**Keko järjestäminen - Heap Sort**

Määrittely dokumentissa annan kekojärjestämiselle tavoite aikavaativuuden O(n log n) ja toteutuksessa tähän ollaan päästy sillä:

**Heapify:**

- Heapify-operaation suoritusaika riippuu puun korkeudesta ja koska keko on lähes täydellinen binääripuu on keon korkeus O(log n) täten n alkiota sisältävälle puulle heapify pahimman tapauksen aikavaativuus on O(log n)

```
Heapify(A, index)
if (right <= heapSize)
        if(A[left] > A[right])
                largest = left
        else
                largest = right
        if(A[index] < A[largest])
                swap(A, index, largest)
                heapify(A, largest)
```

**BuildHeap**

- BuildHeap-operaation rungossa kutsutaan heapify n/2 kertaa keolle jossa on n alkiota, täten BuildHeap käyttää aikaa korkeintaan O(n log n)

Lopuksi sortin aikana kutsutaan vielä heapify-operaatiota n - 1 kertaa, joten kokonaisuus aikavaativuudeksi saadaan **O(n log n)**

```
Sort(A)
        for i = heapSize down to 2
                swap(1, i);
                heapSize - 1;
                heapify(A, 1)
```

Tilavaativuus kekojärjestämisellä on Heapify-operaation rekursio kutsujen takio **O(log n)**

**Kupla järjestäminen - Bubble Sort**

Määrittely dokumentissa annan kupla järjestämiselle aikavaativuuden $O(n^2)$, ja toteutetussa koodissa aikavaativuus on $O(n^2)$ sillä:

- For-loop sisältää valinnan ja vakioajan vieviä sijoitusoperaatioita. For-loopin runko on siis vakioaikainen O(1)
- For-loopin runko sen sijaan suoritetaan pahimmillaan n - 1 kertaa, sillä for-loop suoritetaan niin kauan että ollaan listan läpi menty kertaalleen ilman että listan sisällä on tehty swappauksia.
- Eli runko suoritetaan pahimmillaan 1 + 2 + 3 + … + (n - 1) täten aikavaativuus on **$O(n^2)$**

Tilavaativuus järjestämiselle on **O(1)** eli vakio, sillä järjestämisen aikana käytetään vain

vakioaikaisia apumuuttujia.

Pseudokoodina kuplajärjestäminen

```
BubbleSort(A)
while not sorted
        sorted = true
        for i = 1 up to A.length - 1
                if(A[i] > A[i + 1])
                        swap(A[i], A[i + 1]
                        sorted = false
```