Kreativ kodning

Efter besöket

Hoppas du och dina elever hade ett trevligt och givande besök hos oss på Curiosum.

Ett förslag till hur man kan fortsätta i skolan efter besöket är att arbeta vidare i Microsoft MakeCode. Nedan finns ett utarbetat förslag med att läsa kod och att analysera algoritmer. Sidan med ”Instruktioner elever” kan med fördel delas ut (utskrivet eller digitalt) till eleverna. Men det finns en poäng med att de måste skriva in koden själva istället för att klippa och klistra.

Det går även låta eleverna arbeta vidare med sina projekt som de påbörjade på Curiosum (eller nya), men då använda sig av den inbyggda simulatorn av hårdvaran i MakeCode. Men notera att beroende på vad de gör fungerar simulatorn mer eller mindre bra.

Övning: MakeCode

Instruktioner lärare

Arbete i par eller grupper av tre

Tid: 60 min

Eleverna ska i par eller i grupp av tre analysera och jämföra två olika sorterings-algoritmer: Bubbelsortering och Quicksort. Tanken är att de ska öva på att läsa algoritmer i form av kod och kunna formulera dem som text eller algoritmer på papper. De kommer även göra enkla ändringar i koden för att få ut körtiden för de olika sorteringsalgoritmerna.

Nedan följer förslag på svar på uppgifterna och lite mer info som kan vara intressant att diskutera med eleverna.

Uppgift 1: Bubbelsortering

Här är ett förslag på hur algoritmen för Bubbelsortering kan se ut (detta är endast ett förslag).

1. För position ”i” i listan ”a”, iterera från första elementet till sista elementet i listan ”a”. Öka ”i” med 1 för varje iteration.
    1.1. Iterera från j = 0 till j = i-1. Öka ”j” med 1 för varje iteration.
        1.1.1. Om talet på position ”i” i listan ”a” är mindre än talet på position ”j”,

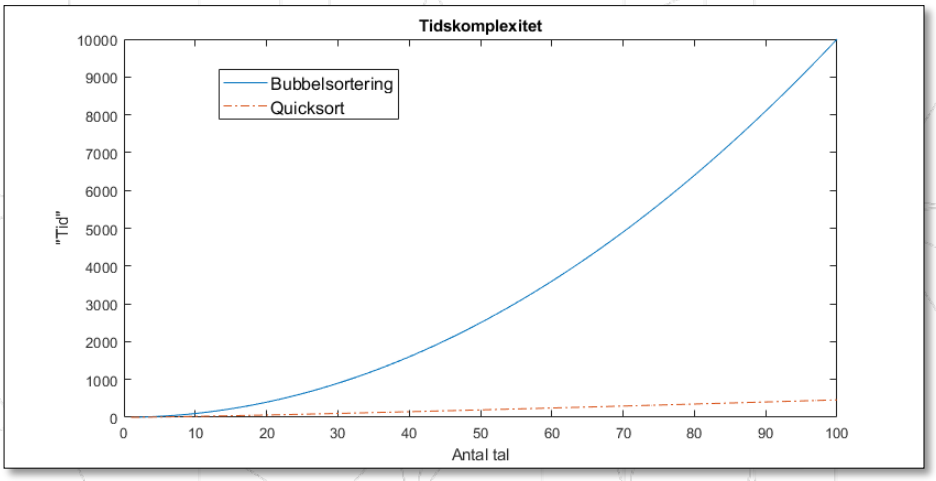
1.1.1.1. byt plats på talet på position ”i” med talet på position ”j”.

I ”korrekt” pseudo-kod kan algoritmbeskrivningen se ut som:

```
funktion bubbleSort( a: lista av tal)
från i = 0 till i < längd(a)
                    från j=0 till j < i
                    om a[i] < a[j]
                    Byt plats på a[i] och a[j]
                    slut
                    slut
slut
returnera a
```

Uppgift 2: Tidskomplexitet

Det teoretiska svaret är att Quicksort har tidskomplexitet $O(n * \log(n))$ och Bubbelsortering $O(n^2)$ i det generella fallet, där n är antalet element (tal) som ska sorteras. För elevernas fall (n = 50 tal) och med den givna koden bör de få att Quicksort är ca 4 gånger snabbare. Teoretiskt borde det vara större skillnad, men i detta praktiska exempel är det inte det. För ännu fler tal sticker skillnaden snabbt iväg då Bubbelsortering ökar med kvadraten medan Quick sort har logaritmisk ökning. Ett illustrativt sätt att visa på skillnaden är att plotta kurvan för tidskomplexiteten för de två algoritmerna (se figuren nedan). Men det finns även specialfall då Quicksort är lika långsam som Bubbelsortering och andra fall då Bubbelsortering är lika snabb eller snabbare än Quicksort (t.ex. om talen redan är sorterade), så det kan skilja sig mycket mellan resultaten de få.



Utmaning: Quicksort

Detta är en ”överkurs”-uppgift och inget som eleverna förväntas att klara utan hjälp. Algoritmen för Quicksort är mer komplex, då vi här även har sk. rekursiva anrop. Med rekursiva anrop menas att en funktion anropar sig själv (normalt med nya parametervärden). Själva algoritmen kan då skrivas väldigt kompakt:

1. Sätt värdet av elementet längst till höger i listan till pivot värde.
2. Partitionera listan i två delar med hjälp av pivot värdet, där elementen i lista 1 är mindre än pivot och elementen i lista 2 är större än eller lika med pivot.
3. Utför Quicksort på lista 1.
4. Utför Quicksort på lista 2.

Koppling till läroplan

Efterarbetets koppling till läroplan för grundskolan, Lgr11

Kursplan – Matematik

Genom undervisningen i ämnet matematik ska eleverna sammanfattningsvis ges förutsättningar att utveckla sin **förmåga** att

- formulera och lösa problem med hjälp av matematik samt värdera valda strategier och metoder,
- föra och följa matematiska resonemang, och
- använda matematikens uttrycksformer för att samtala om, argumentera och redogöra för frågeställningar, beräkningar och slutsatser.

Centralt innehåll i årskurs 7-9

Algebra

- Innebörden av variabelbegreppet och dess användning i algebraiska uttryck, formler och ekvationer.
- Hur algoritmer kan skapas och användas vid programmering. Programmering i olika programmeringsmiljöer.

Problemlösning

- Strategier för problemlösning i vardagliga situationer och inom olika ämnesområden samt värdering av valda strategier och metoder.
- Hur algoritmer kan skapas, testas och förbättras vid programmering för matematisk problemlösning.

Instruktioner elever

En viktig del i att programmera handlar om att kunna läs programmeringskod och förstå algoritmer. Ni ska med den här övningen analysera och förstå en sorteringsalgoritm implementerade i JavaScript som heter Bubbelsortering (eng. Bubble sort). Vi kommer även jämföra den med den snabbare Quicksort algoritmen. Båda algoritmerna är något som man lär sig på en av de inledande kurserna i datavetenskap på universitetet/högskolan och de har historiskt haft stor betydelse, t.ex. för att snabbt sortera sökresultat.

Förberedelser:

1. Ta fram papper och penna.
2. Ta fram datorerna och starta upp en webbläsare.
3. Öppna upp MakeCode för arkad: https://arcade.makecode.com/
4. Bläddra ner till ”Graphics and Math” och öppna projektet ”Sorting Algorithms”.

Nu ska ni ha kommit in i MakeCode där JavaScript koden visas till höger och arkadsimulatorn till vänster. När ni startar simulatorn kommer två av sju sorteringsalgoritmer att slumpas fram. Alla algoritmer sorterar en lista (så kallad array) med slumpmässigt genererade tal.

Initialt test

Börja med att se till att ni ser alla sju sorteringsalgoritmerna. Ni kan lägga till eller ta bort en sorteringsalgoritm genom att ”dra” den vänstra virtuella spaken till höger respektive vänster. Ni startar en ny sortering genom att trycka på A. Ni kan öka eller minska antalet tal som ska sorteras genom att dra spaken upp eller ner.

Testa er fram och se vilken/vilka av de sju algoritmerna som är snabbast. Hur bra är Bubbelsortering och Quicksort jämfört med varandra och de andra? Vilken algoritm är snabbast?

Uppgift 1: Bubbelsortering

Först ska ni analysera en av de enklaste sorteringsalgoritmerna, Bubbelsortering. Gå ner till funktionen bubbleSort (raderna 218-229) i koden till höger:

```
export function bubbleSort(a: number[]) {
```

Genom att studera koden, diskutera tillsammans i er grupp varför och hur algoritmen sorterar talen i listan. I slutet av dessa instruktioner finns det hjälp till vad koden gör/betyder. Använd gärna papper och penna för att sortera en kort

lista av heltal med ”hjälp av” koden. Kan ni lista ut varför algoritmen heter som den gör?

Nu ska ni göra något som heter en algoritmbeskrivning av koden. En algoritmbeskrivning görs oftast i punkt eller numrerad lista, men om ni vill får ni istället skriva den i flytande text. Som numrerad lista skulle en början kunna se ut som (i beskrivande text form):

1. För position ”i” i listan ”a”, iterera från första elementet till sista elementet i listan ”a”. Öka ”i” med 1 för varje iteration.
    1.1. …

Uppgift 2: Tidskomplexitet

Hur bra en algoritm är kan mätas med dess tidskomplexitet. Ni ska göra en enkel jämförelse mellan Bubbelsortering och Quicksort och se hur mycket snabbare Quicksort är för er.

Ni ska börja med att modifiera koden så att ni även får se hur lång tid varje sorterings-algoritm tar på sig (i sekunder). Om ni vill testa själva att göra det så går det bra, annars finns steg-för-steg-instruktioner i slutet av detta dokument.

När ni är klara med förändringarna i koden, starta simulatorn och se till att ni kan se både Bubbelsortering och Quicksort. För att få ett någorlunda korrekt resultat måste vi beräkna medelvärden från flera körningar. Kör minst 5 stycken sorteringar (starta genom att trycka på A) och för varje gång skriv upp tiden det tar för Bubbelsortering respektive Quicksort. Viktigt är att ni inte ändrar antalet tal som ska sorteras, varken i början eller mellan körningarna (om ni har gjort det, starta om arkadsimulatorn och kör om). Nu kan ni beräkna hur många gånger snabbare Quicksort är än Bubblesortering:

1. Beräkna den genomsnittliga tiden Bubbelsortering respektive Quicksort tar på sig för att sortera talen.
2. Nu kan ni räkna ut hur många gånger snabbare Quicksort är än Bubbelsortering.
3. Jämför med de andra grupperna om ni får samma resultat.

Utmaning: Quicksort

Om ni vill ha en utmaning, testa att göra samma analys av Quicksort algoritmen som ni gjorde för Bubbelsorteringen. Koden för Quicksort hittar ni i funktionen `quickSort` någonstans efter rad 257 (vi har ju lagt till några rader i förra uppgiften).

Quicksort är en av de snabbaste generella sorteringsalgoritmerna. Det finns andra som är snabbar men de är då inriktade på specialfall.

Mer info

På Wikipedia finns det bra information om både Bubbelsortering och Quicksort:

https://sv.wikipedia.org/wiki/Bubbelsortering

https://sv.wikipedia.org/wiki/Quicksort