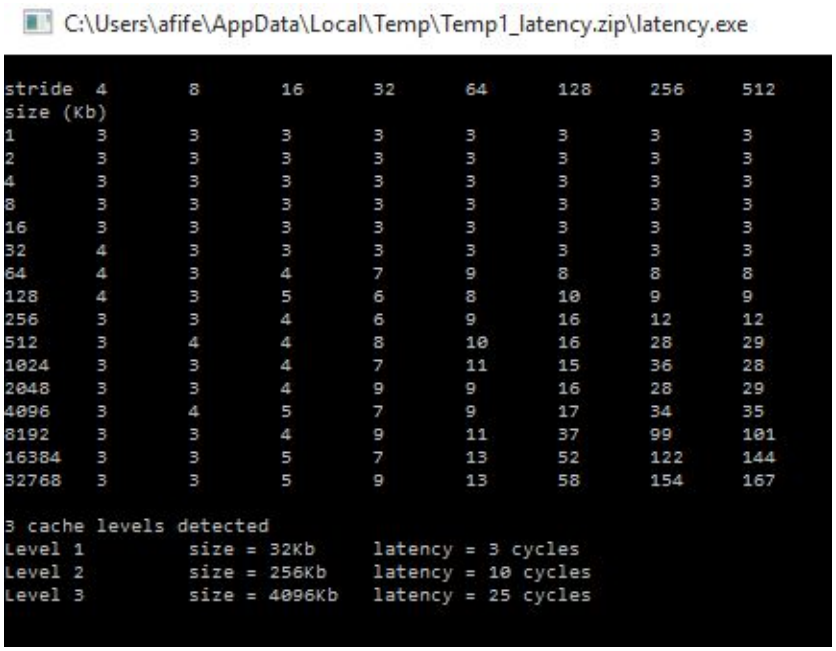
Inf2440 Oblig 4

Alexander Fife

Notes:
Valgte å splitte grafen i 2, fra nederst til venstre, mot øverst til høyre. Fikk en speedup på over 2 fra det alene (Målingen ble tatt etter at alle nodene hadde fått tildelt side første gang). Hadde en mistanke om at ubalanserte trær førte til dårligere speedup fra parallellitet.
(Måling fra rett før første rekursjonskall)
N = 10 mill
1 tråd etter optimalisering = 100ms
1 tråd før optimalisering = 220ms

Fikk ikke til å få punktene på linjen i en sortert rekkefølge. Dette hadde lite med parallellitet å gjøre, og mer med at jeg ikke klarte å finne på en elegant løsning for å gjøre det. Det jeg tenkte på var å sjekke distansen fra punkt til punktet på linjen, og sortere etter det. Er det den beste løsningen?

Hardware:
Modell: Lenovo ThinkPad T440p
CPU: Intel(R) Core(™) i7-4700MQ CPU @2.4GHz - 8 kjerner
Kjøres på jobb laptoppen med 8 kjerner, av en eller annen grunn følte jeg ikke at den fikk utnyttet kjernene på en god måte. Fikk best speedup på 2/4 threads. Tilslutt viser jeg 4 vs 8 threads kjøring.



Tok med en ekstra N verdi for å lettere se speedup.
Verdier er tatt litt på øyemål. Tar ca medianen, og skyver den/opp ned utifra andre verdier

8 tråder	100	1000	10000	100000	1000000	10000000	70000000
Sekv	0.12ms	0.4ms	1.4ms	5ms	30ms	270ms	2100ms
Para	0.7ms	0.9ms	3ms	15ms	26ms	220ms	1650ms
SU	0.171	0.44	0.46	0.3333	1.15	1.22	1.27

Grunn til at speedup ikke er noe spesielt:

1. O(n) kjøretid for å finne punktene med minst/største X/Y verdi før tråder har startet.
2. O(n) tid går til å fordele hvilke sider punktene skal være på etter vi har valgt punktene våre.
3. En del (halveis/helt) sekvensielle deler der man flytter over verdier fra en IntList til en annen
4. Siden algoritmen er svært effektiv til å begynne med O(log n)? Så får man ikke like stor speedup av større tall som man får fra andre (f.ex O(n^x)) algoritmer

Vanligvis, jo større n jo bedre speedup. Men i dette tilfellet med så mye som kjører på O(n) før vi kommer til det parallelle jevner det ut speedupen på de større verdiene vs O(log n).

Kjørte noen raske tester hvor jeg målte tiden først etter jeg hadde fordelt sidene på punktene for å vise hvor parallelliteten hadde mest å si.

	1000000	10000000	70000000
Sekv	18ms	115ms	800ms
Para (8 threads)	11ms	70ms	470ms
	1.6	1.64	1.7
Para (4 threads)	11ms	60ms	430ms

(2 threads var ca samme som 4 threads, stoler ikke helt på den maskinen her).

Det er sikkert mulig å få kjørt den sekvensielle starten parallellt, men så vidt jeg forsto så var det ikke en del av oppgaven.