Programmering: sorteringsalgoritmer

Jesper Svefelt

Anton Norberg

1. Första uppgiften finns i koden som vi skickat med. Det är skrivet i C#. Vi märkte att Quicksort tog mycket längre tid än Mergesort enligt vår kod. Largeints går betydligt snabbare än smallints.

2.

    a.
    b.
    c. Vi ändrade i program.cs, partiton (Shuffle) metoden där vi kontrollerade om array[j] är mindre eller lika med pivotelementet. Istället kontrollerar vi bara om array[j] är mindre än pivotelementet. Detta resulterade i att quicksorts tid med smallints nästan halverades och att den med largeints blev snabbare.

       Alltså när listan shufflas så kommer de element som är samma som pivotelementen kommer inte att flyttas.

       I vår original kod är privotelementet det sista/högsta elementet.

       I vår originalkod är vi provade att sätta privotelementet som medianen av första, sista och mellersta. Detta resulterade i att arrayen inte blev sorterad på ett rätt sätt (>10 flaws).

       Vi provade att sätta privotelementet till det mittersta men fick då Stack Overflow med quicksort. Provade även med det low-elementet som privotelement men då fick vi samma resultat (Stack Overflow).

       Vi fick även samma resultat (Stack Overflow) när vi tog ett random nummer som privotelement.

       Vi har kommit fram till att privotelementet måste vara det sista elementet i arrayen för att den skall kunna sortera alla elementen i våran quicksort-algoritm.

       Om man väljer det lägsta eller det största som pivot, och datasetet redan är sorterat kommer körtiden resultera i kvadratisk tid.

Nedan har vi testresultat:

Quicksort

- largeints: 0.328
- largeints redan sorterad array: Stack overflow
- largeints innan förändring (jämförelse): 0.438
- smallints: 13.484
- smallints redan sorterad array: 13.282
- smallints innan förändring (jämförelse): 28.859

Mergesort

- largeints: 0.469
- smallints: 0.39

Jesper Svefelt

Anton Norberg