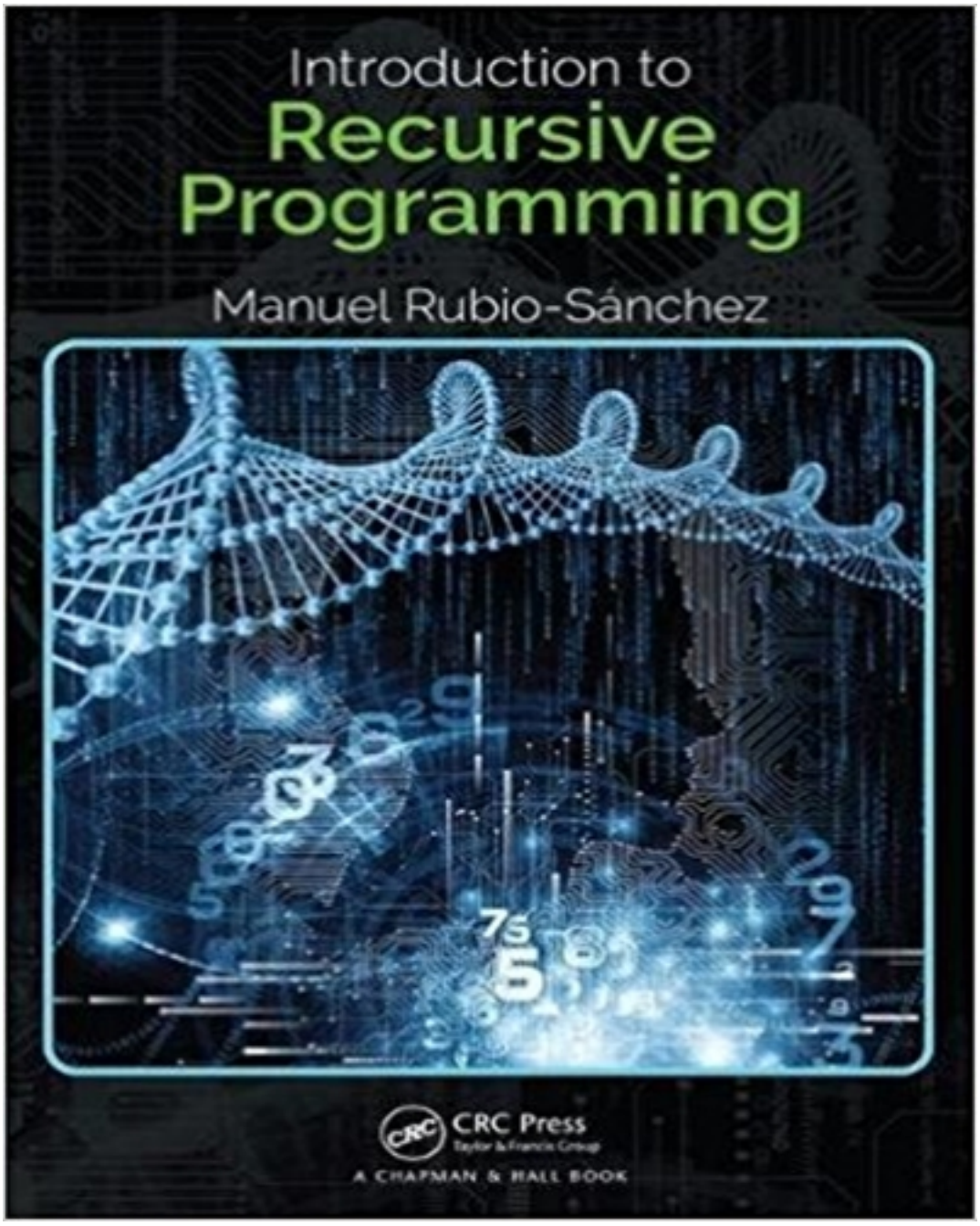
Solutions for Introduction to Recursive Programming 1st Edition by Rubio Sanchez

CLICK HERE TO ACCESS COMPLETE Solutions



Solutions

A.2 CHAPTER 2

Exercise 2.1 — The size of the problem is the total number of bits of the binary representation of n (regardless of how many are set to 1, since all of them have to be analyzed by the algorithm). In particular, the quantity can be expressed formally as:

$$\lfloor \log_2(n) \rfloor + 1.$$

Exercise 2.2 — The more general function is:

$$S(n) = \begin{cases} 0 & \text{if } n = 0, \\ S(n-1) + n & \text{if } n > 0. \end{cases}$$

The base case $S(1) = 1$ is no longer necessary. The function can be coded as in Listing 9.

Exercise 2.3 — Possible diagrams are shown in Figure 1. In (a) four rectangular structures associated with subproblems of size $n/2$ cover the area related to a problem of size n. However, since $n/2$ single square blocks fall outside of the area, we need to subtract $n/2$ in the recursive formula. In (b) four subproblems of size $(n-1)/2$ cover the area of the original problem except for $(n+1)/2$ blocks, which need to be added in the corresponding recursive case.

The more general function is:

$$S(n) = \begin{cases} 1 & \text{if } n = 1, \\ 4S(\frac{n}{2}) - \frac{n}{2} & \text{if } n > 0 \text{ and } n \text{ even}, \\ 4S(\frac{n-1}{2}) + \frac{n+1}{2} & \text{if } n > 0 \text{ and } n \text{ odd}. \end{cases}$$

The function can be coded as shown in Listing 10.

Listing A.9 Sum of the first nonnegative integers.

```
def sum_first_nonnegative_integers(n):
    if n == 0:
        return 0
    else:
        return sum_first_nonnegative_integers(n - 1) + n
```

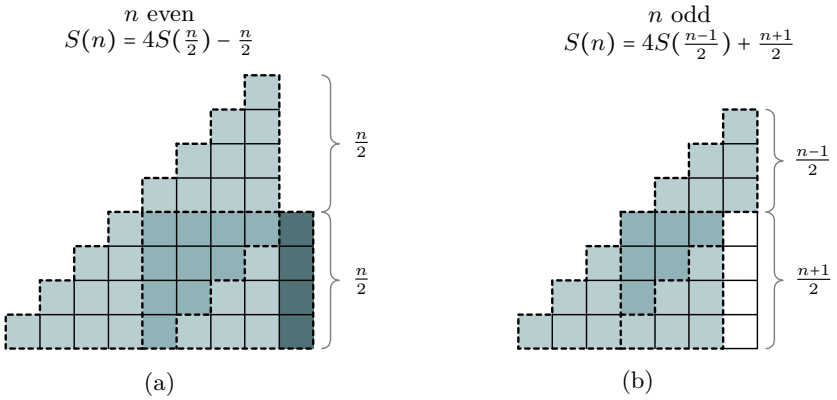
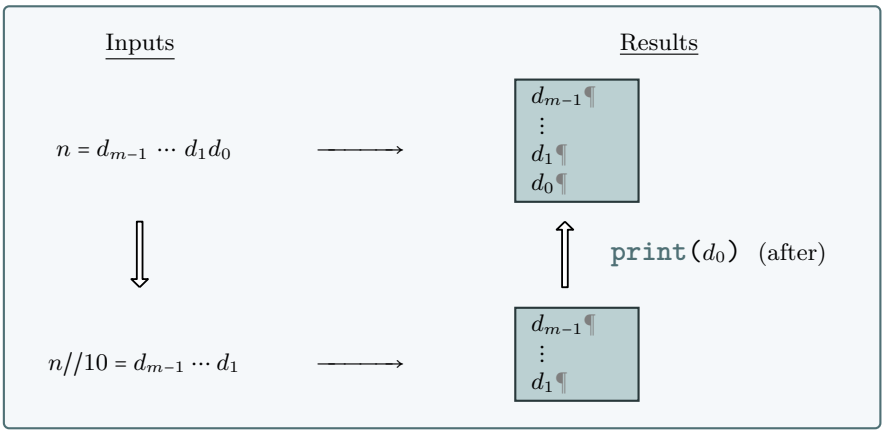


Figure A.1 Diagram showing a decomposition of the sum of the first n positive integers $S(n)$ that uses four subproblems of (roughly) half the size as the original.

Exercise 2.4 — The size of this problem is the number of digits of n. The base case occurs when n contains a single digit ($n < 10$), where the method simply prints n. The general diagram can be the following:



Finally, the recursive procedure can be coded as illustrated in Listing 11.

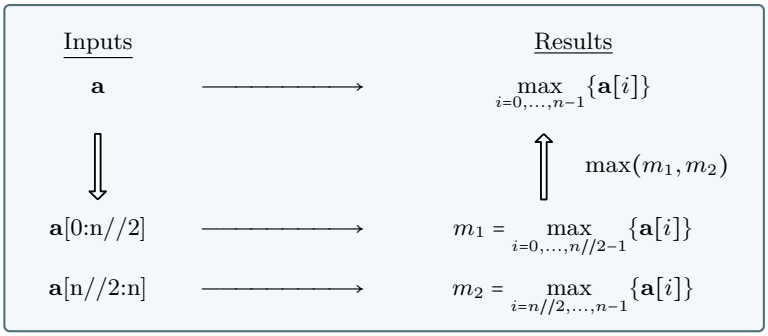
Exercise 2.5 — For a general list **a** of length n the diagram can be defined as follows:

Listing A.10 Sum of the first nonnegative integers, by decomposing the problem into four subproblems.

```
def sum_first_positives_x4(n):
    if n == 1:
        return 1
    elif n % 2 == 0:
        return 4 * sum_first_positives_x4(n / 2) - n / 2
    else:
        return (4 * sum_first_positives_x4((n - 1) / 2)
                + (n + 1) / 2)
```

Listing A.11 Procedure for printing the digits of a number vertically.

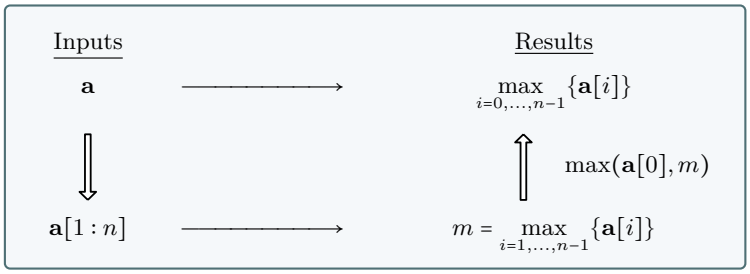
```
def print_digits_vertically(n):
    if n<10:
        print(n)
    else:
        print_digits_vertically(n//10)
        print(n%10)
```



Exercise 2.6 — Firstly, the size of this problem is n, and the base case is reached when the list only contains one element. The proposed decomposition requires discarding one element of the list, which could be either the first or the last. The following general diagram considers the subproblem that does not include the first element of the original list:

Listing A.12 Function that computes the maximum element of a nonempty list.

```
def max_list_length_first(a):
    if len(a) == 1:
        return a[0]
    else:
        return max(a[0], max_list_length_first(a[1:]))
```



The recursive case therefore needs to check whether the maximum value is the individual element ($\mathbf{a}[0]$), or if it is included in the sublist $\mathbf{a}[1:n]$, or in both (note that the largest value could appear several times in the list). Finally, the function can be coded as shown in Listing 12.