Reduction

Questions

1. 3 Applications of reduction:
   (a) Prefix sum: SAT (Summmed Area Table) which can be used for depth of field blurring.
   (b) Computing single values such as max/min, average, or standard deviation.
   (c) Physics simulations: e.g. N-body simulations where we need to compute the sum of forces acting on a particle.

2. Since the loading into shared memory does one addition per thread, and in the reduction step we have $n = \log_2 \texttt{blockDim.x}$ steps doing one addition times the stride or as a finite geometric sum

$$\begin{aligned}\frac{\texttt{blockDim.x}}{2} + \frac{\texttt{blockDim.x}}{4} + \ldots + 1 &= a\frac{1-r^n}{1-r} \\ &= \frac{\texttt{blockDim.x}}{2}\frac{1-(1/2)^{\log_2 \texttt{blockDim.x}}}{1-1/2} \\ &= \texttt{blockDim.x}\left(1 - \frac{1}{\texttt{blockDim.x}}\right) \\ &= \texttt{blockDim.x} - 1\end{aligned}$$

   so we have $(2 \times \texttt{blockDim.x} - 1) \times \texttt{gridDim.x}$ additions in total.

3. All the reads are done in the shared memory step which loads 2 elements per thread, so we have $2 \times \texttt{blockDim.x} \times \texttt{gridDim.x}$ reads in total or $2 \times N$ for an input size N.

4. The writes are done once per block when the `threadIdx.x == 0` so we have `gridDim.x` (number of blocks) writes in total.

5. Min: If `i >= len` then we would have 0 real operations. Max: The thread would have 1 addition in the shared step and $\log_2 \texttt{blockDim.x}$ additions in the reduction step. Average: The average number of operations is the result in a block size from 2. divided by the number of thread in a block:

$$\frac{2 \times \texttt{blockDim.x} - 1}{\texttt{blockDim.x}} = 2 - \frac{1}{\texttt{blockDim.x}}$$

6. The thread block will synch $\lfloor \log_2 \texttt{blockDim.x} \rfloor$ times in the reduction step.

7. The shared memory reduces the the number of global memory accesses to zero in the reduction step!

8. We could use segmented multiblock sums and/or thread coarsening for faster DRAM access.

9. We couuld modify the kernel with a grid-stride loop to really large input sizes that are bigger than the max dimensions, or use the segmented multiblock sum.

10. No, we can not use non-commutative operation because the threads will be mixing up the order of the operations as it runs in parallel.

11. Yes, the floating point arithmatic is not associative, so the order of the operations will affect the result.

OUPUT

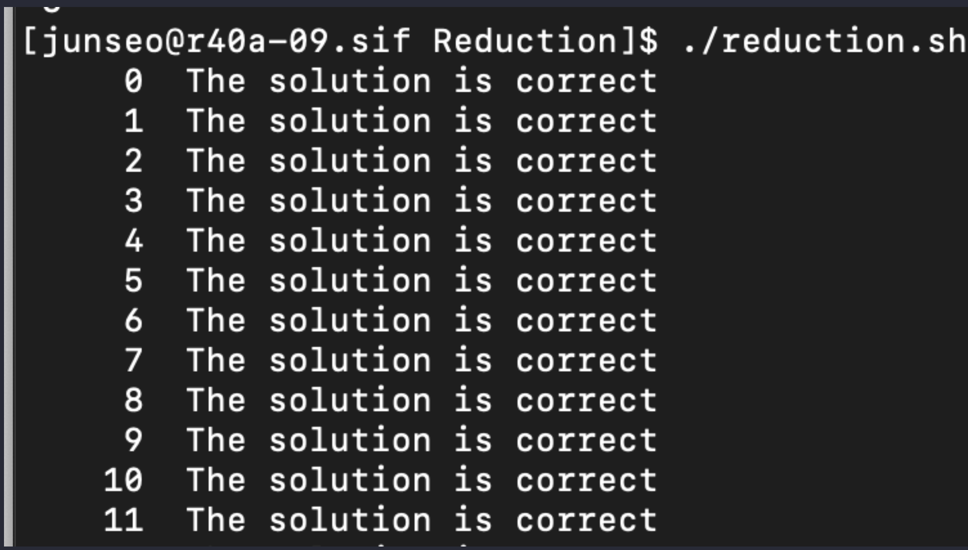

Figure 1: Successes