Johdatus tekoälyyn, syksy 2014, Teemu Roos **Harjoitus 1**

Voit saada kustakin harjoituskerrasta max. 5 pistettä eli yhteensä $6 \times 5 = 30$ *pistettä. Kokeessa näistä luetaan hyväksi enintään 25, joten täydet lisäpisteet saa tekemällä noin 83% tehtävistä.*

Tehtävä 1. Tekoälytutkimus vs. sci-fi

Etsi yksi tekoälyä ja sen tutkimusta käsittelevä uudehko artikkeli, blogikirjoitus, tms., jonka pääsisällöstä (kysymyksen asettelu, johtopäätökset) saat kohtalaisen kuvan. Lue se läpi ja tuo mukanasi laskuharjoituksiin. Koita miettiä vastaukset muun muassa seuraaviin kysymyksiin:

- Minkätyyppistä kysymystä/ongelmaa käsitellään?
- Sopiiko aihe tämän kurssin aihepiireihin?
- Minkälaisen vaikutelman artikkeli antaa (nykyisestä) tekoälytutkimuksesta?
- Minkälaisia opintoja tarvitaan, jotta lukemasi artikkelin voi ymmärtää?

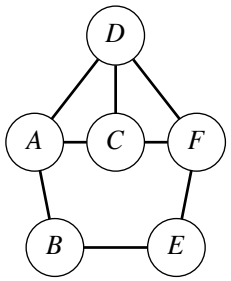
Kuinka realistisilta artikkelin perusteella vaikuttavat sci-fi kirjallisuuden ja elokuvien uhkakuvat, joissa teknologia ja tekoäly kääntyvät ihmiskuntaa vastaan kuten esimerkiksi Terminator-elokuvissa?

Tehtävä 2. Etsintä: leveys- ja syvyyssuuntainen haku

Ajatellaan oheista verkkoa.

Esitä verkon leveys- ja syvyyssuuntainen läpikäynti alkaen solmusta D, kun maalisolmua ei ole. Etsintä ei siis pääty, ennen kuin kaikki solmut ovat käsitelty.

Esitä luennon pseudokoodialgoritmin *Solmulista*:n sisältö kussakin etsinnän vaiheessa, kummallakin etsintätavalla (leveys- ja syvyyssuntaisella).



Tehtävä 3. Etsintä ongelmanratkaisuna

Esitä verkkona kolmen levyn Hanoin torni -pelin tilat ja niiden väliset siirtymät (ks. esim. `fi.wikipedia.org/wiki/Hanoin_torni`).

Etsi verkon avulla ratkaisu, joka vaatii minimimäärän siirtoja.

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:

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

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]
```

Projekti sisältää valmiit junit-testit tätä esimerkkireittiä varten.