README for week 6 in IN3200/IN4200

Exercise 1) Load Balancing

Assuming that each task can't be divided between workers. Each worker gets $\lfloor n/p \rfloor$. This gives us a remainder $r = n \bmod p$. Since $r < p$ we can just choose r workers, first, last or at random, to do the last r tasks.

A variant that gives the first r workers an extra task can be implemented in C as follows

```
int my_tasksize, tasksize_total, myrank, numprocs;

// Find out myrank, numprocs, tasksize_total here
// ...

my_tasksize = tasksize_total / numprocs;
if (myrank < tasksize_total % numprocs) my_tasksize ++;
```

Depending on framework, how you figure out the variables differ. Using MPI you can get `myrank` and `numprocs` like this

```
// Initialize the MPI environment
MPI_Init(NULL, NULL);

// Get the number of processes
int numprocs;
MPI_Comm_size(MPI_COMM_WORLD, &numprocs);

// Get the rank of the process
int myrank;
MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
```

Exercise 2) Scheduling

The fastest solution is when there is zero idle time. The tasks are not dependent on each other so the only risk of idle time is if we can't balance the load between the workers.

If we denote the tasks as T1, T2, ..., T20 we get the following schedules.

Two equal workers:

When there is only two workers we can divide the tasks in to 10 pairs where two and two pairs take the same amount of time. T20 + T17 = T19 + T18, T16 + T13 = T15 + T14, and so on.

Time	1	2
$t = 0$	T20	T19
$t = 19$		T18
$t = 20$	T17	
$t = 37$	T16	T15
$t = 52$		T14
$t = 53$	T13	
$t = 66$	T12	T11
$t = 77$		T10
$t = 78$	T9	
$t = 87$	T8	T7
$t = 94$		T6
$t = 95$	T5	
$t = 100$	T4	T3
$t = 103$		T2
$t = 104$	T1	
$t = 105$	Done	Done

Another possible solution that gives the same execution time is to notice that the tasks can be divided into 10 equal pairs of 21 hours. (20+1), (19+2), (18+3), ..., (11+10). Now each processor can take five pairs each. for a total of 105 hours.

Three equal workers:

Here we can not just blindly give the longest task to the first free worker. This strategy gets us close, but we will end up with some idle time when the last few tasks are done. Setting up the table and rearranging we get

Time	1	2	3
$t = 0$	T20	T19	T18
$t = 18$			T17
$t = 19$		T16	
$t = 20$	T15		
$t = 35$	T14	T13	T12
$t = 47$			T11
$t = 48$		T10	
$t = 49$	T9		
$t = 58$	T8	T7	T6
$t = 64$			T3
$t = 65$		T5	
$t = 66$	T4		
$t = 67$			T2

Time	1	2	3
$t = 69$			T1
$t = 70$	Done	Done	Done

Another possible solution is to divide the last 12 tasks into six pairs of 29. (20+9), (19+10), ..., (15+14). Two pairs are given to each worker. Then we only need to divide the last 8. The remaining tasks take a total of 36 hours to complete sequentially. So a optimal solution is when each worker gets 12 hours (36/3). One way to do this is to divide into (8+4), (7+5) and (6+3+2+1), or (8+4), (7+3+2) and (6+5+1), or (8+3+1), ...

Exercise 3) Idling

The point in this exercise is that some workers/processors must wait while the earlier tasks complete.

We denote the tasks T1 - T15 from the top down, and left to right.

Four workers

Time	1	2	3	4
$t = 0$	T1			
$t = 10$	T2	T3		
$t = 20$	T4	T5	T6	T7
$t = 30$	T8	T9	T10	T11
$t = 40$	T12	T13	T14	T15
$t = 50$	Done	Done	Done	Done

You can shuffle the tasks a bit, but you can not find a quicker schedule. Shortest time is 50 min.

Three workers

Time	1	2	3
$t = 0$	T1		
$t = 10$	T2	T3	
$t = 20$	T4	T5	T6
$t = 30$	T7	T8	T9
$t = 40$	T10	T11	T12
$t = 50$	T13	T14	T15

Time	1	2	3
$t = 60$	Done	Done	Done

Shortest time is 60 min.

Exercise 4) Block Decomposition

We can think of this exercise as the 2D version of exercise 1. The block division in one direction is independent of the other, so the problem become two 1D problems.

Labeling the processes as P_{ij}, and their corresponding workload as $k_i \times l_j$, our task is to find k_i and l_j.

$$k_i = \left\lfloor \frac{M}{S} \right\rfloor + e_i, \qquad l_i = \left\lfloor \frac{N}{T} \right\rfloor + f_i$$

$$e_i = \begin{cases} 1, & \text{if } i < M \bmod S \\ 0, & \text{Otherwise} \end{cases} \quad l_j = \begin{cases} 1, & \text{if } j < N \bmod T \\ 0, & \text{Otherwise} \end{cases}$$

In C code:

```
int k, l, M, N, mycoords[2], numprocs_cartesian[2];

// Find out mycoords, numprocs_cartesian, m, n here
// ...

k = m / numprocs_cartesian[0];
if (mycoords[0] < M % numprocs_cartesian[0]) k ++;

l = n / numprocs_cartesian[1];
if (mycoords[1] < N % numprocs_cartesian[1]) l ++;
```

NOTE: This does not give the fairest possible division (see ex. 5), but with this decomposition the communication between neighboring processes is simple due to the regular boundaries. Given a process P_{ij} the height of the block assigned to it is the same for all its right/left neighbors. similarly it has the same width as all its above/below neighbors.

If you are using MPI for a 2D problem you can create a Cartesian 2D grid of processors like this:

```
// Initialize the MPI environment
MPI_Init(NULL, NULL);
```

```
MPI_Comm comm_cart;
int ndims, reorder, rank_cart;
int dim_size[2], periods[2];

ndims = 2; // 2D grid.
dim_size[0] = S;
dim_size[1] = T;

// Can the processes be reordered in the new communicator?
reorder = 1; // Yes

// Should the grid wrap around in the given dimension? (problem dependent)
periods[0] = 1; // Wrap
periods[1] = 0; // No wrap

// Create new communicator with Cartesian topology.
MPI_Cart_create(MPI_COMM_WORLD, ndims, dim_size, periods, reorder, &comm_cart);

// Get the rank and position of the process in the new grid.
MPI_Comm_rank(comm_cart, &rank_cart);
MPI_Cart_coords(comm_cart, rank_cart, ndims, mycoords);
```

Exercise 5) Load Imbalance

Here i am using the notation from ex. 4.

8 by 1

Here all processors have the same number of columns, $l = 100$. For k we have either $k = \lfloor 100/8 \rfloor = 12$, or $k = \lfloor 100/8 \rfloor + 1 = 13$.

Calculating the load imbalance factor we get

$$\frac{\max_i(W_i) - \min_i(W_i)}{\min_i(W_i)} = \frac{13 \times 100 - 12 \times 100}{12 \times 100} = \frac{1}{12}$$

4 by 2

Both 4 and 2 are divisible by 100. This means that every process gets the same amount of work. $k = 25$ and $l = 50$.

The load imbalance factor is obviously

$$\frac{25 \times 50 - 25 \times 50}{25 \times 50} = 0$$

3 by 3

100 is not divisible by 3. we end up with a remainder of 1 in both coordinate directions.

$$\min_i(W_i) = 33 \times 33 \max_i(W_i) = 34 \times 34$$

$$\frac{34 \times 34 - 33 \times 33}{33 \times 33} \approx 0.0615$$