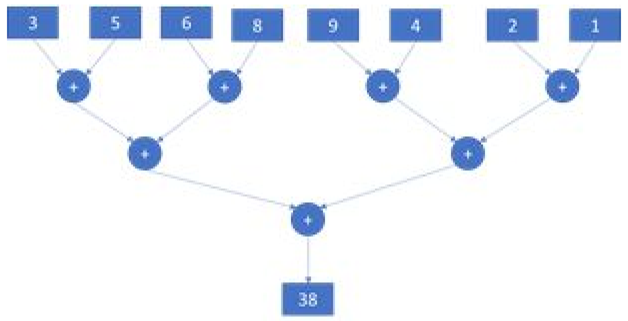
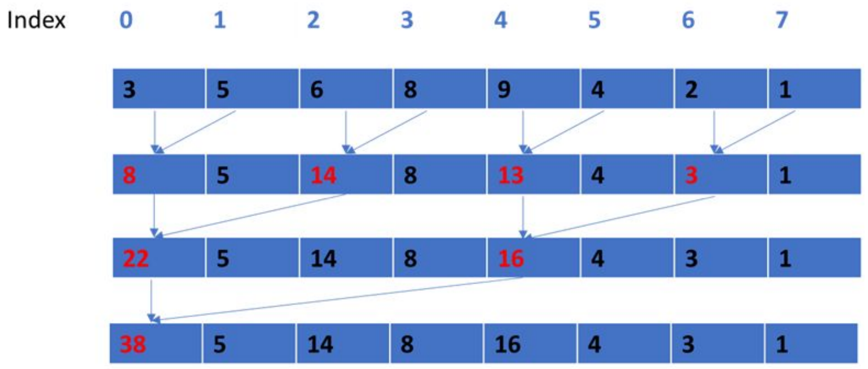
1. Exercise 1: Review the given source code (reduction_example.cu) for parallel reduction using sum operator for 2^20 elements

To run the program:
nvcc <filename>
a //(in windows)
or ./a //(in linux)

2. Exercise 2: Write a kernel that performs reduction as shown in figure using shared memory for 2^20 elements.



Here is how it can be designed



Elements in red represent the reduced values.
Note: Since each block will compute a single result, you will need to add all the block results at the end to get the final reduced result.

Instructor note: The solution is provided in reduction_exercise.cu
The for loop changes the stride by double in every iteration so that it reduces elements at stride 1 in the first iteration, at stride 2 in second iteration, at stride 4 in third iteration and so on
We can use this for loop to do so: for(int i=1; i < blockDim.x; i *= 2)
We can use this if (t_id % (2*i) == 0) to check that we are adding the correct index
Since we use blocks, after first kernel computation is completed, each of the block stores the result in the output array. So we need to add them all to get final result.

3. Exercise 3: Write a program for parallel reduction using max operation to find the max value among 2^20 elements
    a. First define host variables and a serial method to find max
    b. Define device variables and initialize them
    c. Create a kernel to find max similar to exercise 1 using global memory
    d. Create a kernel to find max similar to exercise 1 using shared memory