B-it **Bridging Course (SS05)**

Lecturer: L. El Aimani

Assignment 1

1 Solving Recurrences

1. $C_N = C_{\frac{N}{2}} + N$ **for** $N \geq 2$ **with** $C_1 = 0$
   The recurrence is defined when $N = 2^n$,i.e, when N is a power of 2 In fact:

$$\begin{aligned} C_N = C_{\frac{N}{2}} + N &= C_{\frac{N}{4}} + \frac{N}{2} + N \\ &= C_{\frac{N}{2^n}} + \frac{N}{2^{n-1}} + \cdots + \frac{N}{2} + N \\ &= C_1 + N(2(1 - \frac{1}{2^n})) \\ &= 2N \end{aligned}$$

   If the sum $N + \frac{N}{2} + \frac{N}{4} + \frac{N}{8} + \cdots$ is infinite, it evaluates to exactely $2N$. Since we stop at 1, this value is an approximation to the exact answer. The precise solution involves properties of the binary representation of N.

2. $C_N = 2C_{\frac{N}{2}} + N$ **for** $N \geq 2$ **with** $C_1 = 0$
   Again, the recurrence is precisely defined only when N is a power of 2:

$$\begin{aligned} C_{2^n} &= 2C_{2^{n-1}} + 2^n \\ \frac{C_{2^n}}{2^n} &= \frac{C_{2^{n-1}}}{2^{n-1}} + 1 \\ &= \frac{C_{2^{n-2}}}{2^{n-2}} + 1 + 1 \\ &\vdots \\ &= n \end{aligned}$$

   So the recurrence is about $N \log N$

3. $C_N = 2C_{\frac{N}{2}} + 1$ **for** $N \geq 2$ **with** $C_1 = 1$

$$\begin{aligned}
C_N &= 2C_{\frac{N}{2}} + 1 \\
&= 2(2C_{\frac{N}{2}} + 1) + 1 \\
&\vdots \\
&= 2^n C_{\frac{N}{2^n}} + 2^{n-1} + 2^{n-2} + \cdots + 2 + 1 \\
&= 2^n C_1 + 2^{n-1} + 2^{n-2} + \cdots + 2 + 1 \\
&= 2^n + 2^{n-1} + 2^{n-2} + \cdots + 2 + 1 \\
&= 2^{n+1} - 1 \\
&= 2N - 1
\end{aligned}$$

The recurrence evaluates then to $2N$ (when N is a power of 2).

2 Some recursive algorithms

1. • **factorial function (iterative version)**

   Algorithm 1. ***factI(n)***

   (a) *int fact = 1;*

   (b) *int i = 2;*

   (c) *while (i* $\leq$ *n)*
   fact = fact $\times$ *i++;*

   (d) *return fact;*

   • **factorial function (recursive version)**

   Algorithm 2. ***factR(n)***

   (a) *(if n == 0) then return* 1

   (b) *else return factR(n-1)*$\times$ *n;*

   • **Cost**: solving the recurrence $C_N = C_{N-1}$ results in $C_N = N$.

2. • **gcd function (iterative version)**

   Algorithm 3. ***gcdI(a,b)***

   (a) *int r ;*

   (b) *while (b != 0)*
   - *r = a % b;*
   - *a = b ;*
   - *b = r;*

   (c) *return (a);*

- **gcd function (recursive version)**

  Algorithm 4. ***gcdR(a,b)***

  (a) (if b == 0) then return a

  (b) else return gcd(b,a % b);