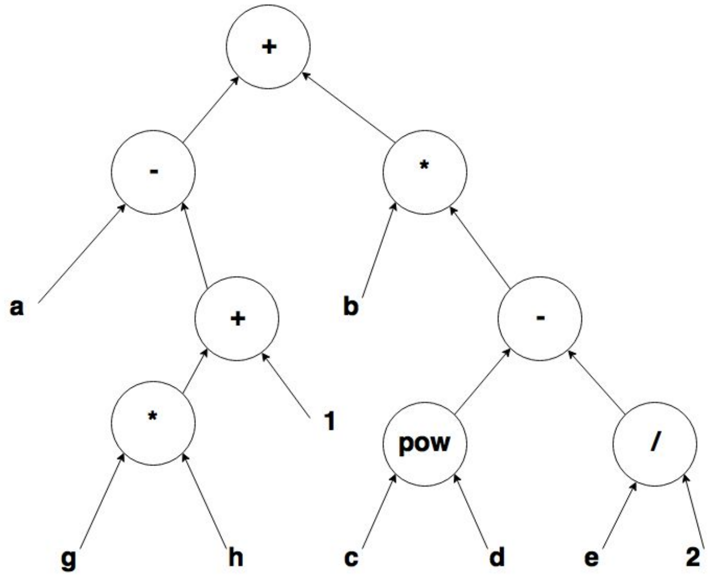
Konstantin Müller (2327697)
Robin Ferrari (2585277)
Vladislav Lasmann (2593078)

Übungsblatt 3

Aufgabe 1

a) DAG **minimaler** Tiefe:



b)
Der DAG hat eine Tiefe von 4, wenn die Blätter nicht mitgezählt werden. Falls diese mitgezählt werden, so hat der DAG eine Tiefe von 5.

c)
Die Ausführungsdauer entspricht der Anzahl der Operationsknoten mal 1ns.
Somit ist die Ausführungsdauer $T^* = 8\,ns$.

d)
Es gilt für die Ausführungszeit mit 2, 3 und 4 Prozessoren:

$T_2 = 5\,ns$ $\quad S_2 = \frac{T^*}{T_2} = \frac{8\,ns}{5\,ns} = 1,6$ $\quad \Rightarrow$ $Speedup\ 60\%$

$T_3 = 4\,ns$ $\quad S_3 = \frac{T^*}{T_3} = \frac{8\,ns}{4\,ns} = 2$ $\quad \Rightarrow$ $Speedup\ 100\%$

$T_4 = 4\,ns$ $\quad S_4 = \frac{T^*}{T_4} = \frac{8\,ns}{4\,ns} = 2$ $\quad \Rightarrow$ $Speedup\ 100\%$

e)
Da der Input von einigen Operationsknoten das Ergebnis von vorher berechneten Operationen ist und es maximal 3 in der gleichen Tiefe gibt, ist der maximal erreichbare Speedup nur mit 3 Prozessoren möglich, da sonst die anderen Prozessoren nicht zur Berechnung genommen werden.

Aufgabe 2

Um das korrekte Ergebnis in log_2(n) Schritten zu bekommen und die CREW-Eigenschaft nicht zu verletzen, muss der Algorithmus einen binärbaumartige Struktur haben.

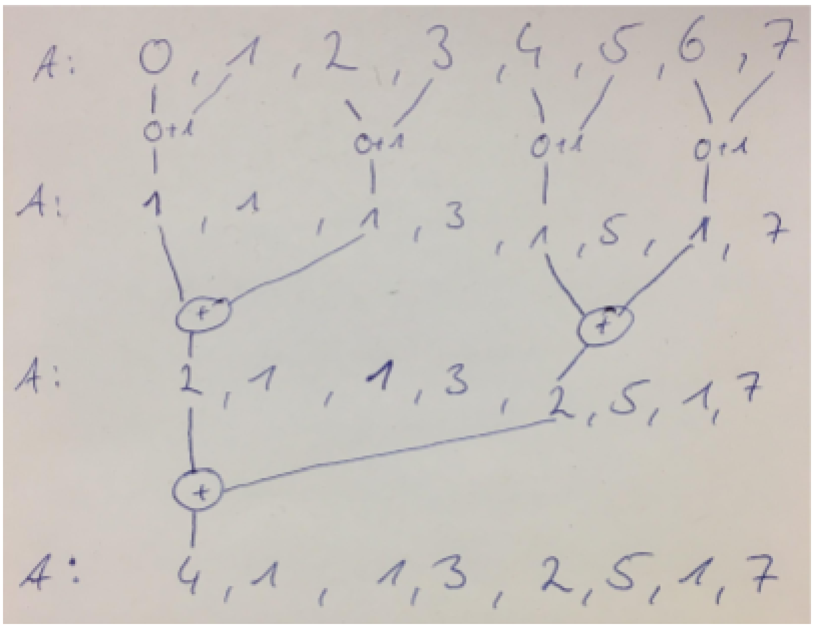
```
INPUT:  Ein Array A der Länge n, in der die Zahlenfolge x_1, x_2, …, x_n gespeicher ist.
        Die Indizes beginnen mit 0. Zudem muss log_2(n) ∈ ℕ gelten.
        Von den n vorhandenen Prozessoren kennt jeder Prozessor seine ID (0, 1, …, n),
        die in pNumber abrufbar ist.
OUTPUT: Nach log_2(n) iterationen ist in A[0] die Anzahl der ungeraden Zahlen der
        Zahlenfolge gespeichert. Es wird nicht garantiert, dass die Zahlenfolge in A
        erhalten bleibt.
ALGORITHMUS:
begin
   for i=1 to log_2(n) do
      if i==1 && pNumber%2 == 0   then
         begin
            result = 0
            number1 = 0
            number2 = 0
            global_read(A(pNumber*2), number1)
            global_read(A(pNumber*2 + 1), number2)

            if (number1%2 != 0) && (number2%2 != 0) then
               result = 2
            else if (number1%2 != 0) || (number2%2 != 0) then
               result = 1

            global_write(result, A(pNumber * 2))
         end
      else if pNumber%2^i == 0    then
         begin
            result = 0
            number1 = 0
            number2 = 0
            global_read(A(pNumber * 2^i), number1)
            global_read(A(pNumber * 2^i + 2^i), number2)
            result = number1 + number2

            global_write(result, A(pNumber * 2^i))
         end
   end
end
```

Veranschaulicht:



Aufgabe 3

Im Folgenden wird die Komplexität für den Best-Case erarbeitet und bewiesen, dass diese schneller wächst als $log\,(n)$.

Best-Case:
Um eine bessere Parallelisierung zu garantieren, wird angenommen, dass das 2D-Gitter quadratisch ist, d.h. bei p Prozessoren ist $\sqrt{p} \in \mathbb{N}$. Um eine schnellere Verarbeitung zu garantieren, wird angenommen dass jedem Prozessor genau eine Zahl einer der n gegebenen Zahlen zugewiesen wird, d.h. $p = n$ gilt $^{(1)}$.
Um die Informationen von einer Seite auf die andere zu addieren & kommunizieren benötigt man $\sqrt{p} - 1$ Schritte. Danach ist die Summe der ganzen Zeile jeweils in einem Prozessor der Zeile gespeichert $^{(1)-(3)}$.
Um die Information von unten nach oben zu addieren benötigt man wieder $\sqrt{p} - 1$ Schritte, bis ein Prozessor die endgültige Summe besitzt $^{(3)-(5)}$.

Somit erhält man folgende Komplexität:
$\Theta(\sqrt{p} - 1) + \Theta(\sqrt{p} - 1) = \Theta(2\sqrt{p} - 2) = \Theta(2\sqrt{n} - 2)$

Behauptung: $O(log\ n)$ wächst langsamer als $\Theta(2\sqrt{n} - 2)$
Beweis:
$$\lim_{n\to\infty} \frac{2\sqrt{n}-2}{log(n)} = \frac{\infty}{\infty} \Rightarrow^{(l'Hospital)} \lim_{n\to\infty} \frac{2\sqrt{n}-2}{log(n)} = \lim_{n\to\infty} \frac{2\frac{1}{2\sqrt{n}}}{\frac{1}{n}} = \lim_{n\to\infty} \frac{\sqrt{n}\,\sqrt{n}}{\sqrt{n}} = \lim_{n\to\infty} \sqrt{n} = \infty$$

(1)

10	10	10
10	10	10
10	10	10

→ (2)

10	20	–
10	20	–
10	20	–

→ (3)

30	–	–
30	–	–
30	–	–

→ (4)

30	–	–
60	–	–
–	–	–

→ (5)

90	–	–
–	–	–
–	–	–

Aufgabe 4