Obligatirisk oppgave 2

INF 3/4130, 2005

NB: Forandret 4. nov.: Leveringsfristen er mandag 21. november (men merk at hele obligen må være endelig godkjent senest 30. nov. for å få gå opp til eksamen). På Oppgave 1 kan man jobbe to og to sammen. Begge skal i så fall levere løsningen, men skal oppgi hvem man har jobbet sammen med. De som er med i konkurransen og lager et program som går ut over det som er krevet i Oppgave 1 under, kan selvfølgelig levere konkurranseprogrammet som svar på denne oppgaven. Oppgave 2 og 3 skal hver student løse alene. Leveringsfristen for å være med i konkurransen er 24. november.

(All tekst etter dette er nøyaktig som den ble lagt ut 1. nov kl 15.30)

Oppgave 1 (rett fram utgave av konkurranse-oppgaven)

(Denne oppgaveteksten er langt på vei den samme som for konkurransen, bare at en del ting er fjernet.)

Oppgaven er å skrive et program for løsing av "15-spillet", som også kommer som 8-spillet og generelt (NxN-1)-spillet (på et NxN-brett). Dette er diskutert i læreboka som "8-puzzle game" på side 717. Programmet skal lages generelt for NxN-brett, og data inn til programmet skal være først tallet N (på egen linje), og så N linjer med N tall i hver, som angir startposisjonen (der 0 angir det tomme feltet). Eksempel:

```
3
1 2 3
0 4 5
7 8 6
```

Programmet skal da finne frem til en måte å flytte brikkene slik at man kommer til mål-situasjonen:

```
1 2 3
4 5 6
7 8 0
```

(og tilsvarende for NxN-brett). Løsningen man finner skal ha færrest mulig trekk. Et lovlig trekk er altså å la den tomme posisjonen (0-en) ”bytte” posisjon med en av nabo-brikkene. Et slikt trekk angis med hvordan *den tomme posisjonen* flytter seg, med V, H, O, N (for Venstre, Høyre, Opp og Ned). En løsning på problemet over er derved: HHN, og denne skal skrives ut på skjermen om programmet finner en løsning. Programmet skal bruke A*-søking, og kan passelig bruke en rett fram Manhattan-heuristikk (se oppgave 23.7).

Etter det kursledelsen har orientert seg ser det ut til at man med denne metoden bør kunne løse alle 8-spill-problemer, og noe mer vil ikke bli forlangt. Det er jo imidlertid morsomt å se om det også kan løse enklere 15-spill-problemer (som kan løses i få trekk), og det er morsomt om dere legger ved i leveringa en kommentar om hva programmet klarer, og på hvilken tid.

Programmet skal få angitt et filnavn ved oppstart, og på den kan det ligge flere oppgaver etter hverandre, med en blank linje mellom. Fila avsluttes ved at N angis til null.

Oppgave 2

Her skal du besvare følgende to spørsmål:

A. Er L (angitt under) uavgjørbart? Bevis svaret.
L = {M | M er en Turing-maskin som svarer 'JA' hvis og bare hvis input-strengen er '010'}

B. Hva er kompleksiteten til problemet angitt under (som kalles: "Lengste vei mellom to noder")? Bevis svaret.
INPUT: Graf G og to noder s og t i V(G) og heltall K.
SPØRSMÅL: Finnes det en enkel sti av lengde K eller mer mellom s og t i G?

Oppgave 3

Denne oppgaven går ut på å implementere FordFulkerson-algoritmen, med bruk av kortest mulig forbedringsveier i hvert skritt (Edmonds og Karps variant). Alle kapasiteter skal være heltallige, og et problem skal angis slik: Først en linje med antall noder m, og så m linjer med m tall i hver som angir kapasitetene mellom hvert par av noder. Vi kan tenke oss at nodene er nummerert fra 1 til *m*, og da er node 1 kilde og node *m* sluk. Et eksempel kan være slik:

```
5
0 5 1 0 0
0 0 1 4 0
0 2 0 0 6
0 0 1 0 1
0 0 0 0 0
```

Merk at det altså kan være positiv kapasitet både fra en node u til en node v, og fra v til u. Langs diagonalen angis tallet null. Det vil aldri gå kanter inn i kilden eller ut av sluket (og dermed vil første kolonne og siste linje alltid ha bare nuller).

Svaret skal angis ved at det på egen linje skrives ut maksimal flyt fra kilde til sluk og hvor mange forbedrings-steg algoritmen brukte. Videre skal det på *m* linjer med *m* tall i hver skrives ut flyt på hver enkelt kant. På egen linje til slutt skal man skrive ut nummeret på de nodene som er på "kilde-siden" av et kutt med kapasitet lik flyten.

```
4 <antall forbedrings-skritt>
0 3 1 0 0
0 0 1 2 0
0 0 0 0 3
0 0 1 0 1
0 0 0 0 0
Kutt: 1 2 4
```

Hint til løsningen: Det kan være greit å holde kant- og kapasitet-informasjonen i *m*m*- arrayer. Da kan man greit unne seg hvertfall tre slike arrayer: Én for det opprinnelige problemet (N), én for den flyten man i øyeblikket har på hver kant (*f*), og én med de flytforandringene som er mulig (N*f*), som i figur 14.8. NB: En ting man må passe litt ekstra på her er de node-parene der det er kapasitet begge veier (og om man her vil bruke en fjerde *m*m*-array, så er det greit). Ellers er det jo bare å gjennomføre et bredde-først søk med en FIFO-kø, og det er kanskje lurt å ha en boolsk array som angir om noden er sett i dette søket.

Programmet skal få angitt to filnavn ved oppstart, og på den første kan det ligge flere oppgaver etter hverandre, med en blank linje mellom. Fila avsluttes ved at *m* angis til null. På den andre fila skal skrive ut svarene som angitt over, med en blank linje mellom.

Selve leveringen

Angående selve leveringen, så er følgende oppskrift gitt av gruppelærerne, for hvordan de vil ha det:

Legg Java-filene og andre filer i en mappe kalt Oblig1_<ditt brukernavn> og zip dem til en tar.gz slik (i Linux, mens du står i katalogen som mappen ligger i) på kommandolinja:

```
tar cvfz Oblig1_<ditt brukernavn>.tar.gz  Oblig1_<ditt brukernavn>
```

og send tar.gz fila til gruppelæreren (karianho@student.matnat.uio.no eller psmaas@ifi.uio.no).

Du behøver ikke legge med utskrift fra noen kjøring, men vi forutsetter at du hvertfall har fått programmene til å virke for de tastdataene som vil bli lagt ut, og helst mange flere du selv har laget. Om du er i tvil om noe, ta kontakt med gruppelæreren pr. mail. Med i leveringen skal også følge en liten rapport der ditt fulle navn og brukernavn står på toppen, og der du besvarer spørsmålene i oppgave 2 . Det er greiest om den er i rent tekst-format eller PDF-format.

Kursledelsen, ved Stein Krogdahl
Lykke til!