Drosjesentralen

I-120: Obligatorisk oppgave 2, 2000

Frist Mandag 20. November 2000 kl.10:00, i skuff merket I120 på UA.

Krav Se seksjon 4 for kravene til innlevering. Merk krav om generisk løsning for pris-ordre og tids-ordre.

Individuellt arbeid Hver student skal selv skrive inn og debugge all sin egen kode. Dersom to innleverte programmer viser seg å være bortimot kopier vil begge parter få 'Ikke godkjent'. Du er selv ansvarlig for at andre ikke kopierer din kode. Opprett derfor en katalog Oblig2 hvor du utvikler dine programmer og kjør Unix-kommando

```
setfacl -s group:i120-adm:r-x,user::rwx,group::---,mask:rwx,other:--- Oblig2
```

(bruk 'copy-and-paste' fra seksjon 'meldinger' på kursets hjemmeside hvor du vil finne denne kommando-strengen.) Da kan filene i Oblig2 aksesseres kun av deg selv og 120-gruppelederne. Kommandoen fungerer på Sun-maskinene på UA.

Gruppearbeid Under selve designprossessen, dvs planlegging for hånd av programmet, vil det være lov å jobbe i grupper på høyst tre personer, og hver av innleveringene skal i dette tilfellet merkes med navnet på partnere.

Programvare Bruk Java. Alle moduler fra felles-området til I120 kan benyttes.

1 Bykart

Vi er gitt et bykart med markerte drosjeholdeplasser og veier som forbinder disse. En *vei* er en direkte forbindelse mellom to holdeplasser (som ikke passerer gjennom andre holdeplasser). En *rute* er en muligens indirekte veiforbindelse mellom to holdeplasser (en vei er en rute men en rute vil, generelt, bestå av flere veier som forbinder endeholdeplasser og passerer gjennom flere andre holdeplasser). Det kan finnes mange ruter mellom to gitte holdeplasser og vi antar at for alle par av holdeplasser finnes det minst en rute som forbinder disse.

Veier er enveiskjørt. Dersom det finnes en toveis direkte forbindelse mellom to holdeplasser A og B, regner vi det som to direkte veier: en fra A til B og en fra B til A. Vi antar (for enkelhets skyld) at det alltid finnes høyst én vei i hver retning mellom to holdeplasser.

Avstand, Kjøretid og Pris

For hver vei har vi en avstand (lengde) og kjøretid. Utfra veis lengde og kjøretid beregner drosjesentralen prisen – $pris = 5 * avstand + kjøretid$ (i minutter).

Ruter

I oppgaven vil vi måtte finne ruter mellom to angitte holdeplasser utfra forskjellige kriterier. En kjørerute er entydig identifisert gjennom: avgangs- og ankomstholdeplass, samt alle mellomholdeplasser. Spesielt antar vi at en rute kjøres kontinuerlig uten noen opphold på veien:

drosjer som er i rute stopper aldri et sted og venter for så å kjøre videre. En rute vil i tillegg aldri inneholde løkker: en drosje på rute fra A til B vil besøke en vilkårlig holdeplass C *høyst* én gang.

Kjøretiden på en rute er summen av kjøretider langs alle veier som utgjør ruten og prisen for en rute er summen av priser for alle veier langs ruten. Når man skal se på kjøretid, pris og ankomsttid for en rute må man, i tillegg til avgangs-, mellom- og ankomstholdeplasser, ha tidspunktet når kjøring starter.

Et eksempel: Anta at vi har tre holdeplasser A, B, C samt veier $A - B$ med avstand 5km og kjøretid 15 minutter, og $B - C$ med avstand 8km og kjøretid 12 minutter. Beregninger for ruten $A - B - C$ som starter kl.13:50 blir da:

- Total kjøretid: 12+15=27 minutter.
- Total pris: 5*(8+5)+27=92 kroner.
- Ankomsttid: 13:50+27=14:17.

2 Bestillinger

Drosjesentralen har et fast (men nærmere uspesifisert) antall drosjer. Disse er plassert forskjellige steder i byen og til enhver tid er denne plassering tilgjengelig gjennom datasystemet. En drosje kan enten befinne seg på en av holdeplassene eller være i rute til en bestemt holdeplass. I begge tilfellene holder programmet kun informasjon om hvor (hvilken holdeplass) drosjen blir ledig og ved hvilket tidspunkt (se kommentar om forenklinger lenger nede).

Drosjebestillinger kommer til sentralen og er behandlet i den rekkefølgen de ankommer. Behandling av en drosjebestilling (som vi kaller "ordre") består av to deler:

1. velge en drosje som skal sendes til å betjene ordren
2. velge en rute som drosjen skal så bruke for å frakte kunden til hans mål

Vi gjør en del forenklinger i forhold til en virkelig situasjon:[1]

- For det første antar vi at for å betjene en ordre trengs kun én drosje og at det, i prinsippet, kan være hvilken som helst drosje som er ledig (modulo noen begrensinger som vi beskriver senere) – mao. overser vi mulige sitteplassbegrensinger.
- For det andre gjelder bestillinger alltid to holdeplasser; vi ignorerer muligheten for at man vil ha drosje tilbake og antar at drosjer alltid kjører fra en holdeplass til en annen.
- Drosjer plukker ikke passasjerer på gaten men betjener kun ordrene mottat fra sentralen.
- Vi antar at alle bestillingene kommer til sentralen på et bestemt tidspunkt – 0:0:0 (se **Tid** i seksjon 3) og gjelder et tidspunkt som er "senere enn" dette. (I det følgende mener vi med "bestillingstid" tiden en drosje er bestilt *til*, dvs. det tidspunktet en drosje er ønsket av kunden på en bestemt avgangsholdeplass.) Programmet skal bare planlegge hvordan drosjene skal sendes rundt gitt disse bestillingene.
- En drosje som er bestilt for å betjene en ordre fra holdeplass A til holdeplass B ved bestillingstid t regnes som opptatt *hele tiden* til den er ferdig med denne ordren. Med andre ord: en drosje kan ikke brukes til å betjene en ny ordre, før den er ferdig med sin siste ordre.

[1] Du står fritt til å eventuelt ta hensyn til mer realistiske situasjoner og forbedre programmet ditt slik at det behandler disse forenklingene på en mer realistisk måte, men dette er ikke krav til oppgaven.

F.eks. hvis en drosje er opptatt fra kl.10:10 og blir ledig klokken 13:15, og den tilordnes til å betjene en ordre kl.18:40 som skal avsluttes kl.19:20, regner vi denne drosjen som opptatt hele tiden fra 10:10 til 19:20, dvs. opptatt mellom 10:10 og 13:15 (fra før), opptatt mellom 18:40 og 19:20 (ordre) men *også* opptatt mellom 13:50 og 18:40!

I det følgende skriver vi $o = (A, B, t)$ for en ordre der kunden vil bli kjørt fra holdeplass A til B og starte kjøring ved tidspunktet t (evt. *ankomme* ved t for *reverserte ordrer* — se seksjon 2.2). En drosje skriver vi som $d = (D, t)$, der D angir den holdeplassen der drosjen blir ledig ved tidspunktet t.

2.1 Valg av drosje

Når man skal velge en drosje for en gitt ordre $o = (A, B, t)$, skal man rangere alle drosjer og velge "den beste" utfra følgende kriterier. For det første må drosjer være istand til å ankomme A ikke senere enn t. Blant alle slike, vil en drosje $d_1 = (D_1, t_1)$ velges fremfor $d_2 = (D_2, t_2)$, hvis avstanden $D_1 - A$ er mindre enn avstanden $D_2 - A$. Hvis det er flere drosjer med samme kjøreavstand til A, velger man vilkårlig en av dem.

2.2 Valg av rute

Det er tre typer ordrer, og minst to av disse må implementeres med en generisk algoritme – d.v.s. at minst to av disse bruker den samme algoritmen og at denne algoritmen gis et objekt som kan brukes til kalkulere verdien av enkelte algoritme-parametre for ordre-typene.

ot *tids* ordre (A, B, t) ber om en drosje fra holdeplass A til holdeplass B med starttidspunkt t og kortest mulig kjøretid $A - B$;

op *pris*-ordre (A, B, t) ber om en drosje fra holdeplass A til holdeplass B med starttidspunkt t og billigst mulig total pris for hele ruten $A - B$ (kunden betaler ikke for drosjes kjøring fram til A);

or *reversert* ordre (A, B, t) ber om en drosje fra holdeplass A til holdeplass B men slik at B skal ankommes ikke senere enn t.

I alle tilfelle kan det hende at ingen drosje rekker å betjene ordren (ingen klarer å ankomme avgangsholdeplass tidsnok). I så fall skal programmet gi beskjed om det. (Vi antar at ingen ordre bestiller en drosje til en tid som er "mindre enn" 0:0:0. Hvis en **or** ordre innebærer at en drosje måtte starte før dette tidspunktet, anses det for umulig.)

Følgende observasjoner skal være nyttige:

- Det kan godt hende at en **op** ordre (A, B, t) skal betjenes gjennom en rute som tar lengre tid enn en tilsvarende **ot** ordre (A, B, t).
- For å betjene en **or** ordre (A, B, t), må man finne *senest mulig* tidspunkt t' slik at man kan starte kjøring fra A ved t' og ankomme B ikke senere enn t. Tidspunktet t' må bestemmes før man kan begynne å velge drosje. Ruten for å betjenne ordren kan mest naturlig velges ved å se på kortest mulig kjøretid.

Husk også at ruter ikke inneholder noen løkker eller opphold som nevnt på slutten av seksjon 1.

3 Hovedprogrammet

Hovedprogrammet skal kunne håndtere følgende kommandoer:

1. `g fil` – les bykart, drosjer og drosjeplassering fra fil med navn `fil`
2. `d` – skriv alle drosjer (sortert etter registreringsnummer) på skjermen; for hver drosje skriv ved hvilken holdeplass den blir ledig og når
3. `h` – skriv alle holdeplasser (sortert etter nummer; se 3.1) på skjermen; for hver holdeplass skriv ut alle drosjene som er (eller blir) ledig der sortert etter tidspunktet (når drosjen er ledig)
4. `t A B tid` – betjen tids-ordre (A, B, tid)
5. `p A B tid` – betjen pris-ordre (A, B, tid)
6. `r A B tid` – betjen reversert ordre (A, B, tid)
7. `f navn` – filen "*navn*" inneholder en serie med kommandoer og deretter ordre (på form som angitt over); disse skal utføres én etter en i den rekkefølgen de står på filen.

For hver ordre (kommandoene 4–6, også når disse kjøres fra en fil som i siste punkt), skal programmet skrive ut: hvilken drosje som skal betjene ordre (og fra hvilken holdeplass den skal kjøre til A), deretter ruten den skal følge for å betjene ordren, avgangs- og ankomsttidspunkt, samt total tid og pris for denne ruten.

3.1 Data og dataformat

Hver drosje er identifisert entydig ved sitt registreringsnummer som består av noen bokstaver etterfulgt av noen siffre (en sammenhengende streng uten mellomrom eller andre skilletegn).

Hver holdeplass er identifisert entydig ved et positivt heltall h med $1 \leq h \leq N$ der N er totalt antall holdeplasser.

Tid

Tid representeres ved tre ikke negative heltall atskilt med et tegn – d:h:m – der d er dagen; $0 \leq h \leq 23$ er timen og $0 \leq m \leq 59$ er minutter. Du kan bruke noen tidshåndteringsrutiner, f.eks. `i120.TimeDHM` (se dokumentasjon for denne pakken); alternativt kan du skrive dine egne metoder. Uansett må representasjonen d:h:m (med "minste" tiden lik 0:0:0) benyttes i grensesnittet mot bruker. Skriver du egne metoder, må du passe på adekvat tidsaritmetikk, f.eks. skal $0 : 22 : 29$ være mindre enn $1 : 3 : 31$, mens summen av disse to skal være $2 : 2 : 0$. Vi tar ikke hensyn til "hva klokka er nå" og betrakter alle ordre med tiden fra 0:0:0 og oppover som relevante (men ikke dermed nødvendigvis mulige å betjene).

Inputfiler

Inputfiler som skal benyttes er av to typer: den første inneholder data om bykart og den andre sekvens av ordrer. Den første har følgende format

s_1 e_1 a_1 t_1
s_2 e_2 a_2 t_2
s_3 e_3 a_3 t_3
...
#
d_1 h_1 t_1
d_2 h_2 t_2
d_3 h_3 t_3
...
#
...*evt*...

- En linje i for hver vei med: start- og endeholdeplass (heltall s_i og e_i), avstand (heltall a_i), kjøretid (i minutter t_i).
- Listen med veier er avsluttet med #.
- Deretter følger oppstilling av drosjer – en linje i for hver drosje med: drosjens registreringsnummer (en streng d_i), holdeplass der drosjen blir ledig (heltall h_i) og tiden t_i når drosjen blir ledig (tre heltall atskilt med et tegn, f.eks. $0:1:30$).
- Listen med drosjer avsluttes igjen med en #, og etter dette kan det stå noen kommentarer som ignoreres av programmet.

Filer med ordrer har en linje for hver ordre, evt. kommando, og formatet for disse skal være som beskrevet tidligere f.eks.:

g by.tst
d
p A C 0 : 0 : 15
t A C 0 : 1 : 05
r C F 0 : 3 : 55
...

3.2 Testing

For å teste programmet kan du bruke filen `by.tst` med et bykart og oppstilling av drosjer, samt filene `ordrer1` og `ordrer2` med sekvenser av ordre. Pekere til disse filene og til et eksempel på korrekt output for ordrefilen `ordrer1` finner du fra kursets hjemmeside (legges ut 23.10).

Du skal levere utskrift fra to kjøringer. Den ene fra kjøringen av `ordrer1` som nevnt over og den andre fra følgende (tilsvarende kommandoer som står på filen `ordrer2`):

```
g by.tst
p 4 3 0:1:00
t 3 4 0:1:10
r 1 7 0:1:00
r 10 8 0:1:30
t 3 10 0:1:00
```

Under retting vil vi kjøre ditt program på andre bykart og ordrefiler for å teste korrektheten.

4 Krav til innlevering

Minstekravet til innlevering er at besvarelsen skal være forklart og at den skal virke. Den består altså av to deler:

Del I

Denne delen er en noe mer høytnivå dokumentasjon av programmet ditt enn det som gjøres i `javadoc`.

Du skal beskrive kort organisasjon av programmet ditt: hvilke moduler (klasser, interface) du bruker og hva er hensikten med hver av disse. For interface'ne forklarer du hvilken abstraksjon de uttrykker og hvilke implementasjoner som skal brukes i programmet. For

klasser, forklarer du kort hovedelementene av datastrukturen som er avgjørende for logikken (ikke nødvendigvis for implementasjonen) av programmet ditt.

Videre, for hver hovedalgoritme (valg av drosje fra 2.1 og tre alternative valg av rute fra 2.2) skal du

- forklare hovedprinsippet bak algoritmen — ikke minst, eventuell datastruktur som den bruker (du kan gjerne bruke pseudo-kode i denne forklaringen)
- angi tidskompleksitet (relativt til implementasjon av de brukte modulene)
- begrunne utfra det ditt valg av implementasjoner av de brukte modulene

Selv om det minste kravet er at alt skal virke, bør du helst forsøke å implementere mest mulig effektive algoritmer – spesielt med tanke på tidskompleksistet. Alle ordrene kan programmeres med en (adekvat modifikasjon av en) passende algoritme uten bruk av "brute force" som tester alle muligheter – slike ineffektive løsninger skal unngås.

Videre, bør du lage et program som er relativt lett å tilpasse eventuelle nye/endrede krav. Du skal spesielt forklare hva som måtte legges til, evt. endres, dersom programmet skulle nå gi mulighet til å:

- velge en drosje (seksjon 2.1) for en gitt ordre (A, B, t) utfra litt andre kriterier, f.eks., ikke bare ved å minimalisere kjøreavstand men også tiden drosjen må vente på kunden ved A (dvs. minimalisere ventetiden fra drosjens ankomst til A til t),
- betjenne nye ordretyper, f.eks., at kunden vil kjøre fra A til B gjennom lengst mulig rute (uten løkker), eller gjennom en bestemt holdeplass C.

Del II

Andre delen er selve programkode. Denne skal følge retningslinjer for god ADT-programmering, og det er et krav at minst to av ordretypene skal håndteres av en og samme generiske algoritme, instansiert etter behov. Alle moduler/klasser skal være dokumentert for `javadoc`. Du trenger ikke å gjenta i dokumentasjonen her det du har sagt i Del I, men du skal spesifisere alle `public` og evt. forklare evt. skjulte (`private`, `protected`) elementene:

- meningen med de forskjellige variablene i datastrukturen
- dokumentasjon av alle `public` metoder: meningen med metoden, forbetingelser for inparametere, bakbetingelser for resultatet/effekt, unntaksituasjoner, evt. kort prinsipp for algoritme.

Besvarelsen skal angi stien til katalogen `X` der programmet ditt ligger.