Aufgabe 3.1

x = 0

for i = 1 to n do

x = x + A[i]

end for

return x

Dieser Algorithmus benoetigt O(n) Operationen,

wobei n die Anzahl Elemente von A ist. Das liegt daran,

dass jedes Element einmal verrechnet werden muss.

for i = 1 to n do | n operationen

A[i] = i

end for

for i = 1 to n do | n mal

C[i] = 0

for j = n downto 1 do

if A[j] > C[i] then | n Operationen = n^2 Operationen

C[i] = A[j]

end if

end for

end for

return C

Dieser Algorithmus benoetigt O(n(n + 1)) Operationen,

da die eine Schleife n^2 Durchlaeufe braucht und

die andere n Durchlaeufe.

for i = 1 to n do

for j = 1 to n do

C[i][j] = 0

for k = 1 to n do

C[i][j] = A[i][k] \* B[k][j]

end for

end for

end for

return C

Dieser Algorithmus benoetigt O() Operationen,

da hier drei Schleifen, die alle n Durchlaeufe

machen, inneinander geschachtelt laufen.

for i = 1 to n do

for j = i downto 1 do

x = x + A[i][j]

end for

end for

return x

Dieser Algorithmus benoetigt O() Operationen,

da die zweite Schleife dem Laufindex der ersten

Schleife entsprechend Durchlaeufe machten muss.