Bygge et strømnett

Et kraftselskap skal bygge ut et strømnett, når man bygger ut et strømnett er det mye å tenke på. Alle hus som ønsker strøm, må kobles til nettet og det må være en plan for hva som skjer når det blir strømbrudd. I denne semesteroppgaven skal dere hjelpe til med å finne gode løsninger for hvordan strømnettet skal utformes. I virkeligheten finnes det mange forskjellige typer kabler og hvilken kabel som skal brukes avhenger av hvor mye strøm den trenger å levere. Vi forenkler i denne oppgaven og sier at det er bare en type kabel og det spiller ingen rolle hvor mange hus som kobles på samme kabel. Implementer interfacet IProblem.

Oppgave 1

Selskapet har samlet inn informasjon om alle punktene de må koble til strømnettet, i tillegg har de samlet inn informasjon for mange par av punkter om hvor mye det koster legge strømledning fra et punkt til et annet. Dette blir modellert som en vektet graf, punkter som trenger strøm er noder og par av punkter som kan kobles sammen med kabel er kanter. De har bestemt å bygge ut strømnettet på billigst mulig måte slik at alle punkter får strøm, kan du finne ut hvilke kabler de må legge?

Dette problemet er i litteraturen kalt «Minimum Spanning Tree», se kapittel 4.3 i boken.

Oppgave 2

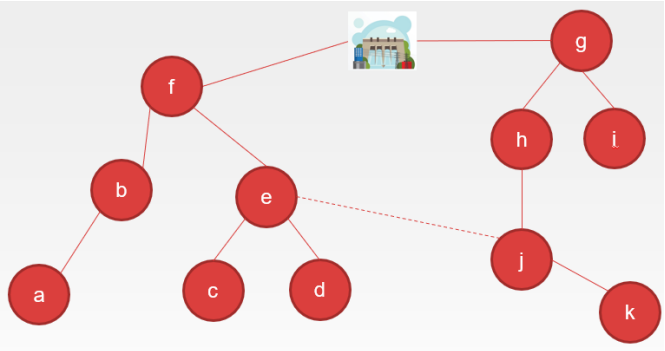
Etter at strømnettet ble bygget har kraftselskapet fått mange henvendelser angående strømbrudd. I moderne strømnett er det automatiske varslingssystemer som forteller hvor feilen har skjedd, men i dette tilfellet har de ikke installert slikt system. De ser at når et strømbrudd skjer så er det ofte mange punkter som mister strømmen samtidig. Et strømbrudd kan sees på som at en kant i grafen er fjernet, de som fortsatt er i samme sammenhengende komponent som kraftstasjonen har fortsatt strøm, mens alle andre punkter ikke har strøm. For å fortest mulig finne feilen vil de finne det punktet det er mest sannsynlig at feilen er og starte feilsøking derfra. Når de har mottatt 2 meldinger om strømbrudd antar de at dette gjelder samme feilen derfor må noden hvor feilen har skjedd være slik at begge de gitte punktene mister strømmen. Samtidig må de finne det punktet slik at minst mulig andre punkter mister strømmen.

Siden strømnettet ble bygget slik som det ble i oppgave 1 så er strømnettet et tre, dette gjør det lettere å finne ut hvor en feil ligger. Dette problemet kalles i litteraturen for «Lowest common ancestor» (kalt LCA). De optimale datastrukturene for LCA (som du vanligvis finner når du googler) er ikke pensum i INF102. Dere trenger ikke implementere en optimal algoritme, men skal finne en enkel algoritme. Det er mulig å finne en metode som har kjøretid O(n).

Oppgave 3

Etter mange klager på store strømbrudd har kraftselskapet skjønt at billigste løsning ikke alltid er den beste, nå skal de prøve å fikse problemet ved å bygge en ekstra kabel (legge til en kant i grafen). Målet er at store strømbrudd ikke skal skje. Ved å se på statistikk over strømbrudd har de funnet ut at det oftest er kablene som forårsaker strømbrudd, grunnen til klagene var at feil i en enkelt kabel førte til at veldig mange hus mistet strømmen. Du skal finne det par av noder der det lønner seg mest å legge til en kant, det vil si den kanten som minimerer det største mulige strømbruddet som følge av at 1 kabel feiler.

Her er det mulig med løsninger som bruker så mye som O(n^4) men for å få full score må dere få mye bedre kjøretid enn det.



I eksemplet over ser vi at hvis kanten mellom **kraftstasjonen** og node **f** feiler så vil 6 noder miste strømmen. Hvis vi kobler sammen nodene **e** og **j** så vil dette ikke lenger være et problem siden da vil grafen fortsatt være sammenhengende hvis kanten mellom **kraftstasjonen** og node **f** feiler. Og det største strømbruddet som kan skje etter at **e** og **j** er koblet sammen er 2, det skjer hvis kanten mellom **f** og **b** feiler.

Vurdering

Denne Obligen teller 15% av endelig karakter og dere vil få en poengsum fra 0 til 15.

- Kodekvalitet gir 2 poeng
  - Koden skal være ryddig og lesbar
  - Du skal unngå repetisjon av kode og ellers bruke konsepter dekket i INF101
- Kjøretidsanalyse og forklarende tekst gir 3 poeng
  - Hver funksjon du skriver skal analyseres med kjøretid, du får poeng hvis kjøretiden stemmer med implementasjonen og trekk hvis kjøretiden er feil.
  - Enten skrives dette i et eget dokument eller som kommentarer i koden
  - Det er først og fremst Worst case kjøretid i big-O notasjon vi forventer, men om dere ønsker å uttrykke f.eks. forventet kjøretid el. lignende er alt greit så lenge det er klart beskrevet i readme filen.
  - Kjøretiden skal uttrykkes best mulig med M – antall kanter i grafen og N – antall noder i grafen. På 2 av oppgavene vil antall kanter være O(N) da bruker dere bare N.
  - I readme filen skal dere også skive en forklarende tekst til hver av de tre oppgavene slik at det blir lettere for gruppeleder å forstå hva dere har tenkt (spesielt viktig om dere ikke helt fikk til å programmere det). Skriv også hvis det var noe som var vanskelig.
- Effektivitet og rett bruk av datastrukturer gir 5 poeng
  - Noen av oppgavene krever bruk av datastrukturer og algoritmer vi har lært i kurset, andre krever at dere finner på egne algoritmer.
  - LinkedList, ArrayList, HashMap eller PriorityQueue kan være aktuelle. Tenk over at du har valgt den som du mener passer best.
  - Vi kommer til å se på endelig kjøretid av de 3 metodene i Interfacet vurdere om vi mener du burde gjort det mer effektivt.
- Korrekthet av koden gir 5 poeng
  - Alle 3 oppgavene kommer med JUnit tester, hvis disse passerer så betyr det at dere har fått til alle oppgavene og får 3 poeng. Det er ikke meningen dere skal endre på eksisterende tester for å få de til å passere.
  - Om testene ikke passerer så kan dere allikevel få poeng på denne posten, men det må de som retter vurdere individuelt.

Innlevering via GIT

Koden finner dere på GIT, dere skal laste ned prosjektet, fylle inn manglende kode og pushe til GIT. Du må selv sjekke på GIT serveren at koden din er lastet opp.
INNLEVERINGSFRIST: Torsdag 29. Oktober kl. 16:00

Lykke til, hilsen Olav, Laura og Martin