Obligatorisk afleveringsopgave: Sociale Netværk

Philip Bille Anders Roy Christiansen

Om opgaven

Den obligatoriske afleveringsopgave indgår i kursets samlede bedømmelse. Opgaven skal laves enkeltvis eller i grupper af 2 eller 3 personer. Opgaven skal afleveres via Campusnet og CodeJudge.

Sociale Netværk

Institut for Analyse og Evaluering af Sociale Medier^{TM} har du brug for effektivt at kunne evaluere statisk information for sociale netværk. Opgaven består i at designe og implementere algoritmer der på skalerbar vis kan evaluere sociale netværk.

Vi definerer sociale netværk på følgende måde. Et *netværk* består af P personer og V *venskabrelationer*. Hver venskabsrelation består af to personer p_1 og p_2 og vi siger at p_1 og p_2 er *venner*. Hvis p_1 er ven med p_2 er p_2 ven med p_1. Hvert netværk er formateret som en tekstfil på følgende måde. Formatet er illustreret i fig. 1. Den første linie består af en liste af navne på de P personer $p_0, p_1, \ldots, p_{P-1}$ separerede med blanktegn. Længden af navnet (antallet af tegn) på person p_i er n_i og $N = \sum_{i=0}^{P-1} n_i$. Hver af de efterfølgende V linier består af to heltal i og j, separeret med et blanktegn, som indikerer at p_i og p_j er venner. Endelig er der en ekstra linie, der angiver en beregning vi ønsker foretaget på netværket (se opgaverne).

```
Inky Pinky Blinky Clyde Luigi Mario Bowser
0 2
1 2
5 6
3 5
2 4
4 5
2 3
1 4
```

Figur 1: Inputformat af sociale netværk.

I opgaverne nedenfor antager vi at de sociale netværk vi arbejder med er *tynde*, dvs. antallet af venskabrelationer er betydeligt mindre end antallet af mulige venskabsrelationer (præcis ligesom alle andre velkendte sociale netværk).

Opgaver

1 Størrelse af netværk Vi vil gerne beregne antallet af personer og venskabsrelationer i et givet netværk. F. eks. er der i netværket angivet i fig. 1 7 personer og 8 venskabsrelationer. Løs følgende opgaver.

1.1 Giv en algoritme, der givet et netværk, beregner antallet af personer og venskabrelationer. Analyser køretiden af din algoritme som funktion af P, V og N.

1.2 [†] Implementer din algoritme. Inputformat er som i fig. 1, hvor der er tilføjet en ekstra linie der starter med ordet `stoerrelse` og output skal være en linie med de to tal P og V separeret med et blanktegn. F. eks. hvis den ekstra linie er `stoerrelse` i input fra fig. 1 skal output være `7 8` .

2 Tætte venskaber Vi er nu interessede i at undersøge om små grupper af personer er meget tætte venner. Specifikt definerer vi en delgruppe X af k personer til at være et *tæt venskab* hvis alle personer i X er venner med alle andre personer i X. F. eks. er `Pinky`, `Blinky` og `Luigi` i et tæt venskab, mens `Pinky`, `Clyde`, `Luigi` og `Mario` ikke er det. Løs følgende opgaver.

2.1 Giv en algoritme, der givet et socialt netværk og en mængde X af k personer, afgør om X er et tæt venskab eller ej. Analyser køretiden af din algoritme som funktion af P, V, N og k.

2.2 [†] Implementer din algoritme. Inputformat er som i fig. 1, hvor der er tilføjet en ekstra linie der starter med ordet `taetvenskab` efterfulgt af en sekvens af tal, der indikerer hvilke personer vi ønsker at undersøge om har et tæt venskab. Output skal være en linie med enten `ja` eller `nej`. F. eks. hvis den ekstra linie er `taetvenskab 1 2 4` i input fra fig. 1 skal output være `ja`.

3 Venskabskæder En *venskabskæde* er en sekvens af personer $p_0, \ldots, p_j$ så p_i er ven med p_{i+1} for alle i, $0 \leq i < j$. *Længden* af en venskabskæde er antallet venskabsrelationer i sekvensen. Givet en person p og et heltal $t \geq 0$ er p's *t-venner* alle personer, der har en venskabskæde til p af længde højst t (vi definerer 0-venner af p til at være p selv). F. eks. er `Bowsers` 2-venner `Bowser`, `Mario`, `Luigi` og `Clyde`. Løs følgende opgaver.

3.1 Giv en algoritme, der givet et socialt netværk, en person p og et heltal t, $t \geq 0$, beregner alle t-venner af p. Analyser køretiden af din algoritme som funktion af P, V, N og t.

3.2 [†] Implementer din algoritme. Inputformat er som i fig. 1, hvor der er tilføjet en ekstra linie der starter med ordet `tvenner` efterfulgt af to tal i og t som indikerer vi skal beregne t-venner af p_i. Output er en liste af t-vennerne af p. F. eks. hvis den ekstra linie er `tvenner 6 2` i input fra fig. 1 skal output være `Bowser Mario Luigi Clyde`.

4 Svageste venskabskæde Ikke alle venskabsrelationer er lige stærke og i det nyeste datasæt er styrken på hvert venskab estimeret. Specifikt knytter vi til hver venskabsrelation v en *styrke*, som er et heltal større eller lig 0. Styrken indikeres i input som et trejde heltal efter hver venskabsrelation, som vist i fig. 2. Styrken af en venskabskæde er summen af styrkerne på hver venskabsrelation i venskabskæden. Givet to personer p_1 og p_2 er vi interesserede i at finde en *svageste venskabskæde* mellem p_1 og p_2, dvs., en venskabskæde af minimal styrke mellem p_1 og p_2. F. eks. er en svageste venskabskæde mellem `Bowser` og `Pinky` `6 5 4 1`. Løs følgende opgaver.

```
Inky Pinky Blinky Clyde Luigi Mario Bowser
0 2 5
1 2 2
5 6 88
3 5 77
2 4 40
4 5 10
2 3 8
1 4 5
```

Figur 2: Det sociale netværk fra fig. 1 med styrke på hver venskabsrelation.

4.1 Giv en algoritme, der givet et socialt netværk og to personer p_1 og p_2 beregner en svageste venskabskæde mellem p_1 og p_2. Analyser køretiden af din algoritme som funktion af P, V og N.

4.2 [†] Implementer din algoritme. Inputformat er som i fig. 2, hvor der er tilføjet en ekstra linie der starter med ordet `svagestevenskab` efterfulgt af to tal i og j, som indikerer vi skal beregne en svageste venskabskæde fra i til j. Output er venskabskæden ordnet fra i til j (hvis der findes forskellige svageste venskabskæder mellem i og j udskrives blot én af disse). F. eks. hvis den ekstra linie er `svagestevenskab 6 1` i input fra fig. 2 skal output være `Bowser Mario Luigi Pinky`. Du kan antage at der altid findes mindst en venskabskæde mellem i og j. Til denne opgave kan du eventuelt benytte den udleverede implementation af en hob (se CodeJudge).

Formalia

Den obligatoriske afleveringsopgave indgår i kursets samlede bedømmelse. Opgaven skal afleveres via Campusnet og CodeJudge. Opgaven skal laves enkeltvis eller i grupper af 2 eller 3 personer. I skal aflevere følgende.

- En `pdf`-fil med jeres løsninger på alle de teoretiske opgaver (dem der ikke er markeret med [†]). Filen skal afleveres via Campusnet.
- Jeres kode til de eksperimentielle opgaver (dem markeret med [†]). Koden skal afleveres via CodeJudge.

Til de teoretiske opgaver vær opmærksom på (som altid) at "giv en algoritme" betyder, at man giver en effektiv løsning, beskriver sin løsning kort, præcist og entydigt og argumenterer for korrekthed. I vurderingen lægges der vægt på følgende.

- Anvendelse af relevante algoritmiske koncepter og teknikker fra kurset (modellering af problemer, valg af datastrukturer, analyse af algoritmer, etc.)
- Kvalitet af beskrivelse af algoritmer.
- Kvalitet af programmer (korrekthed, robusthed, effektivitet, etc.)

Samarbejdspolitik

Opgaven skal laves enkeltvis eller i grupper af 2 eller 3 personer. Derudover er det ikke tilladt at samarbejde om opgaven, udover at diskutere opgaveteksten med undervisere og medstuderende, der tager kurset i samme semester. Man må under ingen omstændigheder udveksle, udlevere eller på anden måde kommunikere løsninger på (dele af) opgaven. Man må heller ikke bruge løsninger fra tidligere år, løsninger fra tilsvarende kurser, eller løsninger fundet på internettet eller andetsteds. Overtrædelse af ovenstående politik er eksamenssnyd og vil blive indberettet.

Afleveringsfrister

Opgaven stilles torsdag d. 26.03.2015 og skal afleveres senest søndag d. 26.04.2015, kl. 23:59 på Campusnet og CodeJudge.