Bilkent University CS 342 Project 2 Report

BILKENT UNİVERSİTY

EGE TURAN – 21502441

CS 342 – SEC1

FALL-2019

Table of Content

1. Runtime Results for the Experiment.

1.1 Part1

File Size: 10000

K	N	File sizes	Run-time
100	1	10000	0.002951
100	3	10000	0.014689
100	6	10000	0.027015
100	9	10000	0.048912

K	N	File sizes	Run-time
2000	1	10000	0.015885
2000	3	10000	0.032432
2000	6	10000	0.045036
2000	9	10000	0.080588

K	N	File sizes	Run-time
9000	1	10000	0.138278
9000	3	10000	0.183552
9000	6	10000	0.223091
9000	9	10000	0.241325

File Size: 100000

K	N	File sizes	Run-time
100	1	100000	0.021289
100	3	100000	0.082109
100	6	100000	0.278872
100	9	100000	0.371207

K	N	File sizes	Run-time
2000	1	100000	0.034030
2000	3	100000	0.092580
2000	6	100000	0.212963
2000	9	100000	0.324741

K	N	File sizes	Run-time
9000	1	100000	0.146953
9000	3	100000	0.264952
9000	6	100000	0.779850
9000	9	100000	1.066164

1.2 Part2

File Size: 10000

K	N	File sizes	Run-time
100	1	10000	0.000320
100	3	10000	0.000564
100	6	10000	0.000879
100	9	10000	0.001024

K	N	File sizes	Run-time
2000	1	10000	0.010484
2000	3	10000	0.011214
2000	6	10000	0.012873
2000	9	10000	0.015000

K	N	File sizes	Run-time
9000	1	10000	0.126974
9000	3	10000	0.133969
9000	6	10000	0.139229
9000	9	10000	0.143256

File Size: 100000

K	N	File sizes	Run-time
100	1	100000	0.000347
100	3	100000	0.000639
100	6	100000	0.000887
100	9	100000	0.000977

K	N	File sizes	Run-time
2000	1	100000	0.007028
2000	3	100000	0.009124
2000	6	100000	0.008985
2000	9	100000	0.009200

K	N	File sizes	Run-time
9000	1	100000	0.098390
9000	3	100000	0.107222
9000	6	100000	0.112527
9000	9	100000	0.128479

2. Explanation for the Data.

I used Virtual Machine while testing and implementing the project. Virtual machine uses multi-processing while computing. My virtual machine uses 5 CPU to compute. This is the first reason why execution time does not increase linearly with N (number of processes) with small input sizes because child processes can be executed in different CPUs concurrently.

For my experiment results in the part b which works with multiple processes in synchronous manner. Increasing process count does not reflect too much time increase in my result. I can explain this with the concurrency provided by multi-process approach. They are running on different CPU's so that when I increased the number of processes 1 to 9, running time of the program does not increase linearly. Actually it ups to 0.098390 to 0.128479 which is a small increase. Other reason why part B is faster than part A is, part B is using shared memory which is very fast because kernel is not involved in intermediate process communication. It makes the program very fast.

3. What are the outcomes?

With both approaches I used –processes and thread- running times are not that much even I give very large input because concurrency helps us to save time. Although, I used swap function in my project which takes some time, running times are very fast thanks to concurrency.