

Algorithm analysis: sample problems

The problems below, with solutions, are models to help solving problems in assignments 3 through 4

1. Sum problem

Problem: Write a recurrence and perform an analysis for the following algorithm:

Sum(A)

```
i ← 1
y ← 0
while i ≤ |A|
  y ← y + A[i]
  i ← i + 1
return y
```

Recurrence:

$$\text{Sum}(A) = \begin{cases} 0 & \text{if } |A| = 0 \\ A[1] & \text{if } |A| = 1 \\ A[1] + \text{Sum}(A[2 \dots |A|]) & \text{otherwise} \end{cases}$$

Time recurrence:

$$T_{\text{Sum}}(n) = \begin{cases} O(1) & \text{if } n \leq 1 \\ O(1) + T_{\text{Sum}}(n-1) & \text{otherwise} \end{cases} \\ = O(n)$$

2. Last-zero problem

Problem: Find the location of the rightmost zero in a series of zeroes followed by a series of ones.

a. Recurrence suggesting linear-time algorithm:

$$\text{Last0}(A) = \begin{cases} \uparrow & \text{if } |A| = 0 \text{ or } A[1] = 1 \\ 1 + \text{Last0}(A[2 \dots |A|]) & \text{otherwise} \end{cases}$$

Time recurrence:

$$T_{\text{Last0}}(n) = \begin{cases} O(1) & \text{if } n \leq 1 \\ O(1) + T_{\text{Last0}}(n-1) & \text{otherwise} \end{cases} \\ = O(n)$$

The algorithm corresponding to this recurrence is $O(n)$, because the depth of recursion is $O(n)$, and at each recursive step, evaluating the base case requires $O(1)$ time.

b. Divide-and-conquer algorithm:

Last0-R(A)

```
// Pre: |A| > 0 and no 0 follows any '1'
if A[|A|] = '0'
  return |A|
mid ← ⌊|A| ÷ 2⌋
if A[mid] = '0'
  y ← mid + Last0-R(A[mid + 1 ... |A|])
else
  y ← Last0-R(A[1 .. mid - 1])
return y
// Post: y stores rightmost 0 in A
```

Time recurrence:

$$T_{\text{Last0-R}}(n) = \begin{cases} O(1) & \text{if } n \leq 1 \\ O(1) + T_{\text{Last0-R}}(n \div 2) & \text{otherwise} \end{cases} \\ = O(\lg n)$$

The running time is $O(\lg n)$, because the depth of recursion is $O(\lg n)$, and at each recursive step, evaluating the base case requires $O(1)$ time. The depth of recursion is proportional to the number of times necessary to divide n by 2 before reaching 1.