**Oblig 3: 15.3.2024**

**Medlemmer:**

Yasmin Maria Hope

Alexandra Davanger Berdal

Daniel Havsgård

Nedenfor finner vi oppgavene for analyse: D og E.

Oppgave D: Analyse.

**Kjøretids analyse for TabellMengde**

|  |  |  |
| --- | --- | --- |
| **Metoder** | **Kjøretid: Best case** | **Kjøretid: worst case** |
| Boolean inneholder (T element) | **O (1)**  Fordi at det må iterere gjennom alle elementene i mengden for å sjekke om ‘element’ finnes. Den sammenligner ‘element’ med første element i tabellen. Og returnerer true om den finnes. Uten å måtte iterere gjennom resten av tabellen. | **O(n)**  Fordi det kan hende at ‘element’ ikke eksisterer i mengden, dermed må metoden iterere gjennom hele tabellen og sammenligne ‘element’ med hvert element i mengden. Og dette blir en lineær tidskompleksistet, n er antall elementer i mengden. |
| Boolean erDelmengdeAv (MengdeADT<T> annenMengde); | **O (1)**  Hvis begge mengdene er tomme, slipper man å gjøre sammenligninger, dermed blir det raskt beregnet ut at den ene mengden er delmengde av ‘annenMengde’. | **O(n\*m)**  Hvis den ene mengden og ‘annenMengde’ har elementer som ikke er like, for hvert element må det gjøres en sammenligning i den nåværende mengden (n elementer) og ‘anneMengde’ (m elementer). |
| Boolean erLik (MengdeADT<T> annenMengde); | **O (1)**  Når antallet elementer i begge mengdene er ulike. Blir det ikke nødvendig med sammenligning av alle elementene og metoden kan raskt avslutte, som fører til O (1) som best case. | **O(n\*m)**  Når begge mengdene har samme antall elementer, vil metoden sammenligne alle elementene med hverandre for å finne om de er like. Dermed blir worst case O(n\*m). |
| MengdeADT<T> union (MengdeADT<T> annenMengde); | **O(n)**  Når én av mengdene er tom. I denne metoden trenger bare metoden å kopiere elementene fra den ikke-tomme mengden til resultatmengden og returnere den. | **O(n\*m)**  Når begge mengdene har flere unike elementer eller er ikke tomme, og for hvert element i den ene mengden må det sjekkes om elementet allerede finnes i resultatmengden, dermed blir lineært søk gjennom mengden for hvert element. |
| T fjern (T element); | **O (1)**  Når elementet er første element i tabellen, dermed kan ‘element’ fjernes raskt fra mengden ved å flytte de andre elementene i tabellen. | **O(n)**  Når elementet er det siste elementet i mengden, eller ikke finnes i mengden. Metoden må iterere gjennom alle ‘element’ og så fjerne det. |

**Kjøretids analyse for LenketMengde**

|  |  |  |
| --- | --- | --- |
| **Metoder** | **Kjøretid: Best case** | **Kjøretid: worst case** |
| Boolean inneholder (T element) | **O(1)**  Oppstår når listen søker og finner det bestemte elementet på første plass i listen. | **O(n)**  Når det bestemte elementet ikke finnes i listen, eller finnes på siste plass. N elementer må sammenligne bestemte element med alle andre elementer i listen. |
| Boolean erDelmengdeAv (MengdeADT<T> annenMengde); | **O(n)**  Når nåværende mengde er delmengde av ‘annenMengde’. Dermed må metoden iterere gjennom elementene til nåværende mengde for å sjekke om hvert element også finnes i den andre. | **O(n\*m)**  Når nåværende mengde ikke er delmengde av ‘annenMengde’. N blir elementer i nåværende mengde, mens m er elementer i ‘annenMengde’ som må sammenlignes. |
| Boolean erLik (MengdeADT<T> annenMengde); | **O(n)**  Når begge mengdene er helt like. Må kun iterere gjennom listen 1 gang for å sammenligne elementene. | **O(n\*m)**  Når mengdene er forskjellige, må fortsatt iterere gjennom listen for å sammenligne elementene. N-antall elementer i ene mengden. M antall elementer i ‘annenMengde’. |
| MengdeADT<T> union (MengdeADT<T> annenMengde); | **O(n)**  Når en av mengdene er tom. Legger til elementene fra den ikke-tomme mengden til ‘resultatMengde’. | **O(n\*m)**  Når både den ene mengden og ‘annenMengde’ har elementer i seg. Og hvert element må legges til i ‘resultatMengde’, og for hver ‘n’ element (første mengden) må det sjekkes med et annet ‘m’ element i resultatmengden. |
| T fjern (T element); | **O (1)**  Når søkte element er det første elementet i listen. | **O(n)**  Når det søkte elementet som skal fjernes, er det siste eller ikke finnes i det hele tatt. Må gå igjennom hele listen og søke etter elementet. N er antall elementer. |

e) Analyse:

Siden TabellMengde og LenketMengde er implementert av MengdeADT, bruker de enten et array eller en lenket liste for å lagre elementene, når det kommer til HashSet og TreeSet, bruker de egne implementasjoner og struktur for å lagre elementer.

For eksempel hashSet bruker en egen hash tabell, og noen ting skiller den fra de andre. Blant annet kan det oppstå hash-kollisjoner , som kan føre til at tidskompleksiteten kan bli forverret, i hashTabell vil verste fall om dette skjer føre til O(n). Hvis vi tar hash tabell i utgangspunkt i til metoden contains( T element) blir gjennomsnittstiden O(1).

Men for eksempel TreeSet, bruker Red. Black tree, som er balanserte binære søketrær som opprettholder balanse mellom venstre og høyre subtrær for hver node. Kjøretiden for contains() har en tidskompleksitet på O(n log n).

Men til sammenligning ser vi at TabellMengde og LenketMengde gir en mindre effektiv ytelse for contains()- metoden enn hashSet og TreeSet, som bidrar til en mer effektiv sortering.

Kjøring av Oppgave F)

Et bilde som inneholder tekst, Font, skjermbilde

Automatisk generert beskrivelse