Aflevering 3 af Anders J.R. Petersen

I denne opgavebesvarelse beskrives og analyseres udførselstiden af et Java program, der løser den tredje afleveringsopgave i kurset "Algoritmer og Datastrukturer, efterår 2021". Opgaven går ud på at designe et program, der kan afgøre hvorvidt det er muligt at ankomme til et busstoppested fra et andet busstoppested, på et rutebil netværk, "samme dag", "en anden dag" eller "slet ikke". Busstoppesteder kaldes herefter stationer.
Som beskrevet i opgaveteksten, opererer programmet med et tekstinput bestående af følgende tre dele:

1. *Første linje af fire heltal.* ***n: antal stationer, m : antal afgange, q : antal forespørgsler og s : start station***
2. *De næste m linjer indeholder fire heltal, der beskriver hver afgang.* ***i : start-station, j : endestation, a : afgangstid og b : ankomsttid***
3. *Til sidst kommer der q linjer med forespørgsler, der indeholder et tal, som spørger om det tidligste ankomsttidspunkt.* ***t : station***

Delopgave 1: Beskrivelse af løsningen

Programmet opbygger først 2 rettede grafer. Den første rettede graf, kaldet **total-graf**, indeholder en knude for hver station og en kant for hver afgang. Den anden rettede graf, kaldet **dags-graf**, indeholder en knude for hver busrutes ankomsttidspunkt til en station. Kanter mellem knuder i dags-grafen svarer til busruter hvis afgangstidspunkt er mindre eller lig startknudens tid og ankomsttiden er lig endeknudens tid.
For hver station oprettes en "binær-hobsbaseret prioritetskø" med dags-graf-knuder, prioriteret efter laveste tidspunkt.
Der udføres en dybde først søgning på hver graf. Hvis det er muligt at nå en dags-graf-knude for station **t** fra **s**, udskrives den mindst mulige ankomsttid fra prioritetskøen. Hvis det kun er muligt at nå station **t** fra **s i** total-grafen udskrives "næste dag". Hvis station **t** ikke kan nås, både i dags-grafen og total-grafen, udskrives "jeg køber sku en bil".

Delopgave 2: Java programmet

Se vedlagt kode eller se: https://github.com/KursusIAlgoritmer/Handin3/blob/main/HandIn3.java

Delopgave 3: Kørselstidsanalyse af programmet

Kørselstiden af programmet afgøres hovedsageligt af den tid det tager at afvikle følgende faser.
1.) Oprettelse af en liste med prioritetskøer, svarende til knuder i dags-grafen.
3.) Oprettelse af dags-graf og total-graf.
4.) Dybde først søgninger i graferne.
5.) Håndtering af forespørgsler.

1.) Først oprettes listen til af prioritetskøer, hvilket koster **n** tid**.**
I dags-grafen er antallet af knuderne det samme som antallet af ankomster plus en ekstra knude, der angiver startstation og tid 0. Dette giver **m+1** knuder. Da hver knude indsættes i en prioritetskøer koster det i værste fald **(m+1)*(1+lg(m+1))** i alt.
Omkostning worst-case: **n + (m+1)*(1+lg(m+1))**

2.) Dags-grafens antal af kanter, kaldet **md**, synes ret svært at sætte på formel, men er bestemt både af afgangs og ankomsttider og antallet af disse. Da **E =V*(V-1)** for en fuldt forbundet rettet graf, hvor E er antal kanter og V antal knuder, må det gælde at **md** sandsynligvis er mindre end **m*(m+1)**.
Oprettelsen af en rettet graf koster i tid E + V tid. Antallet af knuder og kanter i dags-grafen er allerede beskrevet.
I total-grafen er antallet af knuder svarende til de **n** stationer , og antallet af kanter svarende til de **m** afgange.
Oprettelse af dagsgraf: ***m+1 + md*** *Oprettelse af total-graf:* ***n + m***

3.) Dybde først søgningerne i en rettet graf koster ligeledes E +V tid.
DFS af dagsgraf: ***m+1 + md*** *DFS af total-graf:* ***n + m***

4.) Hentning af prioritetskøen koster 1 for hver forespørgsel, dvs. **q** tid ialt.
Best case koster det 1 for hver prioritetskø dvs. **q** tid, i værste fald svarende til antal dags-grafen-knuder, dvs. i alt **q*(m+1)**.
O*pslag i listen af prioritetskøer:* ***Worst case : q + q*(m+1)*** *og* ***Best case = 2*q***

Samlet køretid vurdering:
Inputtet består af mange forskellige parametre, men den, der har størst betydning for kørselstiden er m, antallet af afgange. Antallet af stoppesteder n, indgår kun i et lineært led.

Det ses tydeligt af følgende, at m har størst betydning for kørselstiden (n holdes konstant) :

I del.1 har vi:		O(m*lg(m+1))
I del 2 og 3 har vi, hvis md = m*(m+1)	:	O(m^2)
I del 4. har vi i allerværste fald		O(q*m)

Overordnet kan den allerværste kørselstid for løsningen beskrives med O(m^2)