Sortering med tråder - Quicksort

Skisser til to programmer
INF1010 – våren 2013

Stein Gjessing
Institutt for informatikk
Universitetet i Oslo

Sortering som tema, slikt som valg av sorteringsmetode, hastigheten til de forskjellige sorteringsmetodene mm. er ikke pensum i INF1010.
På den annen side vil de fleste sorteringsmetoder kunne programmeres med det du har lært i INF1010. I så måte er det å kunne sortere pensum.

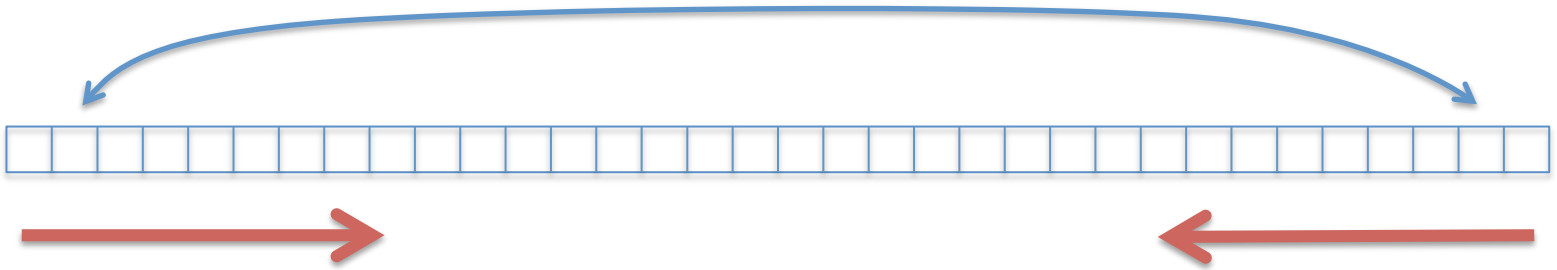
Detaljene i sorteringsmetoden quicksort er ikke pensum i INF1010, men ideene til løsning med tråder som presenteres her er pensum.

En sorteringsmetode kalt Quicksort

Selve sorteringsmetoden quicksort er ikke pensum,
men ideene til løsning med tråder som presenteres her er pensum

- Idé til å sortere en tabell:
  1. Legg alle de små verdiene nederst i tabellen og alle de store øverst
  2. Sorter den nederste delen av tabellen
  3. Sorter den øverste delen av tabellen
  4. Resultat: Hele tabellen er sortert

(Kan gjøres som en rekursiv metode)



Gå mot midten og bytt om hvis verdiene er i feil ende
Litt fiklete når du kommer til midten

Parallelisering av quicksort

- Skissen beskriver et idealisert tilfelle
  - Quicksort klarer i virkeligheten ikke å dele tabellen som skal sorteres i nøyaktig to like store deler
- Første versjon: Etter at vi har flyttet små verdier og store verdier til hver sin del, lager vi **to nye tråder** som sorterer hver sin del
- Andre versjon: Etter at vi har flyttet små verdier og store verdier til hver sin del, lager vi bare **én ny tråd** som sorterer den ene delen, mens vi sorter den andre delen selv

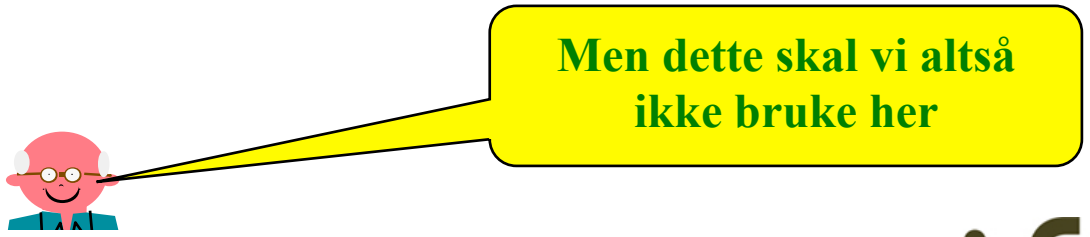
- Tråder kan også vente på ”barn” f.eks. ved hjelp av klassen CountDownLatch og metodene ”countDown” og ”await”, men her skal vi bruke de vanlige ”wait” og ”notify” til dette. Samtidig bygger vi ”monitoren” inn i tråden. Er det bra?

Repetisjon barrierer:
Vente på at noe er skjedd

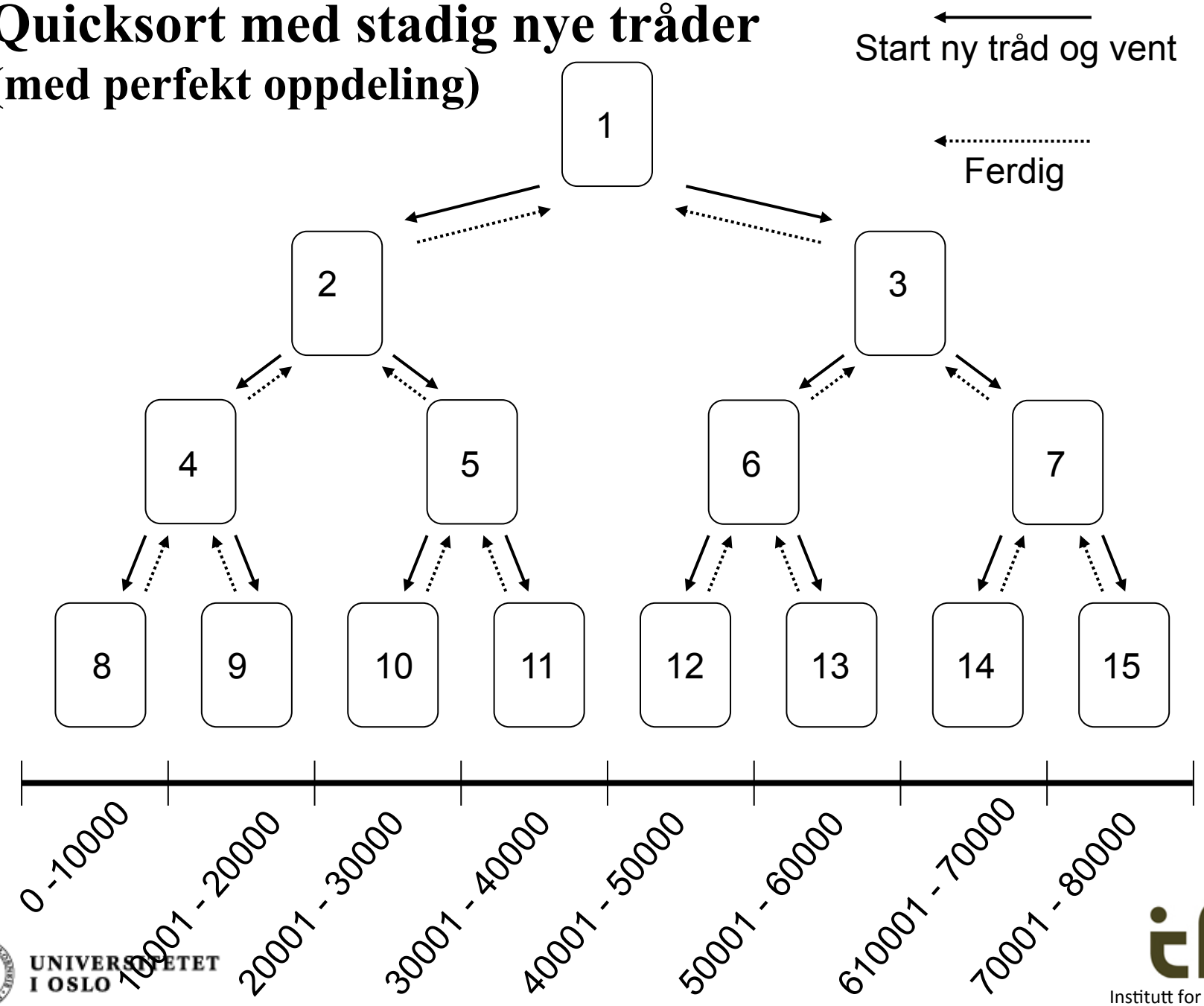
```
CountDownLatch barriere = new CountDownLatch(antallTrader);

for ( . . . )  {
   // lag tråder som gjør en jobb:
   new Trad(. . . ,barriere).start();
}
// Vent på at ALLE er ferdig med jobbene sine:
barriere.await();
```

Når run()-metoden i klassen Trad er ferdig med jobben:

```
barriere.countDown();
```



Quicksort med stadig nye tråder
(med perfekt oppdeling)



```
class QuicksortThread1 extends Thread {
        <peker til tabellen som skal sorters>;
        <grensene for den delen som denne tråden skal sortere>;
         QuicksortThread1    mor;
    QuicksortThread1(<peker til tabell og grenser>, QuicksortThread1 morPeker) {
        <kopier peker til tabell med grenser inn i dette objektet> ;  mor = morPeker;
    }
    public void run(  ){
        if (<Stor nok tabell til å dele opp i tråder>) {
            <Del opp: Flytt små verdier nederst, store verdier øverst>;
            new QuicksortThread1(<nedre del av tabellen>,this) . start( );
            new QuicksortThread1(<øvre del av tabellen>,this) . start( );
             <vent på begge subtrådene>;
        }
        else <sorter hele tabellen på vanlig måte>;
        <si fra til mor at dette barnet er ferdig>;
     } // end  run
}
class Quick1 {
public static void main(String[ ] args) {
            <deklarer og lag selve tabellen>;
            new QuicksortThread1(<hele tabellen>).start( );
    }
}
```

Skisse

Nesten ferdig program (4 sider)

```
class QuicksortThread1 extends Thread {

    int [ ] tab;      int startInd, endInd;
    QuicksortThread1    mor;   // i  roten skal mor ha verdien null
    int antallFerdigeSubtrader;
    final static int minsteLengde = 10000;

    // Konstruktør vanlig node:
   QuicksortThread1(int [ ] tb,  int st,  int end, QuicksortThread1 mr) {
        tab = tb;     startInd = st;   endInd = end;
        mor = mr;
   }

    // Konstruktør rot:
  QuicksortThread1(int [ ] tb) {
       tab = tb;    startInd = 0;    endInd = tb.length;
       mor = null;
  }
```

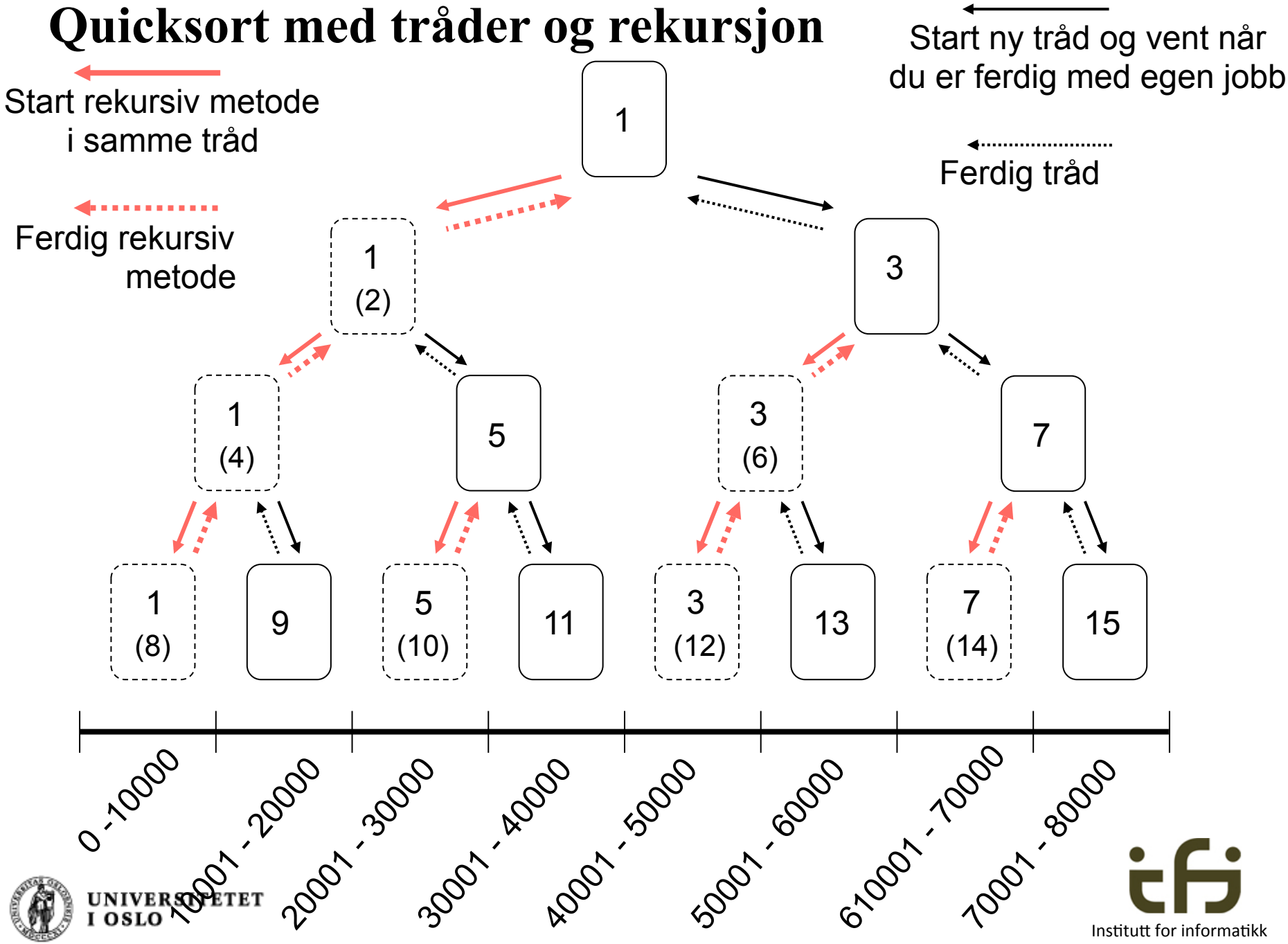
```
public void run(  ){
     if (endInd-startInd > minsteLengde) { // lang tabell, lag tråder:
         antallFerdigeSubtrader = 0;
         int indeks = delOpp(tab,startInd,endInd);
         // lag og start to barn:
         new QuicksortThread1(tab,startInd,indeks,this).start( );
         new QuicksortThread1(tab,indeks+1,endInd,this).start( );
         // vent på subtrådene:
        vent( );
     }  else sorterPaVanligMate(tab,startInd,endInd);

     if (mor != null){
          // signaler at denne tråden er ferdig med jobben:
          mor.ferdig(minId);
     } else System.out.println (”Ferdig sortert”);
} // end  run
```

```
synchronized void vent( ) {
    while (antallFerdigeSubtrader != 2) {
            try {wait( );
            } catch (InterruptedException e) {
                System.out.println(" Ukjent avbrudd");
                System.exit(0);
            }
    } // etter while: antallFerdigeSubtrader = = 2
}

synchronized void ferdig ( ) {
        antallFerdigeSubtrader ++;
        notify();
}
```

```
    void sorterPaVanligMate(int[] tb, int st, int en) {
       // Her skulle det vært sortering
     }

    int delOpp(int[] tb, int st, int en) {
          //  Her skulle vi flyttet og returnert indeksen til ”skillepunktet”
        return ((st+en) / 2);
     }

} // end QuicksortThread1
```

```
class Quick1 {
     static int [ ] tabell;
     public static void main(String[ ] args) {
            tabell = new int[80000];  // + put data i tabellen
            // Lag og start en rot:
            new QuicksortThread1(tabell).start( );
     }
}
```

Quicksort med tråder og rekursjon



```
class QuicksortThread2 extends Thread {
        <peker til tabellen som skal sorters>;
        <grensene for den delen som denne tråden skal sortere>
         QuicksortThread2   mor;
    QuicksortThread2(<peker til tabell og grenser>, QuicksortThread2 morPeker) {
        <kopier peker til tabell med grenser inn i dette objektet> ;  mor = morPeker;
     }
     public void run(){
        sorter(<tabell med grenser>);
         <vent på alle subtråder>;
         <si til mor at jeg er ferdig>;
     } // slutt  run

     public void sorter(<tabell med grenser>) {
        if (<Stor nok tabell til å dele opp i tråder>) {
            <Del opp: Flytt små verdier nederst, store verdier øverst>;
            new QuicksortThread2(<øvre del av tabellen>,this) . start( );
            sorter (<nedre del av tabellen i denne tråden>);
         }
         else <sorter hele tabellen på vanlig måte>;
     } // slutt sorter
} // slutt QuicksortThread2
```

Skisse 2

Nesten ferdig program 2

```
class QuicksortThread2 extends Thread {
      int [ ] tab; int startInd, endInd; int antallBarn;
      final static int minsteLengde = 10000;
      QuicksortThread2 mor;  // mor er null i roten
       int antallFerdigeSubtrader = 0;
       int minId = 0; int minEgentligeId = 0;

  // Konstruktør vanlig
 QuicksortThread2(int [ ] tb, int st, int en, QuicksortThread2 mr, int ind){
    tab = tb; startInd = st; endInd = en;
    mor = mr; minId = ind; minEgentligeId = ind;
    antallBarn = 0;
 }
   // Konstruktør rot:
 QuicksortThread2(int [ ] tb){
     tab = tb; startInd = 0; endInd = tb.length;
     mor = null; minId = 1; minEgentligeId = 1;
     antallBarn = 0;
 }
```

```
public void run(){
      sorter(tab,startInd,endInd);
      ventPaAlle( );
      if (mor != null) {
           // signaler at denne tråden er ferdig med jobben:
          mor.ferdig( );
      }
     else {System.out.println("FERDIG");}
} // end of run

void sorter(int[] tb, int st, int en) {
      if  (en-st > minsteLengde) {
                int indeks = delOpp(tab,st,en);
                antallBarn ++;
                new QuicksortThread2(tab,indeks+1,en,this,minId*2+1).start();
                minId= minId*2;
                sorter(tab,st,indeks);   // rekursivt kall
      }
      else sorterPaVanligMate(tab,st,en);
}
```

```
synchronized void ventPaAlle() {
    while (antallBarn != 0) {  // antall barn ikke ferdig
            try {
                wait();
             }
             catch (InterruptedException e) {
                System.out.println(" Uventet avbrudd ");
                System.exit(0);
             }
    }
}

synchronized void ferdig ( ) {
       antallBarn --;
       notify();
}
```

```
    void sorterPaVanligMate(int[] tb, int st, int en) {        }

    int delOpp(int[] tb, int st, int en) { return (st+en)/2; }

} // end QuicksortThread2
```

```
class Quick2 {
      static int [ ] tabell;
      public static void main(String[ ] args) {
            tabell = new int[80000];  // + put data i tabellen
            // Lag og start en rot:
            new QuicksortThread2(tabell).start( );
      }
}
```