Exclusive prefix scan

1 Exclusive prefix scan

This exercise is about parallelizing a code that computes the exclusive prefix sum of an array of size n. Given an array `x`, this operation produces a new array `y` such that `y[i]=x[i-1]+y[i-1]`:

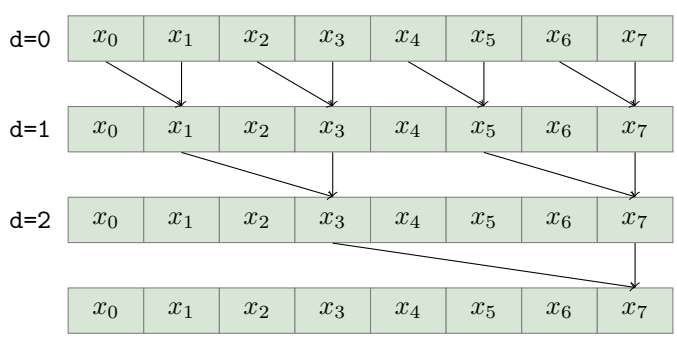
```
x=[3,  6,  7,  5,  3,  5,  6,  2]
y=[0,  3,  9, 16, 21, 24, 29, 35]
```

In our particular case, we will consider an **in-place** version which means that `y` is written directly into `x` in order to save memory. In sequential, this algorithm is very simple to implement and computes the result in a single pass through the `x` array. In parallel, instead, it is much more complicated; in this exercise we will use the algorithm proposed by Blelloch. To make things simpler, we will also assume that the length of `x` is a power of 2, i.e., $n = 2^b$. The algorithm works in two passes called, the *forward sweep* and the *backward sweep*, each made of b steps. The sum of two elements of `x` is computed through the `sum()` function.

Forward sweep The forward sweep performs $d = 0, \ldots, b-1$ steps and, at each step it does $n/2^{d+1}$ iterations. At each iteration, a left `l` and a right `r` coefficients are defined and the right coefficient is replaced with the sum of both:

```
x[r]=sum(x[l],x[r]);
```

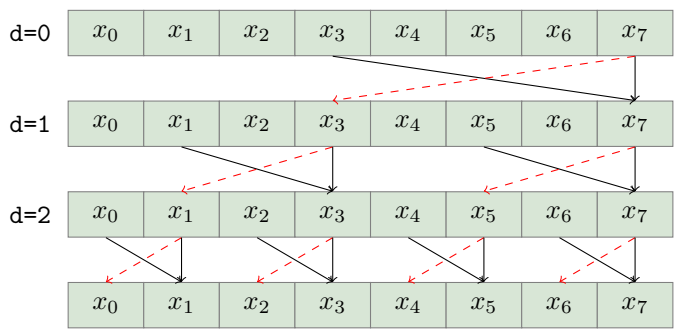
The forward sweep is illustrated in the figure below:



Backward sweep At the very beginning of the backward sweep, the value zero is copied in the last element of `x`. Then, the backward sweep performs $d = b - 1, \ldots, 0$ steps and, at each step it does $n/2^{d+1}$ iterations. As in the forward sweep, at each iteration, a left `l` and a right `r` coefficients are defined; the left is replaced with the right coefficient and the right is replaced with the sum of both

```
t = x[l];
x[l] = x[r];
x[r] = sum(t, x[l]);
```

The backward sweep is illustrated in the figure below:



2 Package content

In the `prefixscan` directory you will find the following files:

- `main.c`: This file contains the main program. This reads from command line the value of b such that the size of the array is $n = 2^b$. Then it computes the exclusive prefix scan sequentially to provide a reference. Finally, it calls the `parallel_scan` routine which implements the Blelloch algorithm and which must be parallelized. **Only the `parallel_scan` routine must be modified for this exercise.**

- `aux.c, aux.h`: these two files contain auxiliary routines and declarations and **must not be modified**.

The code can be compiled with the `make` command: just type `make` inside the `prefixscan` directory; this will generate a `main` program that can be run like this:

```
$ ./main b
```

where `b` defines the size of the array `x` as explained above.

3 Assignment

- The objective of this exercise is to parallelize the code of the `parallel_scan` routine **using OpenMP tasks** in order to reduce its execution time. At the beginning, this routine contains a sequential implementation of the Blelloch algorithm. It is sufficient to add OpenMP directives without modifying the provided code. Please note that the Blelloch algorithm makes $2n$ operations, whereas the simple sequential code only n. Therefore, it is normal if no time reduction is obtained when only two threads are used.