Kodutöö 2

Ül 1. Bubble Sort

Antud on järgnevate arvude loend: [64, 34, 25, 12, 22, 11, 90]. Simuleeri samm-sammult Bubble Sort algoritmi. Iga läbimise järel kirjuta üles tulemuseks olev loend.

Bubble Sort liigub vasakult paremale ja võrdleb kõrvuti asetsevaid elemente, tõstes kõrgema elemendi paremale poole.

[64, 34, 25, 12, 22, 11, 90]

Peale esimest läbimist:
[34, 25, 12, 22, 11, 64, 90]

Peale teist läbimist:
[25, 12, 22, 11, 34, 64, 90]

Peale kolmandat läbimist:
[12, 22, 11, 25, 34, 64, 90]

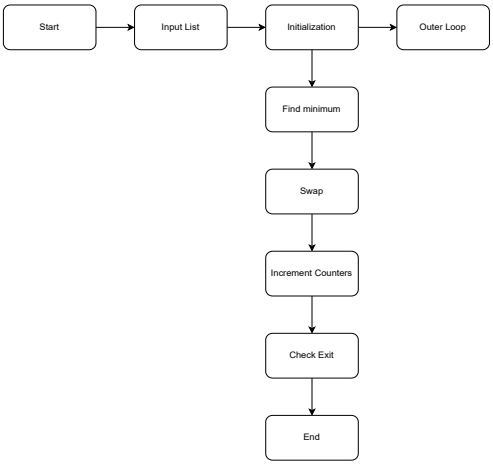
Peale neljandat läbimist:
[12, 11, 22, 25, 34, 64, 90]

Peale viiendat läbimist:
[11, 12, 22, 25, 34, 64, 90]

Arvude loend sai sorteeritud.

Ül 2. Selection Sort

Joonista plokkskeem, mis esindab Selection Sort algoritmi. Kasutades seda vooluskeemi, demonstreeri sortimisprotsessi loendile: [29, 15, 56, 77, 18]. Tuvasta iga iteratsiooni ajal väikseim element.



Loend: [29, 15, 56, 77, 18]

Iteratsioon 1:
Min 15 indeks 1
[15, 29, 56, 77, 18]

Iteratsioon 2.
Min 18 indeks 4
[15, 18, 56, 77, 29]

Iteratsioon 3.
Min 29 indeks 4
[15, 18, 29, 77, 56]

Iteratsioon 4.
Min 56 indeks 4
[15, 18, 29, 56, 77]

Lõpp

Ül 3. Insertion Sort
Antud on osaliselt sorteeritud loend: [12, 11, 13, 5, 6, 7]. Rakenda Insertion Sort algoritmi. Selgita, miks Insertion Sort võib olla tõhusam sellele loendile võrreldes täielikult sortimata loendiga.

Kuna insertion sort loeb elemente ükshaaval ja paigutab need kohe õigesse järjekorda sobib see pooleldi juba sorteeritud loendile paremini, kui sorteerimata loendile kuna see võtaks veel kauem aega.
Sellel osaliselt sorteeritud loendil peab ära vahetama ainult 11 ja 12 ja 5, 6, 7 kõige ette tõstma.
[12, 11, 13, 5, 6, 7]
Esimene:
[11, 12, 13, 5, 6, 7]
Teine:
[5, 11, 12, 13, 6, 7]
Kolmas:
[5, 6, 11, 12, 13, 7]
Neljas:
[5, 6, 7, 11, 12, 13]
Lõpp

Ül 4. Ajakompleksuse analüüs

1) Millistel eelmainitud sortimisalgoritmidel on halvimal juhul ajakompleksus O(n^2)?

   Kõigil kolmel(Bubble, selection, insertion sort) on halvimal juhul ajakompleksus O(n^2).

2) Milline sortimisalgoritm oleks kõige sobivam sortimaks loendit täisarvudega, mis jäävad vahemikku 1 kuni 100 ja miks?

   Kõige sobivam sellisel puhul oleks Counting Sort. See on efektiivne täisarvude sortimisel kui on juba teada ja lõplikus vahemikus loend. Selle ajaline keerukus on O(n + k), kus n on sortitavate elementide arv ja k on sisendväärtuste vahemik.

   Miks see on kõige parem? See on tõhus kuna ta ei võrdle omavahel sorditavaid elemente vaid selle asemel loendab sisendloendis iga täisarvu esinemise ja paigutab need järjestatud järjekorda.
   Lisaks on see stabiilne, lineaarne ajaline keerukus ja tal pole erinevalt QuickSortist või MergeSortist täiendavat mälu vaja.

Ül 5. Stabiilsus ja adaptiivsus

1) Defineeri, mida tähendab, et sortimisalgoritm on "stabiilne." Anna näide stabiilsest sortimisalgoritmist antud loendist.

   Sorteerimisalgoritmi peetakse "stabiilseks", kui see säilitab sorteeritud väljundis võrdsete elementide suhtelise järjekorra, nagu need olid algses sortimata sisendis. Teisisõnu, kui sisendis on kaks sama väärtusega elementi ja üks kuvatakse enne teist, tagab stabiilne

sortimisalgoritm, et esimesena ilmunud element on järjestatud väljundis siiski teisest elemendist ees.
Antud loendis on stabiilne näiteks Insertion Sort
Näide:
Algne loeng: Algne loend: [(4, A), (2, B), (1, C), (4, D), (3, E)]
Sorteerime
Lõplik loend:[(1, C), (2, B), (3, E), (4, A), (4, D)]
Näeme, et (4,A) ja (4,D) on omavahel ikka samade kohtade peal.

2) Selgita "adaptiivsuse" kontseptsiooni sortimisel. Millistest eelmainitud sortimisalgoritmidest peetakse adaptiivseks?

   Sortimisalgoritm on adaptiivne, kui see kasutab ära juba 'sorteeritud' elemente järjendis, mida tuleb sorteerida. Läbi selle on tõhusam. Eelmainitud sortimisalgorütmidest on adaptiivne ainult Insertion Sort, kuna see suudab kohanduda sisendandmete järjestusega ja töötada kiiremini juba eelnevalt sorteeritud või peaaegu sorteeritud andmete puhul.

Boonus

Kuidas näeks loend [8, 3, 5, 4, 7, 6, 2] välja pärast esimest läbimist antud algoritmide puhul
a) Bubble Sort [3, 5, 4, 7, 6, 2, 8]

b) Selection Sort [2, 3, 5, 4, 7, 6, 8]

c) Insertion Sort [3, 8, 5, 4, 7, 6, 2]

Matthias Paluteder 30.10.23