Johdatus tekoälyyn, syksy 2011, Teemu Roos **Harjoitus 2**

Huom: Voit saada tästä harjoituskerrasta max. 8 pistettä.

Tehtävä 1. Minimax ja alpha-beta. (1 piste)

a) Esitä pelipuu alkaen ristinollatilanteesta

```
O|O|X
-+-+-
 |X|
-+-+-
O|X|
```

kun seuraavana vuorossa on 'X'. Lopputilojen kustannus on +1, kun 'X' voittaa, -1, kun 'O' voittaa, ja 0 muuten.

b) Laske solmujen min- ja max-arvot luennolla esitetyn minimax-algoritmin avulla. Mitkä siirrot ovat optimaaliset ja mihin lopputulokseen ne johtavat?

c) Sovella alpha-beta-karsintaa edellisen tehtävän puuhun ja esitä α- ja β-arvojen tila kussakin suorituksen vaiheessa. Karsitaanko joitain alipuita? Miten lapsisolmujen läpikäyntijärjestys vaikuttaa asiaan?

Tehtävä 2. Syvyysrajoitettu alpha-beta. (1-2 pistettä)

Toteuta (ohjelmoimalla) alpha–beta-etsintä seuraavan pseudokoodin mukaisesti:

```
ALPHA-BETA-ARVO(Solmu):
    return(MAX-ARVO(Solmu,-Inf,+Inf))

MAX-ARVO(Solmu,alpha,beta):
    if LOPPUTILA(Solmu) return(ARVO(Solmu))
    v=-Inf
    for each Lapsi in LAPSET(Solmu,'X')
        v=MAX(v,MIN-ARVO(Lapsi,alpha,beta))
        if v>=beta return(v)
        alpha=MAX(alpha,v)
    return(v)
```

```
MIN-ARVO(Solmu,alpha,beta):
    if LOPPUTILA(Solmu) return(ARVO(Solmu))
    v=+Inf
    for each Lapsi in LAPSET(Solmu,'O')
        v=MIN(v,MAX-ARVO(Lapsi,alpha,beta))
        if v<=alpha return(v)
        beta=MIN(beta,v)
    return(v)
```

missä `LAPSET(Solmu,'X')` ja `LAPSET(Solmu,'O')` palauttavat tilanteessa `Solmu` kunkin pelaajan vuorolla mahdolliset seuraavat pelitilanteet. Testaa algoritmia ratkaisemalla tehtävä 1.

Ylimääräinen piste: Muokkaa algoritmia lisäämällä siihen syvyysrajoite, joka rajoittaa sisäkkäisten funktiokutsujen määrää. Kun syvyysrajoite saavutetaan, kutsutaan arvotusfunktion `ARVO(Solmu)` asemesta heuristista arviointikriteeriä `ARVIO(Solmu)`, joka liittää pelitilanteeseen `Solmu` arvon väliltä $(-1, +1)$ sen mukaisesti, kuinka lähellä se on pelaajan Max (+1) tai pelaajan Min (-1) voittoa.

Voit testata algoritmia myös aloittamalla tyhjästä ruudukosta.

Tehtävä 3. Reittiopas. (3-6 pistettä)

a) (3 pistettä) Toteuta edellisen viikon tehtävien ratkaisuja muokkaamalla (tai kokonaan alusta) A*-haku seuraavan pseudokoodin mukaisesti (ks. myös luentokalvot ja Ertelin kirjan kappale 6).

```
A*ETSINTÄ(Alkusolmu):
    Solmulista = [Alkusolmu]
    while Solmulista ei tyhjä:
       Solmu = EKA(Solmulista)
       Solmulista = LOPUT(Solmulista)
       if MAALI(Solmu) return("ratkaisu", Solmu)
       Solmulista = LISÄÄ(NAAPURIT(Solmu),Solmulista)
    return("ei ratkaisua")
```

Toteuta `Solmulista` prioriteettijonona, jossa kukin tila esiintyy enintään kerran. Jos sama tila tulee lisättäväksi toisen kerran, listalla olevan solmun kustannusarvioksi tulee pienin kustannusarvio ("paras-ensin–haku").

Voit käyttää ratkaisusi testaamiseen vaikkapa Ertelin kirjan luvussa 6 annettuja kaupunkeja ja niiden välisiä etäisyyksiä, tai lukea eteenpäin kohdasta *b*...

b) (3 pistettä) Reittiopas: Lue kurssin sivulla annetut pysäkki- ja linjatiedot tiedostoista — voit halutessasi käyttää kurssin sivulta löytyviä valmiita Java-luokkia tiedostojen lukemiseen. Pysäkkitiedosto antaa luettelon pysäkeistä (ks. kuva 1) muodossa

```
nro ycoord xcoord nimi
```

Linjatiedostot antavat ko. linjan reitin muodossa

```
pysäkkinro kokonaismatka-aika-lähtöpysäkiltä
```

Oletetaan että jokainen linja lähtee lähtöpysäkiltään aina tasakymmenminuutein, eli esim. klo 10:00, 10:10, 10:20, ...

Toteuta näitä reittejä käyttäen reittiopas, joka etsii kahden pysäkin välisen nopeimman reitin.

Ohjeita.

Käytä etsinnän tilana (pysäkki, klo) -paria siten, että kukin pysäkki esiintyy prioriteettijonossa korkeintaan kerran. Heuristiikkafunktio $h(N)$ voi olla etäisyys linnuntietä tavoitepysäkille jaettuna maksiminopeudella (arvioi maksiminopeus suhteessa annettuihin tietoihin). Polkukustannus $g(N)$ on tilan kellonajan ero lähtöaikaan.

Pysäkiltä voi siirtyä kunkin sen kautta kulkevan linjan seuraavalle pysäkille. Tähän kuluva aika saadaan laskemalla yhteen odotusaika pysäkillä ja matka-aika seuraavalle pysäkille.

Hakualgoritmin `NAAPURIT(Solmu)` määritellään seuraavalla tavalla:

```
NAAPURIT(Solmu):
    Naapurilista = []
    Pysakki = Solmu.Pysakki # sijanti tilassa Solmu
    Klo = Solmu.Klo         # kellonaika tilassa Solmu
    for each Linja in LINJAT(Pysakki):
        Odotus = LASKE-ODOTUSAIKA(Linja,Pysakki,Klo)
        SeurPysakki = SEURAAVAPYSAKKI(Linja,Pysakki)
        if SeurPysakki != None:
            # matka-aika seuraavalle pysäkille
            Matka = LASKE-MATKAAIKA(Linja,Pysakki,SeurPysakki)
            Naapurilista = Naapurilista + (SeurPysakki,Klo+Odotus+Matka)
    return(Naapurilista)
```

Kun lisäät uuden solmun *Solmulista*:an, talleta lisättyyn solmuun tieto nykyisestä solmusta, jotta voit ”peruuttaa” maalisolmusta takaisin alkusolmuun. Näin saat etsinnän päätyttyä tulostettua reitin alkusolmusta maalisolmuun.

Testaa reittiopasta etsimällä parhaat reitit seuraavien pysäkkien välillä:

1. Kumpulan kampus → Töölöntori
2. Eiran sairaala → Korkeasaari
3. Ruoholahti → Pohjolanaukio

(Huom: Maksimipistemäärä tästä harjoituskerrasta on 8.)

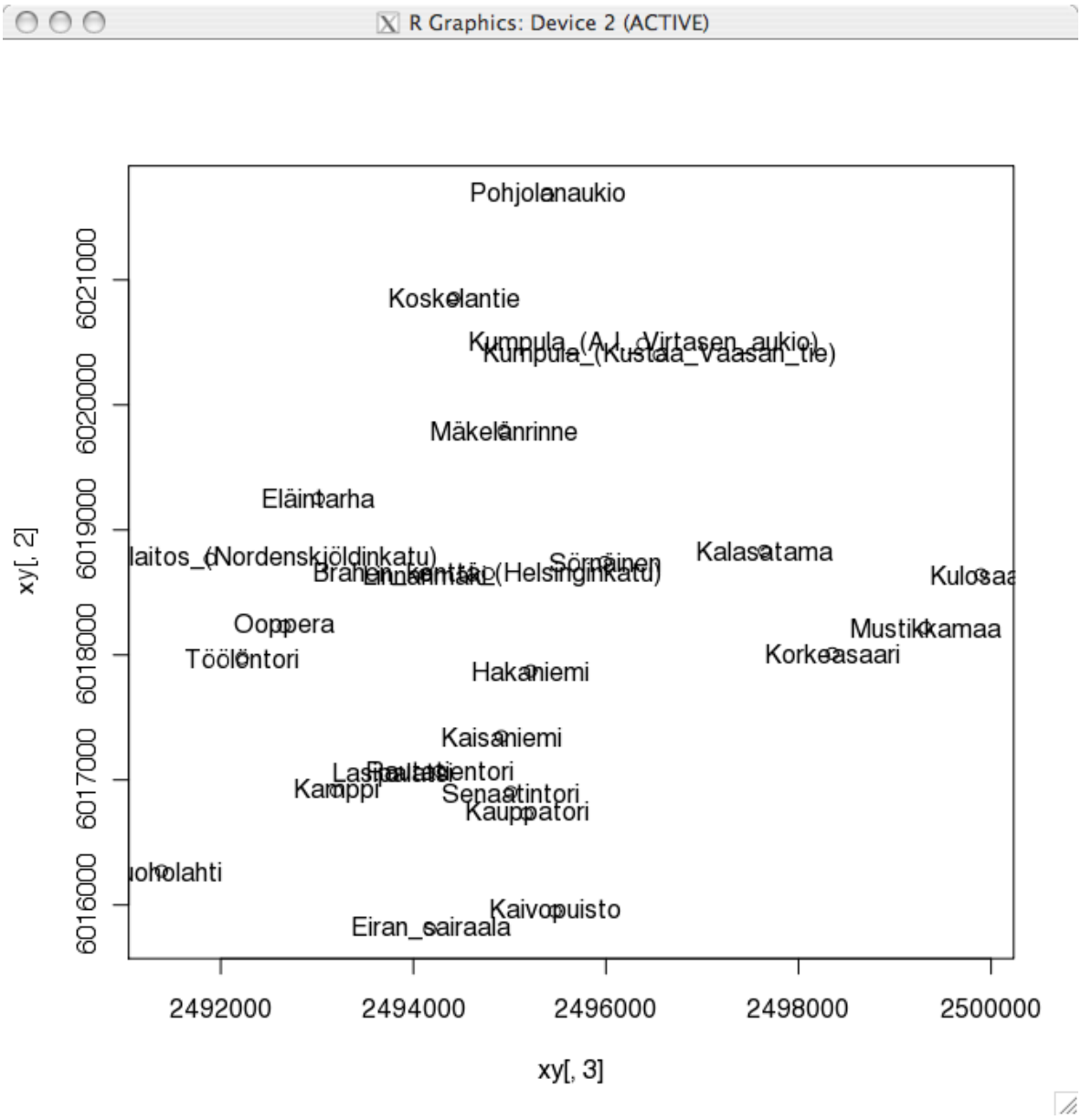

Figure 1: Pysäkit