729G46

Lektion 4 - sorteringsövning

Jody Foo, jody.foo@liu.se

LINKÖPINGS UNIVERSITET

VPN för anslutning utanför LiU

Hej!

Med start **2022-11-01** kommer **thinlinc.edu.liu.se**. **ssh.edu.liu.se** och **rdpklienter.edu.liu.se** ändras så att de endast kan nås med VPN om anslutningen kommer från internet in till universitet.

Studenter som ansluter hemmifrån behöver använda **FortiClient** och ansluta till remote.edu.liu.se innan de kan ansluta med ThinLinc, SSH, RDP.

OBS! Tjänsten remote.edu.liu.se tillåter endast anslutningar från studenter (eller anställdas student_* konton) till precis ThinLinc, SSH, RDP.

Den här förändringen görs för att öka säkerheten på universitet genom att få alla användare att autentisera med pin-kod när de ansluter till universitetet istället för endast pushnotis i Microsoft-authenticator.

Mer information om förändringen kommer att publiceras inom kort på **https://www.student.liu.se/studentstod/itsupport?l=sv**

LINKÖPINGS UNIVERSITET

Lektionsupplägg

- **Gruppindelning**
  - uppdelning i halvor
  - grupper på 3-5 personer i varje halva
- **Genomgång**
  - övning
  - repetition av range()
  - värde eller index - var uppmärksamma!
- **Utdelning av material**
- **Övning**
  - Klura i grupp
  - Rast
  - Förklara för annan grupp + få förklaring från annan grupp
- **Gemensam genomgång av algoritmerna**

Gruppindelning

Två halvor (fördela er jämt)

Bilda grupper på 3-5 personer i varje halva

Övning

- Varje grupp får en känd sorteringsalgoritm på papper och fem platsmuggar
- Målet är att:
  - förstå hur algoritmen gör för att sortera en lista med heltal
  - kunna förklara algoritmen (med ord) för en annan grupp
- *Datorer/telefon får inte användas för att köra koden.*

- **Efter rasten** ska ni **förklara hur er algoritm** fungerar för en grupp i den andra halvan + **få en algoritm förklarad** för er.
- **Om ni blir klara innan rasten**, säg till så får ni en extra algoritm

Saker att tänka på under övningen

- När ni ser ett `range()`-uttryck. **Se till att ni förstår vilka värden ni kommer få från det**.
- `range(stop)` - implicit: `start=0`, `step=1`
- `range(start, stop)` - implicit: `step=1`
- `range(start, stop, step)` - alla värden explicita

- `step > 0`: heltal från start, fortsätt så länge som heltalen är *mindre* än `stop`
- `step < 0`: heltal från start, fortsätt så länge som heltalen är *större* än `stop`

Exempel med range()

```
# range() returnerar ett range-objekt.
# man kan få en lista om man skickar range-objektet till list()

# -> [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
list(range(10))

# -> [3, 4, 5, 6, 7]
list(range(3, 8))

# -> [4, 6, 8]
list(range(4, 10, 2))

# -> [8, 7, 6, 5, 4]
list(range(8, 3, -1))

# -> [12, 9, 6, 3]
list(range(12, 2, -3))
```

Saker att tänka på under övningen

- Var uppmärksamma på om variabler som används som **index** används som **faktiska värden** eller om de används för att **plocka fram ett värde på ett visst index**.

- `index1 = 6`
- `alist[index1] > max_val`
- `cur_val = alist[index1]`
- `index1 < other_index`

Saker att tänka på under övningen

- **Skriv upp vad olika variabler har för värden** så att ni inte behöver hålla det i huvudet.
- Använd muggarna som representation av er lista, men var uppmärksamma på att **för vissa algoritmer så kan det finnas tillfällen där ett värde temporärt förekommer på mer än ett ställe**.

Övning

Delas ut: papper med sorteringsalgoritm, muggar

Ta fram papper & penna att använda i gruppen (kladdpapper finns)

Saker att tänka på under övningen. Rast slut 11:15

- När ni ser ett `range()`-uttryck. Se till att ni förstår vilka värden ni kommer få från det.
- Var uppmärksamma på om variabler som används som **index** används som **faktiska värden** eller om de används för att **plocka fram ett värde på ett visst index**.
- **Skriv upp vad olika variabler har för värden** så att ni inte behöver hålla det i huvudet.
- Använd muggarna som representation av er lista, men var uppmärksamma på att **för vissa algoritmer så kan det finnas tillfällen där ett värde temporärt förekommer på mer än ett ställe**.

- **Om ni blir klara innan rasten**, säg till så får ni en extra algoritm

Sorteringsalgoritmerna

sort_list1: **Selection Sort**

sort_list2: **Insertion Sort**

sort_list3: **Bubble Sort** (extra)

sort_list1 - Selection Sort

- Hitta största värdet, flytta det till slutet av listan
- Hitta näst största värdet, flytta det till näst sista platsen i listan
- osv

sort_list1 - Selection Sort

```
def sort_list1(alist):
    """Sortera värden i alist i stigande ordning."""
    # Sortera alist bakifrån, störst värde hamnar på fill_pos
    # fill_pos börjar längst till höger i listan och vandrar åt vänster
    for fill_pos in range(len(alist)-1, 0, -1):
        pos_of_max = 0

        # Leta efter största värdet mellan index 1 och fill_pos
        for position in range(1, fill_pos+1):

            # Spara position för högsta påträffade värdet
            if alist[position] > alist[pos_of_max]:
                pos_of_max = position

        # Byt plats på värdena på fill_pos och pos_of_max
        temp = alist[fill_pos]
        alist[fill_pos] = alist[pos_of_max]
        alist[pos_of_max] = temp

    # Anmärkning: funktionen ändrar i existerande lista. Inget returvärde behövs.
```

sort_list2 - Insertion Sort

- Gå igenom listan från vänster till höger, börja med det andra värdet, sen det tredje osv.
- För varje aktuellt värde, flytta alla värden till vänster om det åt höger tills alla värden som är mindre än det aktuella värdet ligger till vänster om det aktuella värdet

sort_list2 - Insertion Sort

```
def sort_list2(alist):
    """Sortera värden i alist i stigande ordning."""
    # Varje iteration flyttas infogas värdet på index på den plats som göra att så att alla
    # värden till vänster om det är mindre än det värdet.
    #
    # index börjar på andra värdet listan och vandrar åt höger i efterföljande iterationer
    for index in range(1, len(alist)):
        # currentvalue är det värde som vi ska hitta rätt plats för
        # vi börjar leta från position
        currentvalue = alist[index]
        position = index

        # Kopiera värdet på position-1 till position (dvs kopiera ett steg åt höger).
        # Loopen vandra åt vänster i listan och fortsätter så länge som vi stöter på
        # värden som är större än currentvalue.
        #
        # Efter loopen har vi gjort plats för currentvalue på platsen position
        while position > 0 and alist[position-1] > currentvalue:
            alist[position] = alist[position-1]
            position -= 1

        # Kopiera current value till index position i alist (där det nu finns en lucka)
        alist[position] = currentvalue

    # Anmärkning: funktionen ändrar i existerande lista. Inget returvärde behövs.
```

sort_list3 - Bubble Sort

- Vi går igenom listan från vänster till höger.
- Vi tittar på par av värden och byter ev. plats på dessa så att det större värdet flyttas åt höger och det mindre värdet flyttas åt vänster.
- Efter första genomgången ligger det största värdet sist.
- Vi upprepar processen för att flytta det näst största värdet till den näst sista platsen osv.

sort_list3 - Bubble Sort

```
def sort_list3(a_list):
    """Sortera värdena i a_list i stigande ordning."""
    # Gå igenom listan från slutet av listan (index i), efter varje iteration kommer
    # värdet på index i att vara på sin korrekta plats. Nästa iteration tittar vi på
    # indexet ett steg till vänster.
    for i in range(len(a_list) - 1, 0, -1):
        # Gå igenom listan från index j till i, flytta det största värdet åt höger.
        # Efter loopen kommer det största värdet mellan index i och j att vara på
        # rätt plats (på index i).
        for j in range(i):
            # Se till så att elementen på plats j och j+1 står i stigande ordning.
            if a_list[j] > a_list[j + 1]:
                # Byt plats på elementen på index j och j+1 om det behövs
                temp = a_list[j]
                a_list[j] = a_list[j + 1]
                a_list[j + 1] = temp
```

Att tänka på till Algoritmrapport 1 (kommentera kod)

- Orden "framåt" eller "bakåt" är ibland tvetydiga; t.ex.

  Framåt (=vänster) som mot början av listan? Bakåt (=höger) som i mot slutet av listan?

  Framåt (=höger) som "senare" om man läser listan från vänster till höger? Bakåt (=vänster) som "tidigare" om man läser listan från vänster till höger?

  Framåt som i loopens riktning (=vänster/höger)? Bakåt (=vänster/höger) som "där vi har varit tidigare"?

- Vänster och höger är entydiga.
- Var noga med att skilja på

  "värdet på index i" och

  "index i"

- Skriv t.ex. **inte** *"om i är mindre än j"* om ni menar *"om värdet på index i är mindre än j"* (samma sak om ni menar *värdet på index j*).