Morten Stavik Eggen
Georg Vilhelm Seip

Vi valgte oppgave 1 fordi den sto først.

Jeg(Morten) får ikke lov å kjøre et int arr[million], får en error om stack overflow (trur jeg), så det største jeg har kjørt med i disse testene er 2.5 * 10^5, nærme nok håper jeg.
I bildene jeg viser er Arr1 kjørt med bare quicksort og Arr2 kjørt med hybrid av quicksort og insertion sort.
Vi valgte insertion sort, fordi boka anbefalte den over bubble.
Testet at den kjører begge sorteringene ved å verdien av når deltabellen er stor nok til å kjøre insertion.

```
if (h - v < 2000000)
{
    //median3sort(t,v,h);
    insertionsort(t, v, h);
    break;
}
```

brukte et stort tall bare for å sikkre at den alltid ble sann, da fikk jeg en veldig dårlig tid på arr2

```
checksum: 4.10101e+09
arr1 time: 33[ms]
arr2 time: 33992[ms]

checksum: 4.10101e+09
Arr2 is sorted
Sum1 er lik Sum2

got end
```

```
checksum: 4.10101e+09
arr1 time: 30[ms]
arr2 time: 33759[ms]

checksum: 4.10101e+09
Arr2 is sorted
Sum1 er lik Sum2

got end
```

```
checksum: 4.10101e+09
arr1 time: 31[ms]
arr2 time: 33864[ms]

checksum: 4.10101e+09
Arr2 is sorted
Sum1 er lik Sum2

got end
```

Vi kan se at insertion sort alene bruker ganske nøyaktig 1000 ganer så lang tid som quicksort.

Den mest optimale verdien vi fant er rundt 10-20

```
if (h - v < 10)
{
    //median3sort(t,v,h);
    insertionsort(t, v, h);
    break;
}
```

```
arr1 time: 34[ms]
arr2 time: 32[ms]
```

```
arr1 time: 34[ms]
arr2 time: 30[ms]
```

```
arr1 time: 34[ms]
arr2 time: 29[ms]
```

Arr2 klarer å sånn passe konsistent være 10-20% raskere enn Arr1

Morten Stavik Eggen
Georg Vilhelm Seip

Vi kan sammenligne dette med om vi kjører deltabellen med 200+

```
if (h - v < 250)
{
    //median3sort(t,v,h);
    insertionsort(t, v, h);
    break;
}
```

```
arr1 time: 31[ms]
arr2 time: 43[ms]
```

```
arr1 time: 33[ms]
arr2 time: 44[ms]
```

```
arr1 time: 31[ms]
arr2 time: 43[ms]
```

Konsistent 30% dårligere.

Det er mulig dette hadde blitt mer tydelig med 1 million tall, siden det er 4 ganger så mye å gjøre. Men får dette `[Done] exited with code=3221225725 in 1.289 seconds`

Vi har med sikkerhetstest i form av sjekksum og en for-loop som sammenligner arr2[i]<arr2[i-1] og setter en bool til false om dette skjer.

```
bool sorted = true;
for (size_t i = 0; i < arrSize; i++)
{
    //std::cout<<arr2[i]<<" ";

    if (i != 0 && arr2[i] < arr2[i - 1])
    {
        std::cout << "feil sortering på i: " << i << std::endl;
        std::cout << arr2[i - 1] << ">" << arr2[i] << std::endl;
        sorted = false;
    }
    checksum2 += arr2[i];
}

std::cout << "\nchecksum: " << checksum2 << std::endl;
if(sorted)std::cout<<"Arr2 is sorted"<<std::endl;
if (checksum2 == checksum)
    std::cout << "Sum1 er lik Sum2" << std::endl;
```

Den forteller også hvor feilen skjer om noe skjer, som var nyttig for det var en periode et tall ble satt til noe tilfeldig fra minnet og da slo det ut feil og så veldig rart ut.