Oblig1 inf2440

Da jeg en stund hadde problemer med sorteringsalgoritmen på noen sett med verdier og ikke med andre, ville jeg teste med forskjellige verdier for vært run. Derfor er det ikke satt opp noen verdi i tallgeneratoren. Feilen er nå rettet. Her ser man også at algoritmen som sorterer er viktig.

Algoritmen jeg bruker for å samle de 50 største først i arrayen fra de forskjellige områdene som hver har sine 50 største tar også vare på integriteten til resten av arrayen. En ordinær merge av de forskjellige arrayene, da de allerede er sorterte, hadde vært raskere for å samle de forskjellige '50 største', men hadde økt kompleksiteten på å beholde integriteten til den ordinære arrayen.

Oppgave 1:

Antall verdier	Arrays.sort() (ms)	Sekvensiell A2 (ms)	Speedup S
1000	1,18	0,15	7,76
10000	0,826	0,027	30,59
100000	10,09	0,19	53,97
1000000	118,20	1,86	63,72
10000000	1178,20	18,99	62,06
100000000	11376,78	186,62	60,96

Oppgave 2:

Antall verdier	Sekvensiell A2 (ms)	Parallell A2 (ms)	Speedup S
1000	0,01	1,23	0,00
10000	0,03	1,23	0,02
100000	0,19	1,27	0,15
1000000	1,90	1,68	1,13
10000000	18,65	7,36	2,54
100000000	189,91	48,91	3,88

iSortWrap, iSortSeq og iSortRest blir brukt både av den sekvensielle og den parallelle løsningen (til tross for navnene). De blir kalt med iSortWrap(a, l, r) hvor a er arrayen som skal sorteres, l er verdien lengst til venstre som skal sorteres (0 for sekvensielle) og r for den lengst til høyre (a.length - 1 for den sekvensielle).

```
void iSortWrap(int[] a, int l, int r) {
    iSortSeq(a, l, r);
    iSortRest(a, l, r);
}
void iSortSeq(int[] a, int l, int r) {
    int j;
    int t;
    for (int i = l + 49; i >= l; i--) {
        j = i;
        t = a[i];
        while(j < l + 49 && t < a[j + 1]) {
            a[j] = a[j + 1];
            j++;
        } // end j
        a[j] = t;
    } // end i
} // end iSortSeq
void iSortRest(int[] a, int l, int r) {
    int t, j;
    for (int i = l + 50; i <= r; i++) {
        if (a[i] > a[l + 49]) {
            t = a[i]; a[i] = a[l + 49]; j = l + 48;
            while(j >= l && t > a[j]) {
                a[j+1] = a[j];
                j--;
            } // end j
        a[j+1] = t;
        }
    } // end i
} // end iSortRest
```

Her synes det hvordan trådene kaller iSortWrap og sorterer verdiene fra de forskjellige delene av arrayene.

```
void iSortPar(int[] a) {
    bwait = new CyclicBarrier(q + 1);
    bfinish = new CyclicBarrier(q + 1);
    for (int i = 0; i < q; i++)
        new Thread(new SortWorker(i,a)).start();
    try {
        bwait.await();
        bfinish.await();
    } catch (Exception e) {return;}

} // end iSortPar
void merge(int[] a) {
    int t, j;
    for (int n = c/q; (n + 49) < c; n += c/q) {
        for (int i = n+49; i >= n; i--) {
            if (a[i] > a[49]) {
                t = a[i]; a[i] = a[49]; j = 48;
                while(j >= 0 && t > a[j]) {
                    a[j+1] = a[j];
                    j--;
                }
                a[j+1] = t;
            }
        }
    }
} // end merge
class SortWorker implements Runnable {
    int index;
    int [] a;
    SortWorker(int index, int [] a) {
        this.index = index;
        this.a = a;
    }
    public void run() {
        if (index != q - 1) iSortWrap(a, c/q*index, (c/q*(index + 1))-1);
        else iSortWrap(a, c/q*index, c-1);
        try {
            bwait.await();
            if (index == q - 1) merge(a);
            bfinish.await();
        } catch (Exception e) {return;}
    }
} // end class SortWorker
```