Theoretical Computer Science – Exercise 10

SS 2022
Jochen Schmidt

Technische Hochschule Rosenheim

Exercise 2

b) A(2,3) = 2,3 = 1,2,2 = 1,1,2,1 = 1,1,1,2,0

= 1,1,1,1,1
= 1,1,1,0,1,0
= 1,1,1,0,0,1
= 1,1,1,0,2
= 1,1,1,3
= 1,1,0,1,2
= 1,1,0,0,1,1
= 1,1,0,0,0,1,0
= 1,1,0,0,0,0,1
= 1,1,0,0,0,2
= 1,1,0,0,3
= 1,1,0,4
= 1,1,5
= 1,0,1,4
= 1,0,0,1,3
= 1,0,0,0,1,2
= 1,0,0,0,0,1,1
= 1,0,0,0,0,0,1,0
= 1,0,0,0,0,0,0,1
= 1,0,0,0,0,0,2
= 1,0,0,0,0,3
= 1,0,0,0,4
= 1,0,0,5
= 1,0,6
= 1,7
= 0,1,6
= 0,0,1,5
= 0,0,0,1,4
= 0,0,0,0,1,3
= 0,0,0,0,0,1,2
= 0,0,0,0,0,0,1,1
= 0,0,0,0,0,0,0,1,0
= 0,0,0,0,0,0,0,0,1
= 0,0,0,0,0,0,0,2
= 0,0,0,0,0,0,3
= 0,0,0,0,0,4
= 0,0,0,0,5
= 0,0,0,6
= 0,0,7
= 0,8
= 9

Exercise 3

```
//*************************************************************************
// Compare recursive and iterative computation
// of Ackermann function
//*************************************************************************
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#define SMAX 100000

//*************************************************************************
// Recursive
//-------------------------------------------------------------------------
int ak_r(int x, int y)
{
   if(x==0) return(y + 1);                          // end of recursion
   if(y==0) return(ak_r(x - 1, 1));                 // single recursion
   return(ak_r(x - 1, ak_r(x, y - 1)));             // double recursion
}

//*************************************************************************
// Iterative
//-------------------------------------------------------------------------
int ak_i(int x, int y)
{
  int k, s[SMAX+2], sp=0;                    // Stack and stack pointer
  s[sp++] = x; s[sp++] = y;                  // put x and y on stack
  while(sp!=1)
  {
    y = s[--sp]; x = s[--sp];                // get x and y from stack
    if(x==0) s[sp++] = y + 1;                // put only y+1 on stack
    else if(y==0)                            // put x-1 and 1 on stack
    {
      s[sp++] = x - 1; s[sp++] = 1;
    }
    else                                     // put x-1, x, and y-1 on stack
    {
      s[sp++] = x - 1;
      s[sp++] = x;
      s[sp++] = y - 1;
    }
    if(sp>=SMAX) return -1;                  // stack overflow
  }
  return s[--sp];                        // result: last remaining stack entry
}

//*************************************************************************
// Main program
//*************************************************************************
int main()
{
  int x=1,y=1, a=0;              // init parameters of Ackermann function
  time_t t;                      // for time measurement
  printf("\n\nACKERMANN FUNCTION\n");
  while(x > 0 || y > 0)          // stop program if either x or y is zero
  {
    printf("\nx, y = ");
    scanf("%d,%d", &x, &y);                        // enter x and y
    t = clock();                                   // remember start time
    printf("\nak_r = %d  ", ak_r(x, y));          // ak(x,y) recursively
    printf("(%5.2f sec)\n",(float)difftime(clock(),t)/CLOCKS_PER_SEC);
    t = clock();                                   // remember start time
    a = ak_i(x, y);                                // ak(x,y) iterativ
    if(a < 0) printf("\nStack overflow!");
    else printf("\nak_i = %d  ",a);
    printf("(%5.2f sec)\n",(float)difftime(clock(),t)/CLOCKS_PER_SEC);
  }
}
```