5.4 Recursive Algorithms

An algorithm is called *recursive* if it solves a problem by reducing it to an instance of the same problem with smaller input.

5.4 pg 370 # 3

Trace Algorithm 3 when it finds gcd(8,13). That is, show all the steps used by Algorithm 3 to find gcd(8,13).

Algorithm 3 1 *gcd*(a, b : nonnegative integers with $a < b$)

if $a = 0$ **then**
  return b
else
  return *gcd*($b \mod a, a$)
end if
{output is gcd(a, b)}

gcd(8,13)
  gcd($13 \mod 8 = \mathbf{5, 8}$)
    gcd($8 \mod 5 = \mathbf{3, 5}$)
      gcd($5 \mod 3 = \mathbf{2, 3}$)
        gcd($3 \mod 2 = \mathbf{1, 2}$)
          gcd($2 \mod 1 = \mathbf{0, 1}$)
            return 1

5.4 pg 370 # 7

Give a recursive algorithm for computing nx whenever n is a positive integer and x is an integer, using just addition.

Procedure 2 *product*(n : positive integer, x : integer)

if $n = 1$ **then**
  return x
else
  return $x +$ *product*($n - 1, x$)
end if
{output is nx}

5.4 pg 370 # 9

Give a recursive algorithm for finding the sum of the first n odd positive integers.

Procedure 3 *oddSum*(n : positive integer)

if $n = 1$ **then**
  return 1
else
  return $2(n-1) + 1 + oddSum(n-1)$
end if
{output is sum for first n odd positive integers}

5.4 pg 370 # 11

Give a recursive algorithm for finding the minimum of a finite set of integers, making use of the fact that the minimum of n integers is the smaller of the last integer in the list and the minimum of the first $n-1$ integers in the list.

Procedure 4 *recursive_min*(n : positive integer, $a_1, a_2, a_3, \ldots, a_n$: integers)

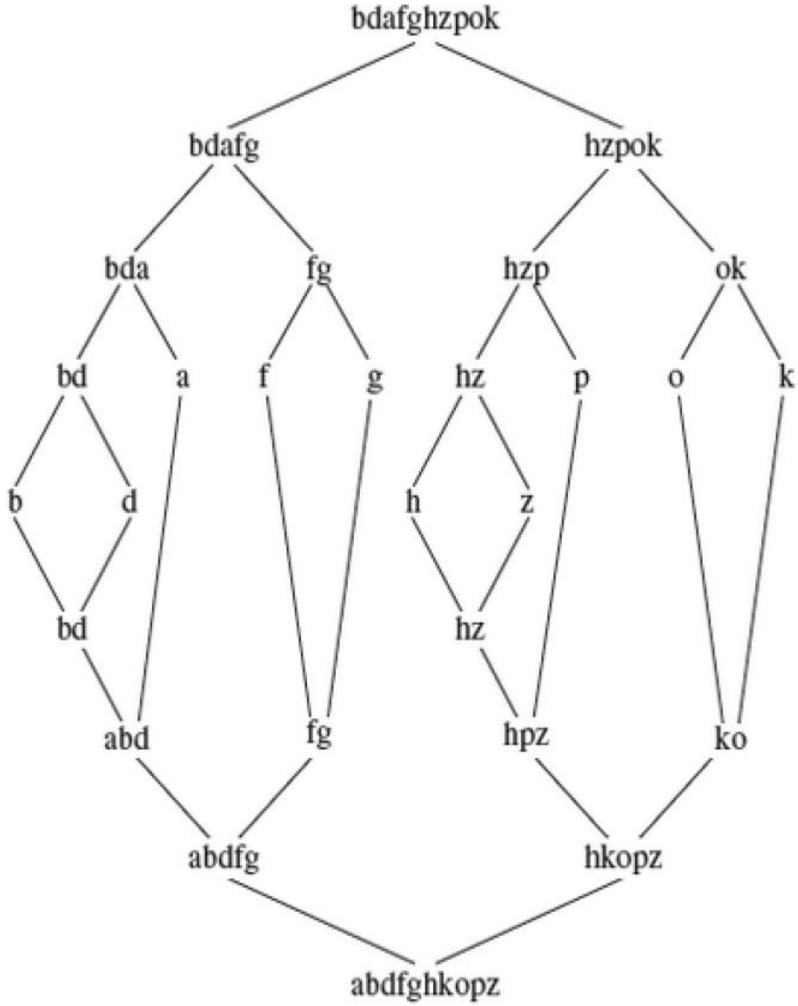
if $n = 1$ **then**
  return a_1
else
  return *min*(a_n, *recursive_min*($n-1, a_1, a_2, a_3, \ldots, a_{n-1}$))
end if
{output is the minimum integer}

5.4 pg 371 # 45

Use a merge sort to sort $b, d, a, f, g, h, z, p, o, k$ into alphabetic order. Show all the steps used by the algorithm

Procedure 5 *mergesort*($L = a_1, \ldots a_n$)

if $n > 1$ **then**
  $m := \lceil n/2 \rceil$
  $L_1 := a_1, a_2, \ldots, a_m$
  $L_2 := a_{m+1}, a_{m+2}, \ldots, a_n$
  $L :=$ *merge*(*mergesort*(L_1), *mergesort*(L_2))
end if
{L is now sorted into elements in nondecreasing order}


bdafghzpok
bdafg
hzpok
bda
fg
hzp
ok
bd
a
f
g
hz
p
o
k
b
d
h
z
bd
hz
abd
fg
hpz
ko
abdfg
hkopz
abdfghkopz