Johdatus tekoälyyn, kevät 2017, Teemu Roos **Harjoitus 1**

Yksi harjoitustehtäväpiste vastaa 2/3 koepistettä. Harjoitustehtäväpisteitä on jaossa yhteensä $6 \times 6 = 36$ *(vastaa 24 koepistettä). Arvostelussa harjoitustehtäväpisteitä luetaan hyväksi enintään 20, joten täydet lisäpisteet saa tekemällä noin 83% tehtävistä.*

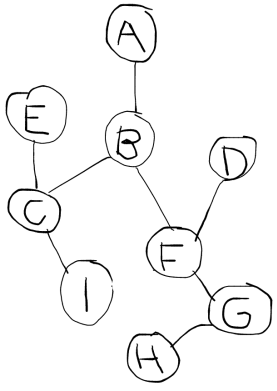
Tehtävä 1. Tekoälytutkimus vs. sci-fi (1 piste)

Etsi yksi tekoälyä ja sen tutkimusta käsittelevä uudehko tieteellinen artikkeli, jonka pääsisällöstä (kysymyksen asettelu, johtopäätökset) saat kohtalaisen kuvan. Hyviä artikkeleita löytyy mm. AAAI-, IJCAI-, ja ECAI-konferenssien julkaisuista (*Proceedings*)[1] ja esim. *AI Magazine* -lehden[2] kaltaisista laajemmalle lukijakunnalle tarkoitetuista julkaisusarjoista.

Lue artikkeli läpi ja tuo mukanasi laskuharjoituksiin. Koita miettiä vastaukset muun muassa seuraaviin kysymyksiin:

1. Minkätyyppistä kysymystä/ongelmaa käsitellään?
2. Sopiiko aihe tämän kurssin aihepiireihin?
3. Minkälaisen vaikutelman artikkeli antaa (nykyisestä) tekoälytutkimuksesta?
4. Minkälaisia opintoja tarvitaan, jotta lukemasi artikkelin voi ymmärtää?
5. Kuinka realistisilta artikkelin perusteella vaikuttavat sci-fi kirjallisuuden ja elokuvien uhkakuvat, joissa teknologia ja tekoäly kääntyvät ihmiskuntaa vastaan kuten esimerkiksi Terminator-elokuvissa?

Tehtävä 2. Etsintä: leveys- ja syvyyssuuntainen haku (2 pistettä)



Ajatellaan oheista verkkoa.

Esitä verkon leveys- ja syvyyssuuntainen läpikäynti alkaen solmusta A, kun maalisolmua on H.

Esitä luennon pseudokoodialgoritmin *Solmulista*:n sisältö kussakin etsinnän vaiheessa, kummallakin etsintätavalla (leveys- ja syvyyssuuntaisella). 1 etsintätapa = 1 piste.

[1] `www.ijcai.org`, `www.aaai.org`, `www.ecai2016.org`
[2] `www.aaai.org/Magazine`

Tehtävä 3. Etsintä ongelmanratkaisuna: Susi, lammas ja kaali (1 piste)

Paimen (p) kuljettaa sutta (s), lammasta (l) ja kaalia (k). Ylitettävä on joki ($\sim$), jonka yli paimen voi kulkea veneellä, johon mahtuu vain yksi kolmesta kuljetettavasta kerrallaan. Jos paimen jättää suden ja lampaan vartioimatta samaan paikkaan, susi syö lampaa suihinsa. Samoin käy, jos lammas ja kaali jätetään vartioimatta samalle rannalle. Miten paimen saa lastin joen yli?

Esitä ongelma tilasiirtymäkaaviona, jossa alkutila, jossa kaikki kolme kuljettavaa ja paimen ovat vasemmalla rannalla, esitetään merkinnällä "$slkp \sim$". Alkutilasta ainoa sallittu siirtymä on siten "$sk \sim lp$" eli paimen vie lampaan vastarannalle.

Esitä tilasiirtymäkaavio verkkona.

Etsi verkon avulla ratkaisu ongelmaan.

Tehtävä 4. Etsintä: Reittiopas, osa I (2 pistettä)

Täydennä liitteenä olevaan Java-ohjelmarunkoon tai ohjelmoi haluamallasi ohjelmointikielellä (pascal, fortran, ttk-91, ...) leveyssuuntainen haku Helsingin raitiovaunuverkossa. Toteuta sen avulla hakualgoritmi joka ottaa syötteenä lähtö- ja maalipysäkkien tunnisteet ja palauttaa pysäkkien välisen reitin, jonka varrella on minimimäärä pysäkkejä. (Leveyssuuntainen haku löytää aina tällaisen reitin.)

Raitiovaunulinjojen tiedot löytyvät liitteenä olevasta materiaalista. Java-projektin runko sisältää valmiin toteutuksen pysäkki-luokalle, joka tuntee omat naapuripysäkkinsä. Tiedostosta `verkko.json` löytyvät tiedot raitiovaunulinjoista json-muodossa, jos et halua käyttää valmista toteutusta.

Mikäli käytät valmista Java-pohjaa:

1. Lataa ja pura kurssisivulta löytyvä tiedosto ja avaa purettu Maven-projekti suosikkikehitysympäristössäsi (esim. Netbeans).

2. Toteuta hakualgoritmi luokkaan Reittiopas. Palauta reitti taaksepäin linkitettynä listana `Tila`-olioita, joista ensimmäinen osoittaa maalipysäkkiin ja jokainen tuntee pysäkin ja tilan, josta kyseiseen tilaan päästiin (viimeisen solmun `Pysakki` on lähtöpysäkki ja `edellinen` on `null`).

   Esimerkiksi kutsuttaessa reittioppaan `haku()`-metodia lähtöpysäkillä `1250429(Metsolantie)` ja maalipysäkillä `1121480(Urheilutalo)` ja kelaamalla palautettua `Tila`-oliota taaksepäin saadaan reitti:

   ```
   1121480(Urheilutalo)[MAALI] -> 1121438(Brahenkatu) -> 1220414(Roineentie)
   -> 1220416(Hattulantie) -> 1220418(Rautalammintie) -> 1220420(Mäkelänrinne)
   -> 1220426(Uintikeskus) -> 1173416(Pyöräilystadion) -> 1173423(Koskelantie)
   -> 1250425(Kimmontie) -> 1250427(Käpylänaukio) ->1250429(Metsolantie)[LÄHTÖ]
   ```

3. Voit käyttää algoritmisi testaamiseen yllä olevan reitin testaavia junit-testejä, jotka tulevat projektin mukana.

(Vinkki python-koodaajille json-tiedoston lukua varten: `import json`.)