CSC458 Assignment 1

Chi-Chun, Chen

1 Program environment

I use python2.7 and the code should not be run on python3 since the usage of __cmp__ is different on these two versions.

Exercise 1

Below is the table recording the answer of exercise 1.1 and 1.2, which is the total tasks on critical on each graph and the minimum processors with same speed as infinite core respectively. All of the output file has been checked by des.py

Table 1: Exercise 1

	Total tasks on Critical Path	Minimum Processors
3x5x2.gr	3	4
3x5x4.gr	2	3
small-1.gr	11	92
small-2.gr	10	61
medium-1.gr	23	1227
large.gr	44	4304

```
// Below is the list of all the tasks in the critical path of
// graph large.gr, and the reason to list of them here is because
// there are three critical paths and each has some sharing
// dependences.

len:  37 [17253, 17224, 17211, 17177, 17115, 17081, 17029, 16959,
   16945, 16907, 16745, 16642, 16586, 16516, 16407, 15923, 15720,
   15488, 15433, 15187, 15031, 14718, 14007, 13560, 13386, 12997,
   12605, 12276, 11783, 8673, 8510, 8192, 7541, 6971, 2592, 695,
   199]
len:  37 [17257, 17224, 17211, 17177, 17115, 17081, 17029, 16959,
   16945, 16907, 16745, 16642, 16586, 16516, 16407, 15923, 15720,
   15488, 15433, 15187, 15031, 14718, 14007, 13560, 13386, 12997,
   12605, 12276, 11783, 8673, 8510, 8192, 7541, 6971, 2592, 695,
   199]
```

```
len:  38 [17258, 17239, 17197, 17179, 17153, 17105, 17081, 17029,
   16959, 16945, 16907, 16745, 16642, 16586, 16516, 16407, 15923,
   15720, 15488, 15433, 15187, 15031, 14718, 14007, 13560, 13386,
   12997, 12605, 12276, 11783, 8673, 8510, 8192, 7541, 6971, 2592,
   695, 199]
'There are', 44, 'tasks on critical path'
'Minimum finite number of processors needed to execute: ', 4304
```

Exercise 2

To prove whether B is a critical task I first assume task B to be not a critical task. Also, from the description of exercise 2, we know that the slack time $S_A = 0$, since task A is a critical task. From the assumption above, we know that $S_B = T_{B\_latest} - T_{B\_earliest} > 0$. And then, we now compute the slack time of C from the above information, which get $S_C = T_{C\_latest} - T_{C\_earliest}, T_{C\_latest} = S_B + E_B$ (E_B is the execution time of task B). Since $S_C = S_B + E_B - T_{C\_earliest}$ and by the assumption that task C is a critical path, we know that E_B should be equal to $T_{C\_earliest}$, and $S_C = S_B$. However, it is a contradiction that $S_C = S_B$ which is greater than zero because in the assumption of task C is a critical path has been violated. Therefore, we proved that task B is a critical task if task A and task C both are critical tasks.

Exercise 3

1. Executing time v. the number of processors of each graph:

Table 2: Execute the task with longest execution time

	One core	Two cores	Four cores	Eight cores	Sixteen cores
3x5x2.gr	113	61	50	50	50
3x5x4.gr	72	39	37	37	37
small-1.gr	4104	2068	1052	631	379
small-2.gr	2381	1224	610	346	249
medium-1.gr	67076	33546	16814	8418	4248
large.gr	268611	134318	67184	33619	16838

2. How did you schedule tasks when the number of tasks that were ready to execute exceeded the number of processors?

I implemented a Data Structure to trace the execution time of each core and use a priority queue to store these cores. By using this strategy, I could easily assign tasks to the core with minimum usage and therefore achieve load balancing on each core.