TIES3510 Joukko-oppi ja logiikka ohjelmoinnissa harjoitustehtäviä

Antti Valmari
Jyväskylän yliopisto, Informaatioteknologian tiedekunta

19. lokakuuta 2021

Kunakin viikkona kirjoita palautuksista yksi PDF-tiedosto. Meilaa se antti.valmari@jyu.fi viimeistään sunnuntai-iltana klo 23:59.

Osa tehtävistä on niukasti määritelty. Tarkoituksena on, että etsit tai järkeilet tarvittavat lisätiedot itse. Tosielämän tilanteissa on usein erikoistapauksia ja poikkeuksia ja poikkeusten poikkeuksia, jotka paisuttavat tehtävän kohtuuttoman isoksi, jos niitä ei mitenkään rajata. Aseta johonkin järkevään kohtaan raja siihen, mitä otat huomioon ja mitä et. Jos olet epävarma rajauksista, niin kysy opettajalta (paitsi jos olet sitä mieltä, että ohjelmistovalmistajan ei koskaan pidä kysyä loppukäyttäjiltä mitään).

Viikko 1 (5.9.2021)

1. Lue `https://www.aamulehti.fi/a/24277883` ja kerro mitä ajatuksia se herätti, varsinkin tietojärjestelmien suunnittelemisen kannalta. Jos tunnet muita samankaltaisia tapauksia, olipa niissä tietokone mukana tai ei, voit kommentoida myös niitä. Kirjoita vähintään 50 sanaa mutta korkeintaan puoli A4-sivua.

2. Lakiasiantoimisto on tilannut sinulta ohjelman perinnön arvon jakamiseen kullekin perilliselle. Ohjelman tulee tulostaa jotain tyyliin ”Jaana Jälkeläinen saa 25000€, Pekka Perijä saa 50000€, ja Emilia Edunsaaja saa 25000€”. Ohjelman ei tarvitse osata käsitellä testamentteja, ennakkoperintöjä yms., vaan riittää, että se jakaa perinnön arvon euroina ilmaistuna sukulaisille sillä tavalla kuin Suomen lainsäädäntö määrää normaalitapauksessa. Ohjelman täytyy osata käsitellä tilanteet, joissa myös osa perillisistä on kuollut.

Esitä jollakin selkeällä tavalla (mielellään pseudokoodina tai ohjelmakoodina) se ohjelman osa, joka laskee, kuinka paljon kukin saa.

3. Kertaa Backus-Naur-formaattia eli BNF:ää sen verran kuin tarvitset, jotta saat tehtävän 4 (ja myöhemmin lisää saman kaltaisia) suoritettua. Voit käyttää apuna seuraavia veppisivuja. Tästä tehtävästä sinun ei tarvitse palauttaa mitään.

- `http://users.jyu.fi/~ava/t_BNF.html`
- `http://users.jyu.fi/~ava/t_BNF2.html`

4. Suunnittele BNF-esitystapa tiedoille, jotka tehtävän 2 ohjelma tarvitsee syötteekseen. Voit olettaa, että leksikaalitason symbolit on valmiiksi määritelty. Ota huomioon, että ohjelma tarvitsee riittävät tiedot sukulaissuhteista.

Suunnittele toinen esitystapa, jolla lakiasiaintoimiston henkilökunta syöttää tiedot sisään. Ei tarvitse kertoa mihin kohtaan ruutua ja millä värillä mikäkin tekstikenttä tulee; riittää kertoa, mitä tekstikenttiä on olemassa ja mikä tieto kuhunkin tulee.

Sisällytä palautukseesi myös molemmilla tavoilla esitetty esimerkkisyöte, joka mielestäsi riittää havainnollistamaan olennaisimmat asiat.

Viikko 2 (12.9.2021)

5. Lue `https://ai.googleblog.com/2006/06/extra-extra-read-all-about-it-nearly.html` ja `https://www.uni-weimar.de/fileadmin/user/fak/medien/professuren/Mediensicherheit/Teaching/SS15/SSS/p190-pattis.pdf` ja kerro, mitä ajatuksia ne sinussa herättivät. Kirjoita vähintään 100 sanaa mutta korkeintaan yksi A4-sivu.

6. Kevään 2020 Tietotekniikan pääsykokeessa järjestettiin videokuulustelu. Tarjolla oli joukko ajankohtia. Niillä oli nimet, kuten ke09. Jokainen hakija ilmoitti hänelle sopivat ajankohdat. Myös hakijoilla oli nimet, kuten Ylermi_Ylioppilas. (Nimet eivät välttämättä olleet oikeita nimiä, vaan esimerkiksi sähköpostiosoitteita.) Kuulustelijoita oli monta, mutta kaikki eivät olleet käytettävissä kaikkina ajankohtina. Siksi niiden hakijoiden määrä, jotka enintään voitiin jonain ajankohtana kuulustella, vaihteli ajankohdittain. Tämä tieto ilmoitettiin lukuna kullekin ajankohdalle.

Suunnittele syöttömuoto ohjelmalle, joka valitsee hakijoille kuulusteluajat. Esitä syöttömuoto BNF-määritelmänä, jonka alkusymboli on *S*. Ajankohdan nimen määritelmää ei tarvitse purkaa auki; riittää, että käytät välisymbolia *A*. Hakijan nimen määritelmää ei tarvitse purkaa auki; riittää, että käytät välisymbolia *H*. Luvun määritelmää ei tarvitse purkaa auki; riittää, että käytät välisymbolia *L*. Sisällytä määritelmään riittävästi välimerkkejä, niin että syötettä lukeva ohjelma pystyy tunnistamaan missä mikäkin tieto loppuu ja seuraava alkaa. Ohjelman täytyy muun muassa kyetä erottamaan, missä yhdelle hakijalle sopivat ajankohdat loppuvat ja seuraavalle sopivat alkavat.

Sisällytä palautukseesi BNF-määritelmän lisäksi esimerkkisyöte, jossa on ajankohtina ke09 ja ainakin kaksi muuta, ja hakijoina Ylermi_Ylioppilas ja ainakin kaksi muuta. Yhdelle hakijalle tulee sopia täsmälleen yksi ajankohta ja toiselle täsmälleen kolme. Muilta osin saat valita esimerkin yksityiskohdat kuten haluat.

7. Tämän tehtävän tavoitteena on saada sinut kirjoittamaan näkyville asioita, jotka ovat niin itsestään selviä, että niitä voi olla vaikea huomata. Eräs sellainen on, että aikatauluohjelman tulee antaa niin monelle hakijalle kuulusteluaika kuin mahdollista. Mitä muita ehtoja ohjelman antamien kuulusteluaikojen täytyy toteuttaa? Huomaa, että kysymys koskee tulostuksen asiasisältöä eikä tulostuksen syntaksia.

8. Oleta, että ajankohtien nimiä voivat olla pienet kirjaimet m, t ja k; hakijoiden nimiä a, b ja c; ja lukuja 0, 1 ja 2. Kirjoita näin täydennetty tehtävän 6 BNF-määritelmäsi veppisivun `http://users.jyu.fi/~ava/t_BNF2.html` oikean alareunan isoon laatikkoon ja testaa sitä oikean alareunan pieneen laatikkoon kirjoittamillasi esimerkeillä. Tarvittaessa korjaa BNF-määritelmääsi. Palauta lopullinen BNF-määritelmäsi ja tehtävän 6 mukainen esimerkkisyöte, joka läpäisee testin.

Viikko 3 (19.9.2021)

9. Jos ei voida antaa aikaa jokaiselle, niin mikä olisi mielestäsi hyvä tapa valita, kuka saa ajan ja kuka ei? Perustele vastauksesi. Kerro lisäksi, ketkä saavat ajan seuraavassa esimerkissä. Tarjolla on kaikkiaan kolme aikaa, to08, to10 ja to12. Xantippalle kelpaa vain to10, Yvonnelle vain to12 ja Zachariakselle kelpaavat to10 ja to12 mutta ei to08.

10. Kukin kuulustelija on kertonut hänelle parhaiten sopivat kuulusteluajat kuulustelujen järjestäjälle (siis ei aikatauluohjelmalle). Järjestäjä yrittää aikatauluohjelman avulla saada jokaiselle hakijalle hänelle sopivan kuulusteluajan. Tarvittaessa järjestäjä neuvottelee kuulustelijoiden kanssa, voisiko joku heistä kuulustella jonain sellaisena ajankohtana, jota hän ei ollut ilmoittanut parhaiten sopiviinsa. Jos esimerkiksi aikataulu on muuten valmis mutta yksi hakija on jäänyt ilman kuulusteluaikaa, ja hänelle sopii ti12, ti13 tai ke09, niin järjestäjä voi kysellä suostuisiko joku kuulustelemaan jonain noista ajankohdista.

Mitä aikatauluohjelman pitää tulostaa, jotta se olisi järjestäjälle mahdollisimman kätevä? Esitä tulostuksen syntaksi BNF:llä ja kerro tulostuksen asiasisältö.

11. Oletetaan, että ohjelma tulostaa ”On mahdotonta antaa jokaiselle hakijalle hänelle sopiva kuulusteluaika, koska Nipa_Nirso ei ilmoittanut yhtään ajankohtaa hänelle sopivaksi.” Syötteestä voidaan helposti tarkastaa, onko totta, että Nipa_Nirso ei ilmoittanut yhtään ajankohtaa hänelle sopivaksi. Jos se on totta, niin on ilmeistä, että on mahdotonta antaa jokaiselle hakijalle hänelle sopiva kuulusteluaika, koska Nipa_Nirsolle sopivaa ajankohtaa ei ole olemassakaan.

Tilanne on saman kaltainen, jos ohjelma tulostaa ”On mahdotonta antaa jokaiselle hakijalle hänelle sopiva kuulusteluaika, koska sekä Matti_Myöhäinen että Vilma_Viimehetki ilmoitti sopivaksi vain pe15, ja pe15 voidaan järjestää vain yksi kuulustelu.” Nytkin on helppo tarkastaa syötteestä, että sanan ”koska” jälkeen tulostetut tiedot pitävät paikkansa. Nytkin on selvää, että jos ne pitävät paikkansa, niin on mahdotonta antaa jokaiselle hakijalle hänelle sopiva kuulusteluaika.

On olemassa yleinen tämänkaltainen ehto, joka voidaan tulostaa perusteluksi sille, että on mahdotonta antaa jokaiselle hakijalle hänelle sopiva kuulusteluaika; joka on helppo tarkastaa syötteestä; ja joka toimii aina kun on mahdotonta antaa jokaiselle hakijalle hänelle sopiva kuulusteluaika. Se on vaikea keksiä eikä ihan helppo ilmaista, ja tarkoitus onkin palata tähän seuraavalla viikolla. Yritä kuitenkin keksiä edes jotain sinnepäin jo tällä viikolla! Raportoi mitä keksit ja arveletko sen olevan yleispätevä. Jos et keksi mitään, niin kerro jotain siitä miten yritit ratkaista tämän tehtävän.

12. Useista samana ajankohtana tapahtuvista kuulusteluista puhuminen on hankalaa. Myös niiden miettiminen aikataulun muodostamisalgoritmia suunniteltaessa on hankalaa. Onneksi on olemassa yksinkertainen keino muuntaa tehtävä sellaiseksi, että on joukko kuulustelulokeroita, joista kussakin voidaan kuulustella eninään yksi hakija. Kerro tällainen keino. Jos et keksi sitä, niin kerro miten yritit sen löytää, esimerkiksi kerro mitä ideoita kokeilit.

Viikko 4 (26.9.2021)

13. Aikatauluohjelma löytyy täältä: `http://users.jyu.fi/~ava/aikataulu.html`. Suunnittele sen avulla tiedostossa `http://users.jyu.fi/~ava/aikaesim.html` olevalle ai-

neistolle aikataulu, jossa kuulustelijoita on viisi; kuulusteluun tarvitaan yksi kuulustelija kahdeksi tunniksi; kukaan kuulustelija ei pidä aamu klo 8 ajoista mutta jokainen suostuu ottamaan niitä tarvittaessa; jokaiselle kuulustelijalle pitää järjestää lounastauko eli vapaata puolenpäivän tienoilla; jos on aivan pakko, niin hakijalle voidaan antaa hänelle sopimaton aika; ja mitä aikaisemmin kaikki kuulustelut on saatu valmiiksi, sen parempi.

Kerro palautuksessasi suunnittelemasi aikataulu. Jos jouduit valitsemaan ristiriitaisten tavoitteiden väliltä, kerro miten ja miksi valitsit. Kerro palautuksessasi myös mitä etuja ja mitä haittoja aikatauluohjelman tulostuksella on verrattuna tehtävässä 10 suunnittelemaasi tulostukseen.

14. Lue `http://users.jyu.fi/~ava/TIEP1020.pdf` ruudut 119–133. Niissä on ilmoitettu tehtävän 11 vastaus sanallisesti olettaen, että kuulustelijoita on vain yksi. Ilmoita tämä vastaus joukko-opin kaavana. Hakijoiden joukko on H ja ajankohtien joukko A. Voit käyttää merkintää $|X|$ tarkoittamaan joukon X alkioiden määrää, ja $s(a,h)$ tarkoittamaan, että ajankohta a sopii hakijalle h. Voit käyttää myös logiikan merkintöjä. Vihje 1: niiden hakijoiden määrä, joille ajankohta a sopii, voidaan ilmaista $|\{h \in H \mid s(a,h)\}|$. Vihje 2: väite, että mikään ajankohta ei sovi hakijalle h, voidaan ilmaista $\neg\exists a \in A : s(a,h)$. Tämä on enemmän ongelmanratkaisutehtävä kuin joukko-opin osaamistehtävä, mutta jos haluat, niin voit kerrata joukko-oppia `http://users.jyu.fi/~ava/TIES3510.pdf` luku 1.1 ja logiikkaa tämän tehtävän alussa linkatusta tiedostosta.

15. Tässä tehtävässä aikataululla tarkoitetaan sellaista, jossa kukin hakija saa täsmälleen yhden ajan, se aika sopii hänelle, ja kukin ajankohta annetaan enintään yhdelle hakijalle. Vastaa seuraaviin kysymyksiin ja perustele vastauksesi lyhyesti. Älä vetoa tietoon ”todistetusti **P**:ssä” (vaikka se löytyykin edellisen tehtävän linkin ruudusta 123), vaan perustele vastauksesi muulla tavalla.

(a) Onko tehtävä ”laadi aikataulu” **NP**:ssä?

(b) Onko tehtävä ”onko aikataulua olemassa” **NP**:ssä?

(c) Onko tehtävä ”onko niin, että aikataulua ei ole olemassa” **NP**:ssä?

(d) Onko tehtävä ”laadi aikataulu” **NP**-täydellinen?

(e) Onko tehtävä ”onko aikataulua olemassa” **NP**-täydellinen?

(f) Onko tehtävä ”onko niin, että aikataulua ei ole olemassa” **NP**-täydellinen?

16. Luonnostele algoritmi aikataulun suunnittelemiseksi. Tehtävän helpottamiseksi oleta, että kuulustelijoita on vain yksi. Tässä vaiheessa sinulla ei ehkä ole tarpeellisia taustatietoja hyvän algoritmin suunnittelemiseksi, mutta suunnittele jotakin, jotta saisit tuntumaa aihepiiriin — tähän palataan tulevilla viikoilla paremmilla eväillä. Luonnostele niin hyvä algoritmi kuin tässä vaiheessa kohtuullisella vaivalla pystyt. Älä ratkaise tätä tehtävää kuuklaamalla ”aikataulun suunnittelualgoritmi”, vaan ajattele itse. Saat kuuklata tarvitsemiasi apukäsitteitä. Esim. jos ajattelit käyttää binäärihakupuita, niin saat kuuklata ”binäärihakupuu”.

Selosta palautuksessa algoritmisi niin selkeästi, että opiskelutoverisi pystyvät ymmärtämään sen. Tämä tehtävä on myös algoritmisten ajatusten selkeän ilmaisemisen harjoitus! Arvioi palautuksessa algoritmisi ajan ja muistin kulutusta. Lähetä tämän tehtävän

vastaus erillisenä PDF-tiedostona, jossa ei lue nimeäsi eikä muita tietoja jotka voisivat paljastaa kuka sinä olet, ja jonka alussa lukee ”tämän tiedoston saa näyttää kurssin muille opiskelijoille.” Opettajan aikomus on lähettää ensi viikolla palautuksesi yhden tai useamman muun opiskelijan luettavaksi, ja vastaavasti itse saat luettavaksesi yhden tai useamman muun palautuksen.

Viikko 5 (3.10.2021)

17. Saat opettajalta meilitse kahden kurssitoverisi kirjoittamat algoritmikuvaukset. Kirjoita niihin vastaukset, joissa kerrot, olivatko ne ymmärrettävissä ja olivatko ne helpot lukea. Kerro, mitkä kohdat olivat vaikeita ymmärtää. Jos algoritmikuvauksesta saa selvän, kommentoi sen ideoita: mikä on mielestäsi hyvää ja mikä ei. Jos suinkin mahdollista, niin kerro käsityksesi siitä, mitä algoritmi tekee ja kuinka paljon aikaa ja muistia se vie, jos aikoja on yhtä paljon kuin hakijoita, kumpiakin on n, ja

(a) jokaiselle hakijalle kelpaavat kaikki ajat paitsi syötteessä viimeisenä oleva.

(b) kuten (a), mutta syötteessä ensimmäiselle hakijalle kelpaa myös syötteen viimeinen aika, ja myös syötteessä viimeiselle hakijalle kelpaa myös syötteen viimeinen aika.

Kuitenkin, jotta sinun ei tarvitsisi kirjoittaa ylettömän pitkää vastausta, saat lopettaa kunkin vastauksen yhden A4:n tultua täyteen ja jättää loput asiat kirjoittamatta. Silti, jos mahdollista, kirjoita vastaukset kohtiin (a) ja (b) — jätä mieluummin jotain muuta pois.

Lähetä tämän tehtävän vastaukset erillisinä PDF-tiedostoina, joissa ei ole nimeäsi eikä mitään muitakaan sinua yksilöiviä tietoja. Kirjoita PDF-tiedoston alkuun ”tämän tiedoston saa näyttää algoritmin laatineelle opiskelijalle”. Sisällytä kummankin vastausmeilin otsikkoon (ei PDF-tiedostoon!) tunnuskoodi, jonka sait algoritmin mukana.

Siihen tiedostoon, jossa palautat tämän viikon muiden tehtävien vastaukset, laita tämän tehtävän kohdalle käsityksesi siitä, mitä oma algoritmisi (siis se, joka on tällä hetkellä kahden muun opiskelijan kommentoitavana) tekee kohtien (a) ja (b) tilanteissa.

18. Lue `http://users.jyu.fi/~ava/TIES3510.pdf` luku 2. Komisario Korppi haluaa lopettaa kulmarallin. Hän aikoo laittaa poliiseja risteyksiin siten, että jokainen keskusta-alueen kaduista muodostuva silmukka sisältää ainakin yhden risteyksen, jossa on poliisi. Korppi etsii poliiseille paikat syvyyteen ensin -haulla. Risteykset tunnetaan numeroilla. Haku aloittaa risteyksestä 1 ja tutkii risteyksestä lähtevät kadunpätkät kadunpätkän toisen pään mukaisessa kasvavassa numerojärjestyksessä.

(a) Saat itse valita säännön, joka ratkaisee, mihin risteykseen tai risteyksiin asetetaan poliisi, kun algoritmi on löytänyt silmukan. Kerro sääntö, jota käytät.

(b) Risteyksiä on viisi ja kadunpätkät ovat $(1,2)$, $(2,3)$, $(2,4)$, $(3,4)$, $(3,5)$, $(4,1)$, $(5,1)$. Selosta risteys kerrallaan, mitä algoritmi tekee.

(c) Risteyksiä on yhdeksän ja kadunpätkät ovat $(1,2)$, $(1,4)$, $(2,3)$, $(2,5)$, $(3,1)$, $(3,6)$, $(4,5)$, $(4,7)$, $(5,6)$, $(5,8)$, $(6,4)$, $(6,9)$, $(7,1)$, $(7,8)$, $(8,2)$, $(8,9)$, $(9,3)$, $(9,7)$. Luettele risteykset, joihin algoritmi asettaa poliisit.

(d) Algoritmi on juuri raportoinut löytäneensä silmukan 37, 25, 42, 51, 18, 37, ja ehdottaa siksi poliisin lisäämistä erääseen risteykseen. Risteykseen 42 on jo aiemmin päätetty sijoittaa poliisi. Onko varmaa, että algoritmin nyt ehdottama poliisi voidaan jättää lisäämättä? Perustele ”kyllä”-vastaus tai anna vastaesimerkki, joka perustelee ”ei”-vastauksen.

(e) Vilkaise `https://en.wikipedia.org/wiki/Feedback_vertex_set`. Kuinka helppoa arvioit olevan löytää tehokas algoritmi, joka löytää poliiseille sellaiset risteykset, että tarvitaan mahdollisimman pieni määrä poliiseja?

19. Tässäkin tehtävässä hakijalle saa antaa vain hänelle sopivan ajan, kenellekään ei saa antaa enempää kuin yhden ajan, ja samaa aikaa ei saa antaa useammalle kuin yhdelle hakijalle. Oletetaan, että osalle hakijoista on annettu aika. Merkitään heidän joukkoaan H':lla. Eräälle hakijalle h ei ole annettu aikaa. Joukon H' alkioiden saamia aikoja saa vaihtaa, mutta ketään H':ssa olevaa ei saa jättää ilman aikaa. Jos esimerkiksi A:lle sopii ajat 1 ja 2, B:lle sopii vain aika 1, A:lle on annettu aika 1, $H' = \{A\}$ ja $h = B$, niin B:lle saadaan aika vaihtamalla A:n ajaksi 2 ja antamalle B:lle aika 1.

Esitä graafikäsitteiden avulla seuraava tehtävä: voidaanko h:lle antaa aika? Havainnollista vastaustasi esimerkillä, jossa h:lle voidaan antaa aika, mutta se edellyttää ainakin kahden jo annetun ajan vaihtamista.

20. Tee tehtävä 16 uudelleen. Hyödynnä sitä uutta tietoa, jonka olet kurssilla saanut edellisen kerran jälkeen. Tätä vastausta ei lähetetä muille opiskelijoille, joten voit kirjoittaa sen samaan tiedostoon kuin muutkin tämän viikon tehtävät.

Viikko 6 (10.10.2021)

21. Hakijat käsitellään yksi kerrallaan. Hakijan käsitteleminen on helpointa selittää käyttäen suunnattua graafia, jonka solmuja ovat hakijat ja ajat. Hakijasta on kaari jokaiseen aikaan, joka sopii hänelle. Vapaasta ajasta ei ole kaarta mihinkään, ja varatusta ajasta on kaari hakijaan, jolle aika on varattu. Hakija käsitellään hänestä alkavalla leveyteen ensin -haulla. Jos haussa löytyy vapaa aika, niin on löydetty polku hakijasta vapaaseen aikaan. Tällöin haku keskeytetään ja jokainen polussa oleva hakija saa ajakseen polussa seuraavan ajan. (Tätä ennen heistä muilla kuin uudella hakijalla oli polussa edeltävä aika.) Jos haussa ei löydy vapaata aikaa, käsiteltävänä oleva hakija jätetään ilman aikaa.

(a) Perustele, että jos edellä kuvattu haku ei löydä vapaata aikaa, niin sellaista aikataulua ei ole olemassa, jossa sekä käsiteltävänä oleva hakija että jokainen ajan jo saanut hakija saa ajan.

(b) Merkitse hakijoiden määrää $|H|$ ja aikojen määrää $|A|$ ja arvioi, kuinka kauan enintään kestää aikataulun laatiminen, jossa mahdollisimman moni hakija saa hänelle sopivan ajan. Ota huomioon myös aikojen uusiminen polkua pitkin. Osana vastausta kerro, kuinka monta solmua ja kuinka monta kaarta graafissa on enintään.

(c) Oletetaan, että algoritmi on juuri jättänyt h:n ilman aikaa. Kerro, miten voidaan tulostaa mahdollisimman pieni tehtävän 14 mukainen joukko hakijoita, joka sisältää h:n.

(d) Oletetaan, että hakija käsitellään sitä aikaisemmin, mitä useampia aikoja hän ilmoitti itselleen sopiviksi. Takaako algoritmi, että jos hakija saa ajan, niin myös jokainen häntä enemmän sopivia aikoja ilmoittanut saa ajan? Jollei, niin missä mielessä algoritmi suosii hakijaa sitä enemmän, mitä enemmän aikoja hän ilmoitti itselleen sopiviksi?

(e) Edellä oletettiin, että kuulustelijoita on vain yksi, eli että kunakin ajankohtana on enintään yksi kuulustelu. Tehtävässä 12 keksittiin kikka, jolla voitiin mahdollistaa samalle ajankohdalle monta kuulustelua (jos kuulustelijoita on monta). Jos samana ajankohtana voi olla monta kuulustelua, niin millä perusteella ohjelman tulee päättää, suositaanko hakijaa A vai hakijaa B?

22. Suunnittele edellisen tehtävän algoritmin toteuttamiseen sopivat tietorakenteet olettaen, että sekä ajat että hakijat ovat luonnollisia lukuja. Saat itse valita, mitä luonnollisia lukuja (esim. onko 0 käytössä). Ota huomioon, että tietorakenteiden on mahdollisesti kyettävä esittämään sekä hakijalle annettu aika, että se, että hakijalle ei ole annettu aikaa (ei ole vielä annettu tai ei pystytä antamaan).

Suunnitelmasi ei tarvitse noudattaa pilkuntarkasti edellä kuvattua algoritmia ja tehtävän esitystapaa suunnattuna graafina; voi olla, että ohjelmointi helpottuu ja ehkä tehokkuuskin paranee kun valitset yksityiskohtia toisin. Esimerkiksi sitä kannattaa ehkä hyödyntää, että ajalla voi olla enintään yksi varaaja. Kerro vastauksessasi, miten edellä kuvattu graafi ja leveyteen ensin -haun perusoperaatiot ilmenevät tietorakenteissasi.

23. Tarkastellaan tehtävän 18 ongelmaa. Huomaa, että jos solmua u tutkittaessa löytyy kaari (u,v) missä v on harmaa, niin syvyyteen ensin -haku raportoi silmukan, mutta muuten sen toiminta on täysin samanlainen kuin mitä se olisi ollut jos graafissa ei olisi kaarta (u,v) lainkaan.

Opettaja väittää, että yksi syvyyteen ensin -haku riittää, jos aina silmukan löytyessä varmistetaan, että joko nykyisessä solmussa on poliisi tai siinä harmaassa solmussa on poliisi, johon löytyi kaari nykyisestä solmusta. Varmistaminen voi tapahtua toteamalla, että jommassa kummassa niistä on jo poliisi, tai sijoittamalla poliisi jompaan kumpaan niistä.

Mistä voidaan olla varmoja, että tämä riittää? Tämä on tärkeä kysymys, koska kohdassa 18 (d) todettiin esimerkin avulla, että ei riitä, että saman silmukan jossakin muussa solmussa on jo poliisi: (1,2), (2,2), (2,3), (3,1), (1,3). Vastaus on melkein, mutta ei aivan, suoraan poimittavissa täältä: `http://users.jyu.fi/~ava/TIES3510.pdf`. Tämän kohdan tavoitteena on pistää miettimään siellä esitettyä ajatuskulkua huolellisesti.

24. Kirjoita pseudokoodilla tai jollain ohjelmointikielellä aliohjelma, joka lukee standardiinputista yhden tai useampia merkkejä, tulkitsee ne alla kuvatulla tavalla ja palauttaa yhden tulkitun merkin. Aliohjelman on luettava täsmälleen niin monta merkkiä kuin tarvitaan yhden tulkitun merkin tuottamiseksi. Jos syöte loppuu kesken tai tulee virhetilanne, aliohjelman pitää palauttaa `\0` eli se merkki, jonka lukuarvo on 0.

Merkki `+` korvataan välilyönnillä. Merkki `%` aloittaa kolmen merkin kokonaisuuden, jossa seuraavat kaksi merkkiä esittävät lukuarvon heksadesimaalina. Heksadesimaaliluvussa voi esiintyä numeromerkkejä ja/tai pieniä ja/tai isoja kirjaimia väliltä `A`, ..., `F`. Aliohjelma palauttaa lukuarvoa vastaavan merkin. Muut merkit palautetaan sellaisinaan.

HTML-lomakkeen kenttään kirjoitettu data tulee aikatauluohjelmalle tällä tavalla koodattuna. Koodauksesta on lisää tietoa Wikipedian sivulla `https://en.wikipedia.`

`org/wiki/Percent-encoding`. Oikeassa tilanteessa tarvitsee myös ohittaa HTML-lomakkeen kentän nimi ja tunnistaa kentän loppu, mutta tehtävän helpottamiseksi unohda ne. Tämä ei ole niin iso tehtävä kuin miltä saattaa näyttää: aikatauluohjelman vastaava osuus on alle 20 riviä.

Käytä haluamaasi ohjelmointikieltä tai pseudokoodia. Pseudokoodin tapauksessa käytä seuraavia C++:n toimintoja: `std::cin.get()` lukee ja palauttaa yhden merkin standardi-inputista, ja sen kutsumisen jälkeen `std::cin` palauttaa `true` tai `false` sen mukaan onnistuiko juuri tehty lukeminen. Voit siis kirjoittaa `if( std::cin ){ ... }`. Saat käyttää valmista muunnosta merkistä sen lukuarvoksi, mutta ohjelmoi sitä isommat muunnokset itse, kuten muunnos heksadesimaalikoodista lukuarvoksi. Siis älä käytä siihen ohjelmointikielen tai kirjastojen valmiita aliohjelmia tms. Aliohjelmasi saa käyttää itse kirjoittamiasi apualiohjelmia yms.

Jos haluat, niin saat hyödyntää seuraavia tietoja: Syötemerkit ovat 8-bittisiä. Ne syötemerkit, joiden eniten merkitsevä bitti on 0, tulkitaan ASCII-koodin mukaisesti. Ne syötemerkit, joiden eniten merkitsevä bitti on 1, palautetaan sellaisinaan. Jos jokin on standardin mukaan virhe mutta aliohjelmasi pystyy järkevästi tulkitsemaan sen merkiksi, niin saat valita, palauttaako aliohjelmasi `\0` vai k.o. merkin.

Perustele, että aliohjelmasi toimii oikein.

Viikko 7 (17.10.2021)

25. Veppisivu `http://users.jyu.fi/~ava/m_paattely.html` on tarkoitettu päättelyyn liittyvien käsitteiden opettelemiseen. Sen lopussa on erityisen yksinkertainen ohjelman oikeaksi osoittaminen. Osa veppisivun kohdista on tarkoitettu motivoimaan tulossa olevaa asiaa, eikä siksi välttämättä ratkea ensimmäisellä yrittämällä. Osa kohdista on vaikeampia kuin muut, jotta innokkaimmillekin opiskelijoille riittäisi haastetta.

Tee se läpi tai ainakin niin pitkälle kuin kohtuullisella ajankäytöllä pystyt (se on melko pitkä!). Jos jokin kohta ei kohtuullisella vaivalla ratkea, niin jatka eteenpäin ja yritä sitä myöhemmin uudelleen. Jos se ei toisellakaan yrittämisellä ratkea, saat jättää sen tekemättä.

Kerro kuinka pitkälle teit sitä. Anna palautetta kohdista, jotka eivät mielestäsi toimi teknisesti tai pedagogisesti. Kerro palautteessasi *kaikkien* niiden kohtien numerot, joita et saanut toisellakaan yrittämisellä tehdyksi. Myös muunlainen palaute on tervetullutta.

Veppisivu on aivan uusi ja vielä niukasti testattu. Jos huomaat vakavia virheitä, niin lähetä niistä heti tieto sähköpostitse, jotta voin korjata ne nopeasti. Veppisivua on muutenkin tarkoitus testata kuluvan viikon aikana, joten siihen saattaa tulla korjauksia kesken kaiken.

Jos esitietosi eivät riitä veppisivun tekemiseen, niin yritä `http://users.jyu.fi/~ava/t_propositiot1.html` ja ehkä `http://users.jyu.fi/~ava/m_logiikka_taso.html`. Jos joudut tekemään paljon esitietoja, saat jättää `m_paattely.html` nyt kokonaan tekemättä — teet sen sitten ensi viikolla.

26. Vertaa veppisivulta `http://users.jyu.fi/~ava/TIES3510_ato.html` löytyviä tietorakenteita tehtävässä 22 suunnittelemiisi. Mikä on omissa tietorakenteissasi huonommin? Mikä on omissa tietorakenteissasi paremmin? Huom! Veppisivulta löytyvän ohjelman nimistöä ja kommentteja ei ole hiottu viimeisen päälle. Älä välitä `pula_`-alkuisista muuttujista (niillä tuotetaan ehdotuksia, mihin kannattaa lisätä aikoja).

27. Aliohjelman `inp_cgi_decode()` kutsuja on ennen kutsua lukenut merkin syötteestä muuttujaan `inp_chr`. Aliohjelman `inp_cgi_decode()` on joko jätettävä se ennalleen tai muutettava sitä kuten HTML-lomakkeelta tulevan tiedon koodauksen purku vaatii. Tarvittaessa `inp_cgi_decode()` lukee lisää merkkejä syötteestä. Toiminto `std::cin.get()` lukee syötteestä seuraavan merkin. Toiminnolla `std::cin` voi testata, onnistuiko edellinen merkin lukeminen. Se palauttaa `false`, jos syöte oli loppunut.

```
/* From hexadecimal character to a value */
inline char from_hex( char ch ){
  if( 'a' <= ch && ch <= 'f' ){ return ch - 'W'; }
  if( 'A' <= ch && ch <= 'F' ){ return ch - '7'; }
  if( '0' <= ch && ch <= '9' ){ return ch - '0'; }
  return '\x10';
}

/* Cgi-decode the input character, reading more characters if necessary. */
inline void inp_cgi_decode(){
  if( inp_chr == '+' ){ inp_chr = ' '; }
  else if( inp_chr == '%' ){
    char ch = from_hex( std::cin.get() );
    inp_chr = from_hex( std::cin.get() );
    if( !std::cin || ch == '\x10' || inp_chr == '\x10' ){ inp_chr = '\0'; }
    else{ inp_chr |= ch << 4; }
  }
}
```

Vertaa tehtävässä 24 kirjoittamaasi aliohjelmaa tähän. Mikä on omassa aliohjelmassasi huonommin? Mikä on omassa aliohjelmassasi paremmin? Älä välitä siitä, että tiedonsiirto aliohjelman ja kutsujan välillä on tässä toteutettu eri lailla kuin tehtävässä 24.

28. Mitä virheitä löydät `http://users.jyu.fi/~ava/TIES3510.pdf` luvussa 3.1 esitellystä Quicksortin toteutuksesta? Minkälaisia ei-toivottuja toimintoja ne voivat aiheuttaa?

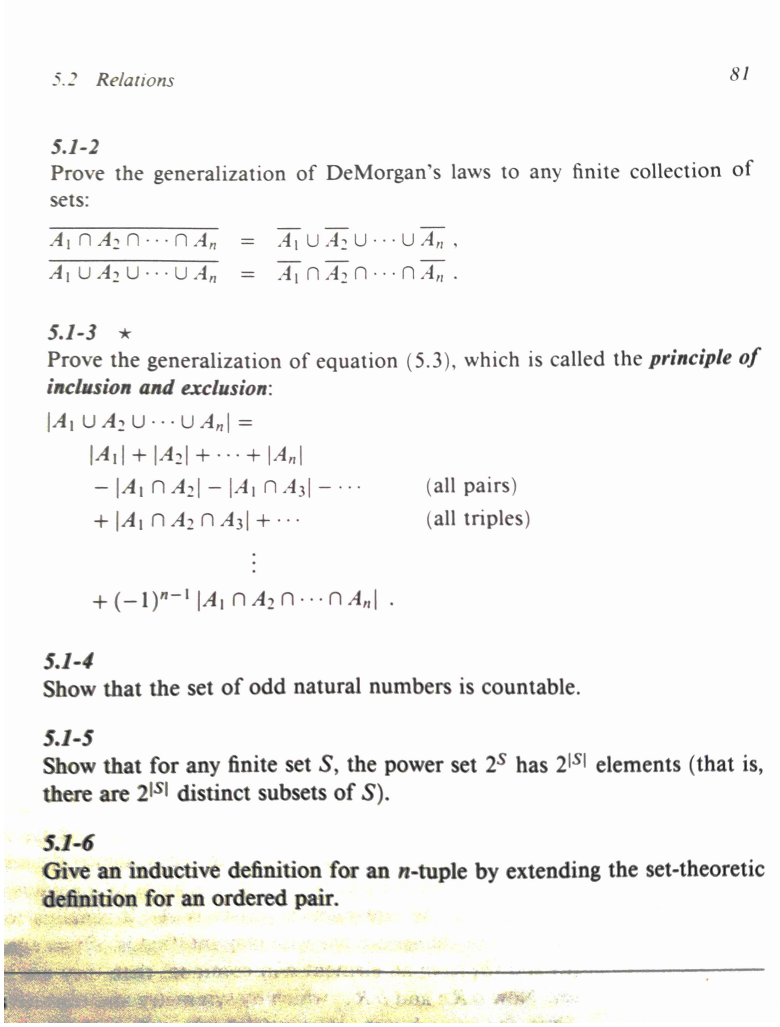
Viikko 8 (24.10.2021) — tämä on viimeinen viikko

29. Tarkastellaan vieläkin aikojen leveyteen ensin -jakoalgoritmia. Kunakin ajankohtana voidaan järjestää enintään yksi kuulustelu. Hakijat käsitellään järjestyksessä A B C D E ja kullekin hakijalle kelpaavat ajat käsitellään järjestyksessä 1 2 3 4 5. Hakijoille kelpaavat ajat ovat seuraavat: A 1 2, B 2 3, C 4 5, D 1 4, E 5. Aluksi kellään ei ole aikaa. Sitten algoritmi käsittelee A:n, jonka jälkeen A:lla on aika 1 ja muilla ei ole aikaa. Sitten algoritmi käsittelee B:n. Jatka simulaatio loppuun asti niin että jokaisen hakijan käsittelyn jälkeen kirjoitat luettelon siitä, mikä aika kullakin hakijalla on vai onko mikään. Kerro vastauksessasi nämä luettelot.

Vielä tehtävän 21 vastauksissa esiintyi vääriä käsityksiä algoritmin toiminnasta, ainakin (1) “Kun kaksi hakijaa kilpailee jostain ajasta niin aiemmin käsitelty hakija pitää sen” ja (2) ”Oletetaan, että hakijalle A on aikaa jouduttu muuttamaan niin monta kertaa, että hänestä lähtevä kaari osoittaa viimeiseen hänelle kelpaavaan aikaan. Oletetaan lisäksi, että nyt löydetään toinen hakija X, jolle on pakko antaa tämä viimeinen kelpaava aika.

. . . eli ei ole olemassa aikataulua, jossa sekä käsiteltävänä oleva hakija että jokainen ajan jo saanut hakija saa ajan.” Yllä oleva syöte on suunniteltu vastaesimerkiksi näille väitteille. Kerro vastauksessasi tilanne, jossa (1) ei toteudu sekä tilanne, jossa (2) ei toteudu (huomaa että (2):n A ei ole välttämättä tehtävän A).

30. Monien merkkien mukaan ohjelmoinnin ja ohjelmistotekniikan koulutus on huippuyliopistoja lukuun ottamatta maailmanlaajuisesti vaikeuksissa. Tässä ote vuonna 1991 ilmestyneen, sittemmin hyvin laajasti käytetyn oppikirjan Cormen–Leiserson–Rivest: Introduction to Algorithms joukko-oppia käsittelevän osuuden ensimmäisistä harjoitustehtävistä:



5.2 *Relations* 81

5.1-2
Prove the generalization of DeMorgan's laws to any finite collection of sets:

$$\overline{A_1 \cap A_2 \cap \cdots \cap A_n} = \overline{A_1} \cup \overline{A_2} \cup \cdots \cup \overline{A_n} ,$$
$$\overline{A_1 \cup A_2 \cup \cdots \cup A_n} = \overline{A_1} \cap \overline{A_2} \cap \cdots \cap \overline{A_n} .$$

5.1-3 ⋆
Prove the generalization of equation (5.3), which is called the ***principle of inclusion and exclusion***:

$$|A_1 \cup A_2 \cup \cdots \cup A_n| =$$
$$|A_1| + |A_2| + \cdots + |A_n|$$
$$- |A_1 \cap A_2| - |A_1 \cap A_3| - \cdots \quad \text{(all pairs)}$$
$$+ |A_1 \cap A_2 \cap A_3| + \cdots \quad \text{(all triples)}$$
$$\vdots$$
$$+ (-1)^{n-1} |A_1 \cap A_2 \cap \cdots \cap A_n| .$$

5.1-4
Show that the set of odd natural numbers is countable.

5.1-5
Show that for any finite set S, the power set 2^S has $2^{|S|}$ elements (that is, there are $2^{|S|}$ distinct subsets of S).

5.1-6
Give an inductive definition for an n-tuple by extending the set-theoretic definition for an ordered pair.

Seuraavalla sivulla on vastaava ote vuodelta 2019, oppikirjasta Stephen Davies: A Cool Brisk Walk Through Discrete Mathematics:

2.13 Exercises

Use an index card or a piece of paper folded lengthwise, and cover up the right-hand column of the exercises below. Read each exercise in the left-hand column, answer it in your mind, then slide the index card down to reveal the answer and see if you're right! For every exercise you missed, figure out why you missed it before moving on.

1. Is the set { Will, Smith } the same as the set { Smith, Will }?	Yes indeed.
2. Is the ordered pair (Will, Smith) the same as (Smith, Will)?	No. Order matters with ordered pairs (hence the name), and with any size tuple for that matter.
3. Is the set { { Luke, Leia }, Han } the same as the set { Luke, { Leia, Han } }?	No. For instance, the first set has Han as a member but the second set does not. (Instead, it has another set as a member, and that inner set happens to include Han.)
4. What's the first element of the set { Cowboys, Redskins, Steelers }?	The question doesn't make sense. There is no "first element" of a set. All three teams are equally members of the set, and could be listed in any order.

Julkaisussa Pears & al.: A Survey of Literature on the Teaching of Introductory Programming (2007) päädyttiin seuraavaan johtopäätökseen: "The goal of this paper has been to give an overview of research issues relating to the teaching of introductory programming, with specific reference to curriculum, pedagogy, and languages and tools for supporting learning. ... We conclude that despite the large volume of literature in this area, there is little systematic evidence to support any particular approach. For that reason, we have not attempted to give a canonical answer to the question of how to teach introductory programming."

Teidän ohjelmointitaitonne lienee ainakin yhtä hyvä, veikkaan että jopa parempi, kuin keskimääräisellä samassa opintojen vaiheessa olevalla JY:n tietotekniikan opiskelijalla, päätellen mm. siitä, että löysitte hyvin tehtävän 28 Qucksortin virheitä. Kuitenkin uskon, että tarvetta olisi vielä vahvemmalle osaamiselle. Melkein jokaisen perinnönjakoohjelma laski väärin jo siinä tilanteessa kun vainajalla on elossa olevia jälkeläisiä. Ensimmäiset algoritmi- ja käyttöliittymäehdotuksenne tuskin olisivat toimineet riittävän hyvin kesän 2020 pääsykokeiden haastattelujen aikataulujen laatimisessa. Aikataulualgoritmin toiminnasta oli vääriä käsityksiä vielä tehtävän 21 vastauksissa.

Miten ohjelmoinnin ja algoritmien opetusta voisi mielestäsi parantaa? Voit, mutta ei ole pakko, vastauksessasi pohtia mm. seuraavia asioita: kuinka paljon ja millaisia ohjelmointitehtäviä? Kuinka paljon ja millaisia muita harjoitustehtäviä, esimerkiksi annetusta ohjelmakoodista päättelemistä, suoritusaikojen laskemista $O()$-merkinnällä, valmiin tietorakenteen toteutuksen selvittämistä kokeellisesti, tai ohjelman suoritusajan mittaamista? Mikä on teoreettisten kurssien kuten Algoritmit 1 & 2 rooli? Miten opitaan laatimaan hyviä spesifikaatioita, niin että valmis tuote on loppukäyttäjien kannalta hyvä? Mitä taitoja tarvitaan, jotta saataisiin aikaan parempaa laatua kuin Sisu? Voit myös halutessa-

si kommentoida julkaisua `https://www.researchgate.net/publication/2955262_What_knowledge_is_important_to_a_software_professional`

31. (Vapaaehtoinen, ei ole pakko tehdä eikä pisteytetä vaikka tekisit) Tämän numeron alla voit antaa ehdotuksia, miten MathCheck-tehtävien ohjeistusta voisi parantaa. Ohjeita on kahta lajia: yksittäisten symbolien kirjoitusohjeita tyyliin kertomerkki kirjoitetaan `*` ja F kirjoitetaan `FF`; sekä yleisempiä ohjeita kuten että ruskeiden alueiden alta paljastuu vihje, mallivastaus tms. kun vie kursorin sen päälle; MathCheckille saa ja toisinaan kannattaa antaa tahallaan vääriä vastauksia ja koettaa oppia sen antamasta palautteesta; ja käänteisen opetuksen periaatteen vuoksi jokin kohta saattaa olla sellainen ettei sitä ole tarkoituskaan osata ensimmäisellä kertaa.

Pitäisikö ohjeiden olla jokaisella tehtäväsivulla erikseen, vai yhtenä tiedostona johon on jokaiselta tehtäväsivulta linkki? Pitäisikö aina ennen kurssin ensimmäistä MathCheck-tehtävää olla ”näin käytät MathCheckiä” -tehtävä, samankaltainen kuin nykyinen `http://users.jyu.fi/%7eava/yleista.html`?

Jossain vaiheessa keksin laittaa ohjeita ruskeiden otsikoiden taakse sivujen oikeaan yläreunaan. Luulin sen olevan hyvä idea koska se teki ohjeet käteviksi käyttää, mutta on käynyt ilmi, että ohjeita ei välttämättä hoksata etsiä sieltä. Onko siitä hyötyä? Onko sivun oikean alareunan lähellä olevasta kopioitavien symbolien luettelosta hyötyä?

Pitkään minulla oli käytäntönä laittaa joka tehtävän alkuun linkki ”Lyhyt MathCheck-ohje (uuteen välilehteen)” tänne: `http://users.jyu.fi/%7eava/brief_help.html`. Sinä aikana kun sitä käytin, ei tullut valituksia siitä, että ohjeet eivät löydy. Siihen voisi lisätä edellä mainitut yleisluontoiset asiat ja jättää sivun yläreunan ohjeet pois. Olisiko se hyvä ratkaisu?

loppu