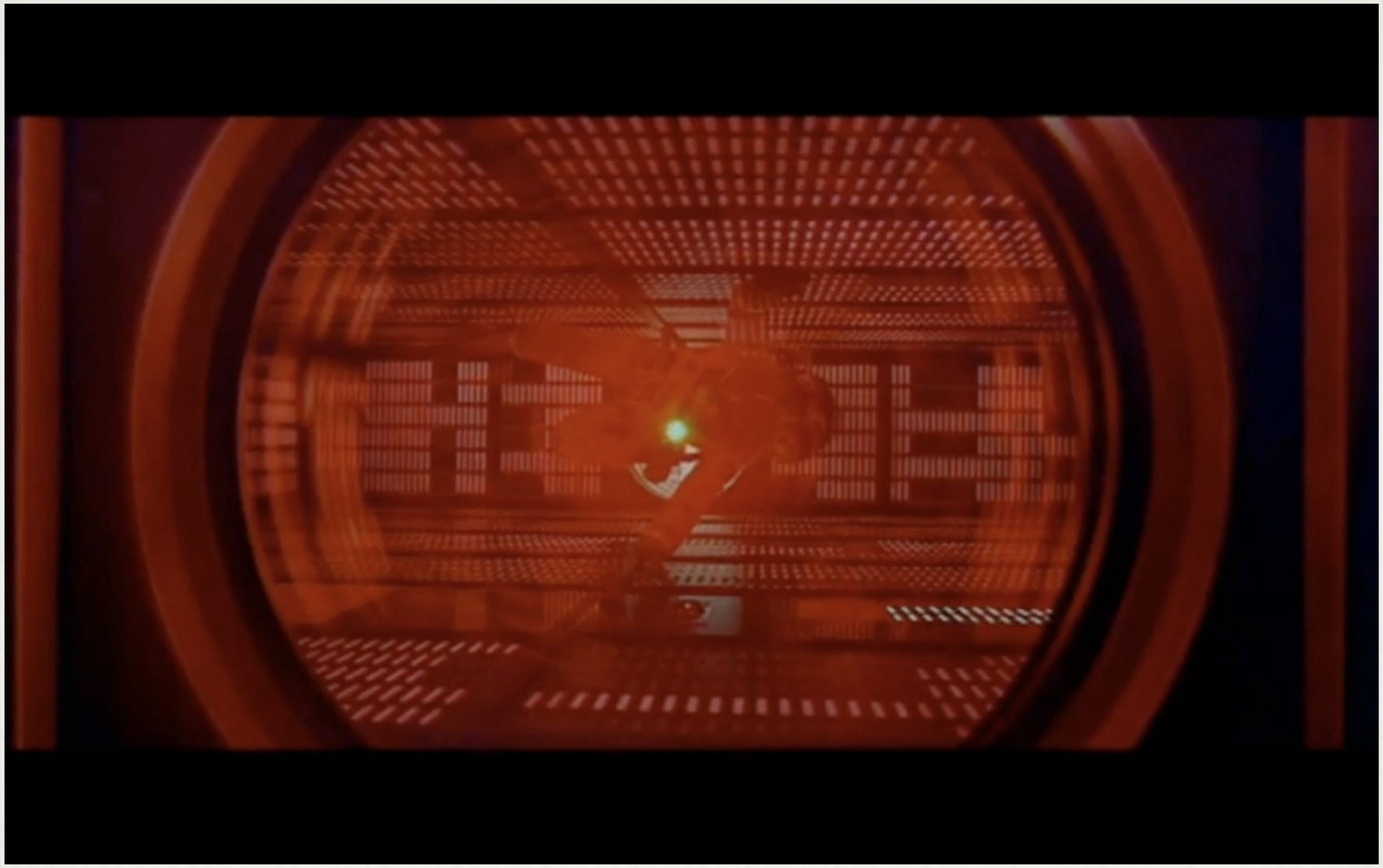


JOHDATUS TEKOÄLYYN

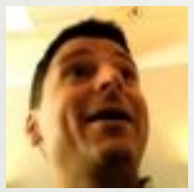
TEEMU ROOS

HELSINGIN YLIOPISTO

KURSSIN PERUSTIEDOT

- VALINNAINEN AINEOPINTOTASOINEN KURSSI, 5 OP
- PERIODI 1: 4.9.2014-17.10.2012 (7 VIIKKOA+KOE)
- LUENNOT (B123, LINUS TORVALDS -AUDITORIO):
  TO 10-12, PE 12-14
- LASKUHARJOITUKSET:
  RYHMÄ 1: TI 16-18 (MIKKO)
  RYHMÄ 2: TO 12-14 (NOORA)
  RYHMÄ 3: PE 14-16 (NOORA)
  RYHMÄ 4?
- KURSSIKOE PE 24.10.2012 KLO 9 A111/B123/CK112

TIIMI



TEEMU.ROOS@CS.HELSINKI.FI

IRC: teemuroos HUONE: A322

MIKKO KUMARA

IRC: kumikumi

NOORA IMBERG

IRC: theeta

IRC: #johtek

ESITIETOVAATIMUKSET

- TIETORAKENTEET-KURSSI
- JOHDATUS YLIOPISTOMATEMATIIKKAAN -KURSSI
- TODENNÄKÖISYYSLASKENNAN KURSSISTA HYÖTYÄ
- OHJELMOINTITAITO
- KIELI VAPAA.
  JAVAAN OHJAUSTA.

MITÄ PITÄÄ TEHDÄ?

- LUENNOILLA EI OLE PAKKO ISTUA
- KURSSIKIRJAA EI OLE -- MATERIAALI KURSSIN SIVULLA
  - KURSSIMONISTE
  - LUENTOKALVOT (SIS. LINKKEJÄ)
- LASKUHARJOITUKSET MAX 25 PISTETTÄ
- KURSSIKOE MAX 35 PISTETTÄ
- HYVÄKSYMISRAJA N. 30 PISTETTÄ

VIELÄ PARI JUTTUA

- ERILAINEN KUIN TYYPILLINEN AI-KURSSI:
  - VARHAISEMMASSA VAIHEESSA OPINTOJA
  - VÄHEMMÄN MATEMATIIKKAA (MUTTA > 0)

- RAKENTAVA KRITIIKKI TERVETULLUTTA!

- TAVOITE: 100% LÄPÄISEE

- TYÖMÄÄRÄ:
  YHTEENSÄ N. 125 TUNTIA TAI 18 TUNTIA VIIKOSSA

- "NO PAIN, NO GAIN!"

AIHEITA

1. MITÄ ON TEKOÄLY? HISTORIA JA FILOSOFIA

2. PELIT JA ETSINTÄ "GOFAI"

3. ~~LOGIIKKA (OHJELMOINTI)~~

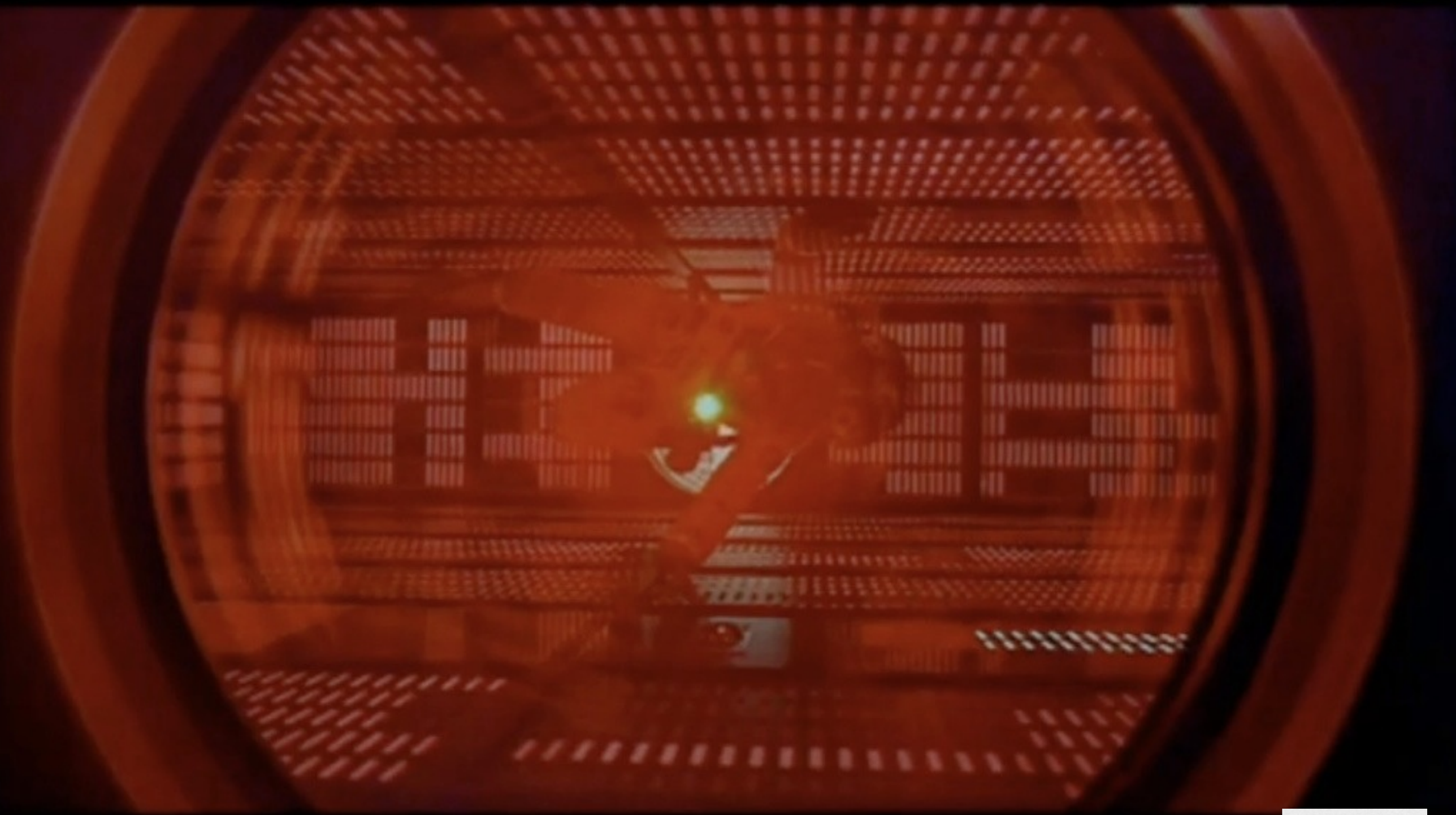
4. ROBOTIIKKA JA SIGNAALINKÄSITTELY "MODERN AI"

5. KONEOPPIMINEN JA PÄÄTTELY EPÄVARMUUDEN VALLITESSA

6. ~~LUONNOLLISEN KIELEN KÄSITTELY~~

KESKUSTELUA

- TEKOÄLY KULTTUURISSA
  - _____SKYNET_____
  - _____HAL9000_____
  - _____TERMINATOR_____
  - _____CLAP-TRAP (BORDERLANDS)_____
  - _____GLADOS (PORTAL)_____

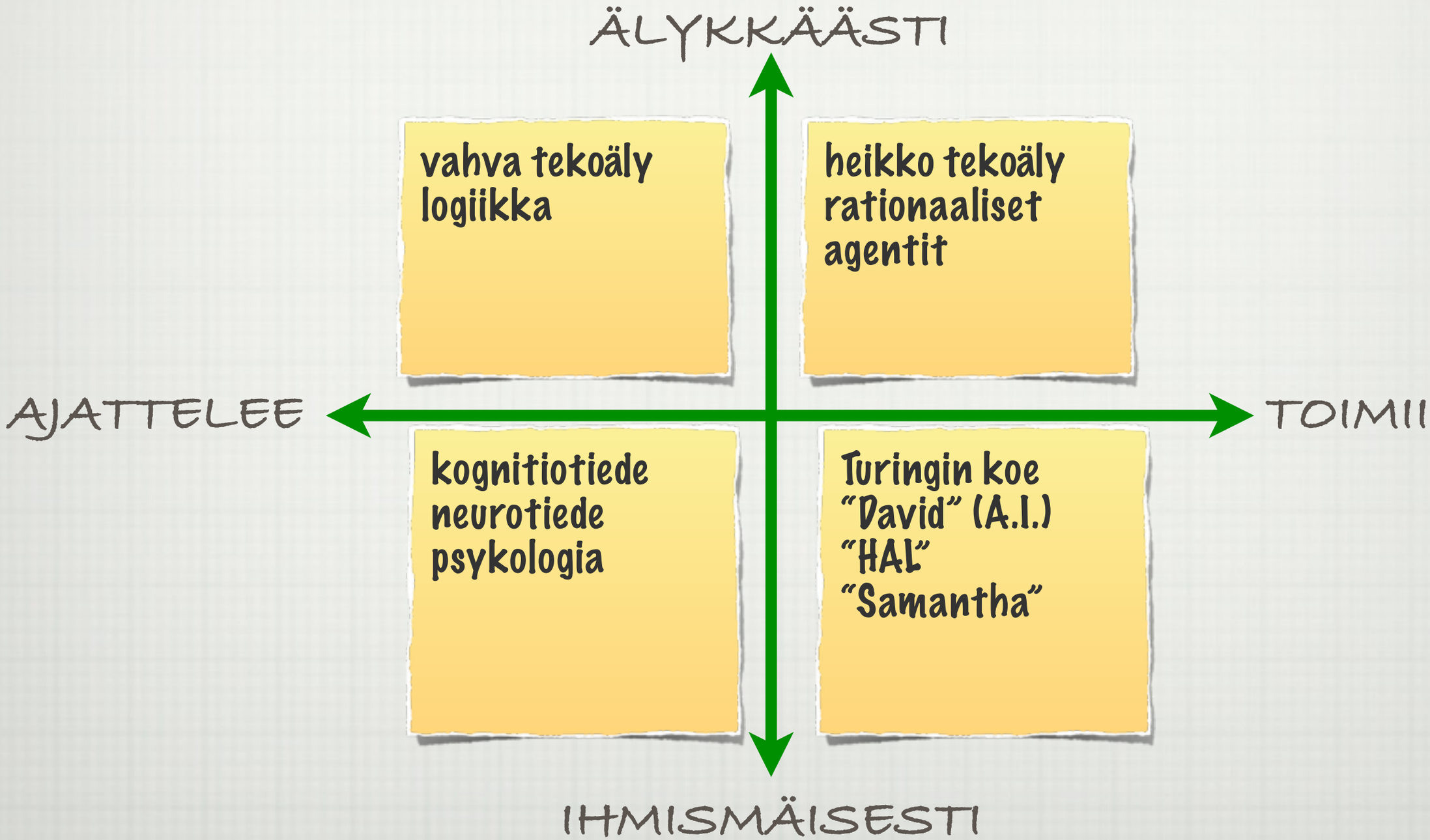


Deactivati

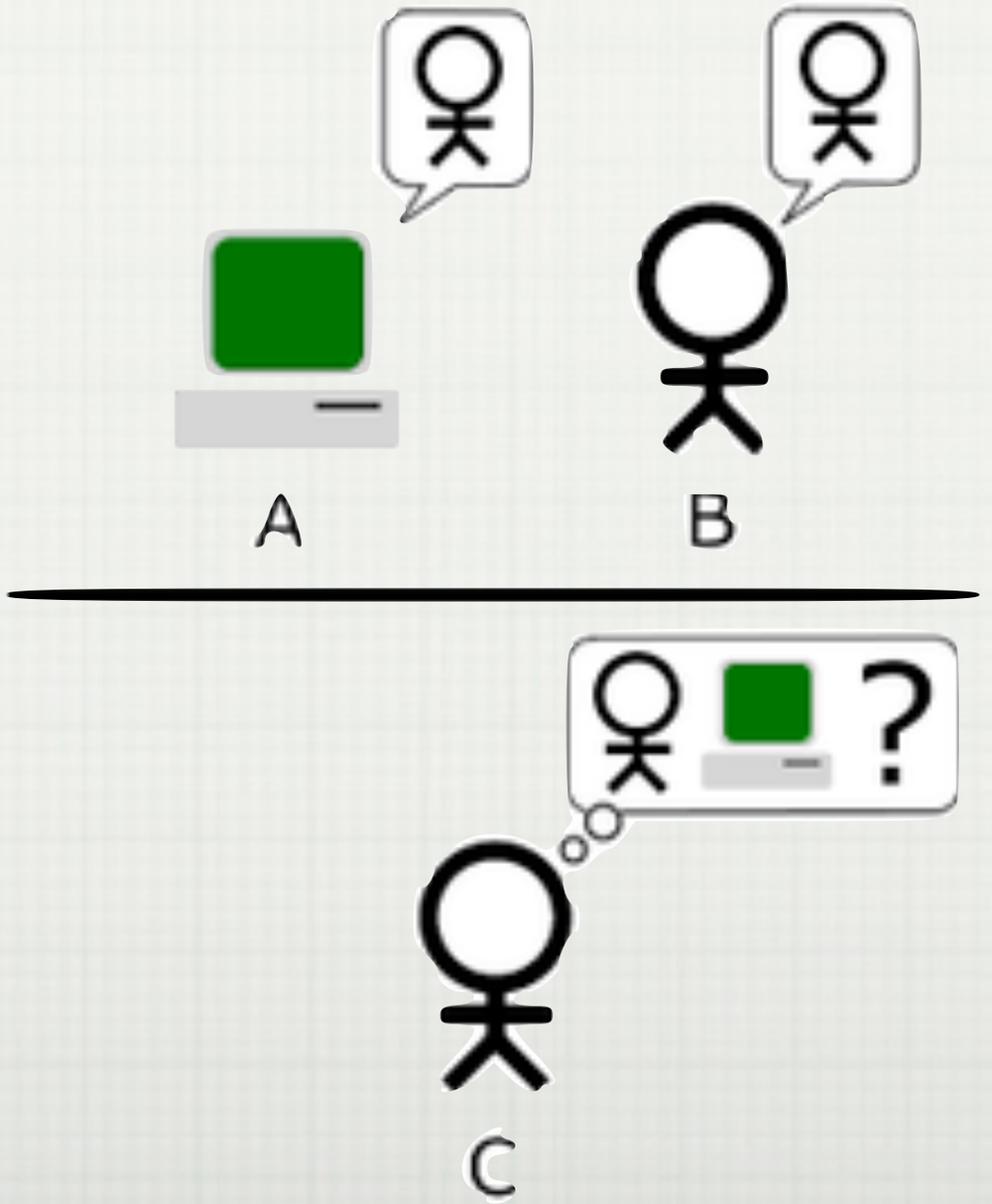
KESKUSTELUA

- MITÄ KAIKKEA HAL/SAMANTHA OSAA?
- MITÄ NÄISTÄ EI OSATA VIELÄ TOTEUTTAA?
- ONKO HALILLA/SAMANTHALLA TIETOISUUS?

TEKOÄLYN FILOSOFIAA



TOIMII IHMISMÄISESTI: TURINGIN TESTI




DELL
SAMSUNG
Are you good?
Artificial Intelligence
A Modern Approach

:13 / 1:24

KIINALAINEN HUONE

- VOIKO TOIMIA ÄLYKKÄÄSTI ILMAN ETTÄ AJATTELEE?
- TIETOISUUS?

MITÄ TEKOÄLY OIKEASTI ON?

KONENÄKÖ

REITINOPTIMOINTI

KONEOPPIMINEN

NLP

PUHE

PELIT

LOGIIKKA

TIEDONHAKU

TIEDON LOUHINTA

KONEKÄÄNNÖS

SUOSITTELU

MITÄ TEKOÄLY OIKEASTI ON?

Daily program | IJCAI 2013

Google Calendar | Information-Theo... | Johdatus tekoälyy... | SAPA07 Nimenhuu... | Data Science study... | Johdatus tekoälyy... | Daily program | IJ... | ▶ Her OST – Morni...

ijcai13.org/program/day/6

youtube

IJCAI-13

23rd. INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE

August 3-9, 2013, Beijing, China
Beijing International Convention Center (BICC)

HOME | CALLS | PROGRAM | COMMITTEES | REGISTRATION | ATTENDING | SPONSORS &EXHIBITS | STUDENTS &MENTORING | BEIJING

What are you looking for? Go

PROGRAM

- Conference at a glance
- Technical Program
- Workshops
- Tutorials
- Invited talks
- Doctoral Consortium
- IJCAI-13 Awards
- Angry Birds AI Competition
- Robot Competition and Exhibition
- Video competition
- Special meetings

ACCEPTED PAPERS

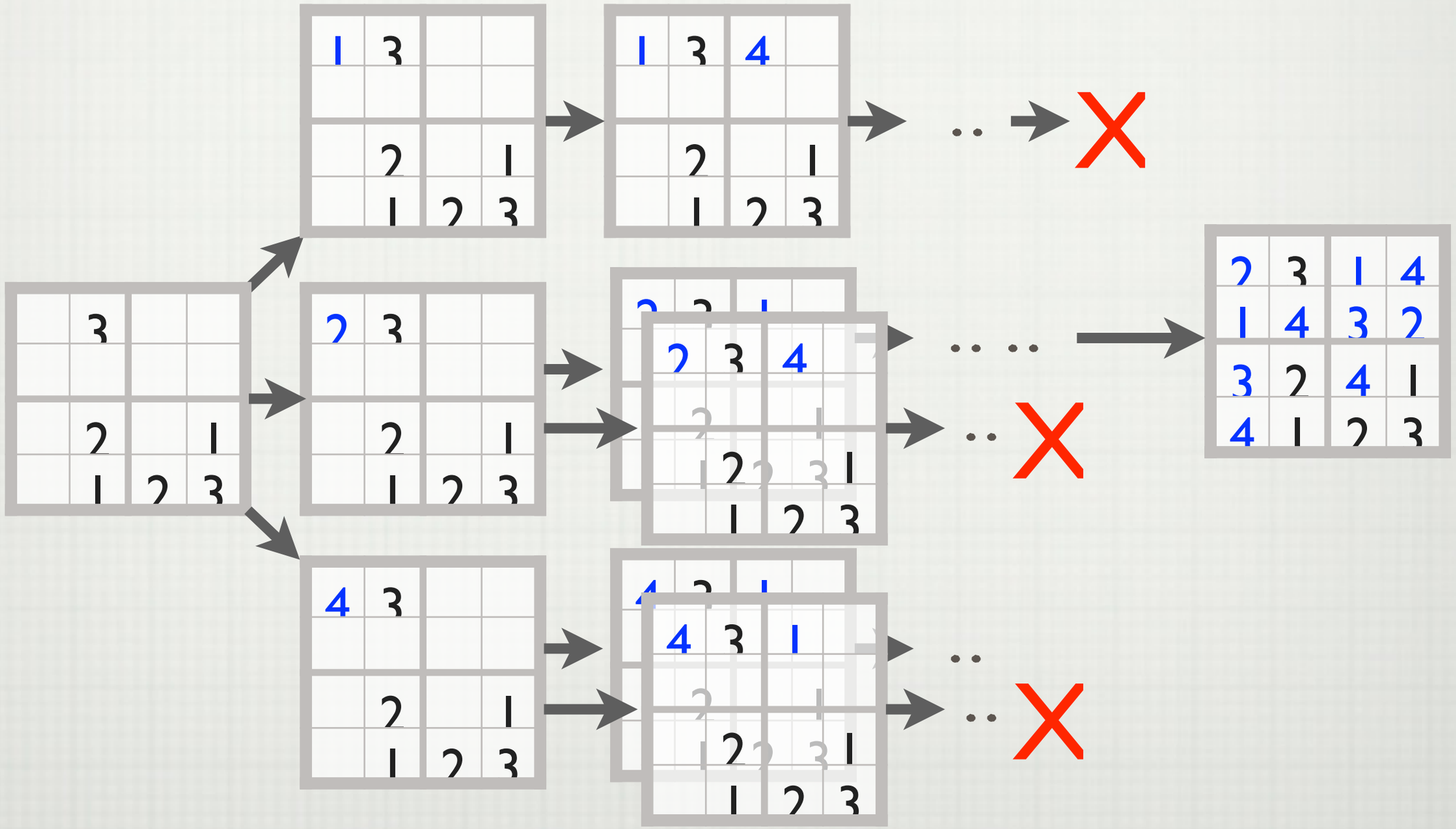
- Main and AICS track

TUESDAY, AUG 6TH 2013 (DAY 1)

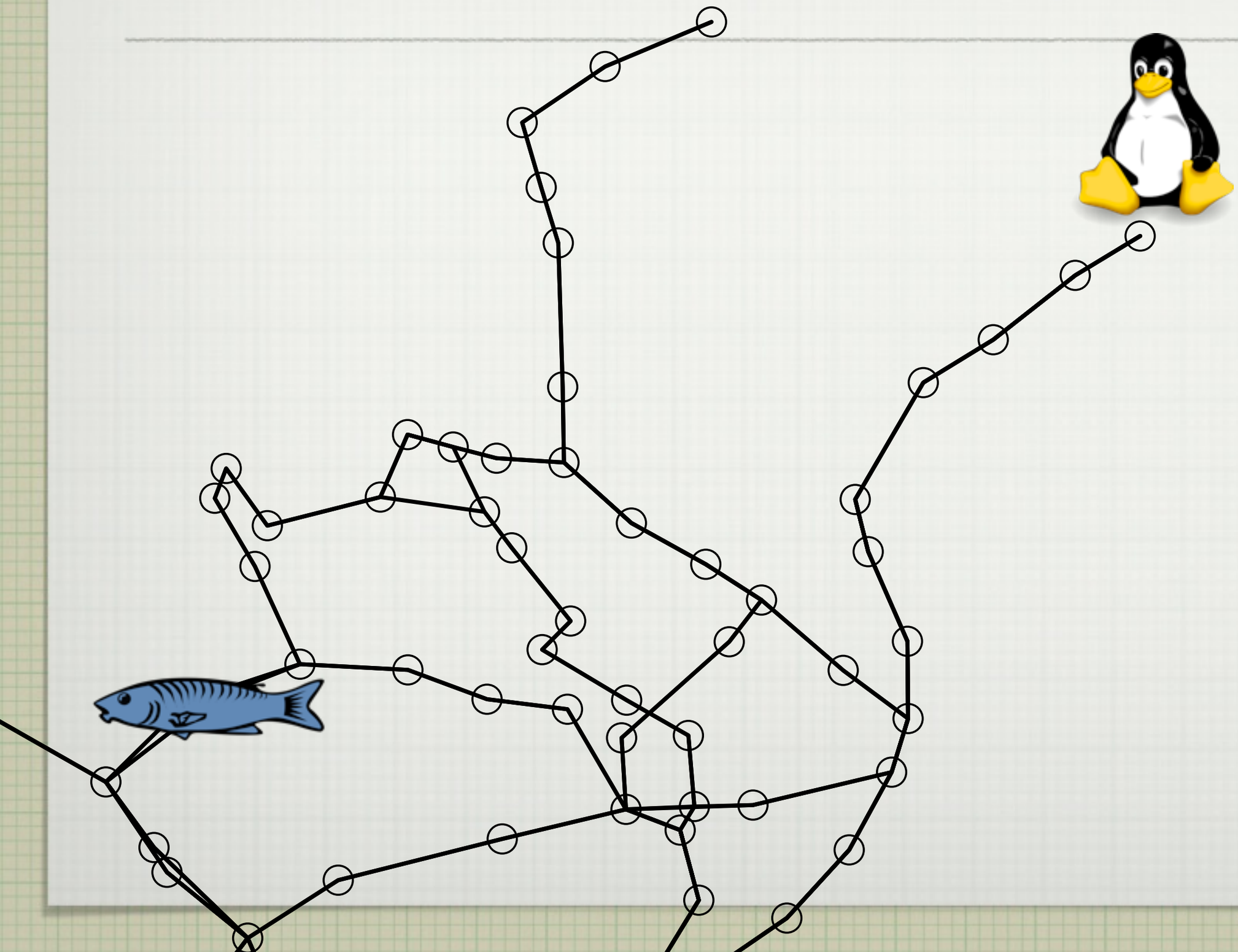
Next >>

08:30 - 09:00	Hall #2	**Welcome address and program overview**		**Francesca Rossi**
09:00 - 09:45	Hall #2 Chair: Francesca Rossi	**Invited talk**	**Computational Disaster Management**	**Pascal Van Hentenryck**
09:45 - 10:15	Coffee Break			
10:15 - 11:05	Hall #2 Chair: Kevin Leyton-Brown	**Invited talk**	**Computational Perspectives on Social Phenomena at Global Scales**	**Jon Kleinberg**
11:10 - 12:00	Hall #2 Chair: Craig Knoblock, Introduced by Raymond J. Mooney	**IJCAI-13 Computers and Thought Award**	**Towards Large-Scale Visual Recognition and Search**	**Kristen Grauman**
12:00 - 13:30	Lunch			
13:30 - 15:15	Room 201CD Session Chair: Francesca Rossi	**Award papers**		
		IJCAI 2013 distinguished paper	Bayesian Optimization in High Dimensions via Random Embeddings	Ziyu Wang, Masrour Zoghi, Frank Hutter, David Matheson, Nando de Freitas
		IJCAI 2013 distinguished paper	Maximizing Flexibility in Simple Temporal Networks	Bob Huisman, Tomas Klos, Michel Wilson, Cees Witteveen
		IJCAI-JAIR 2013 best paper award	Adaptive Submodularity: Theory and Applications in Active Learning and Stochastic Optimization	Daniel Golovin, Andreas Krause
		2012 ECCAI		

ETSINTÄ


1 3
2 1
1 2 3
1 3 4
2 1
1 2 3
..
X
3
2 1
1 2 3
2 3
2 1
1 2 3
2 3 4
2 1
2 1 3
1 2 3
.. ..
.. X
4 3
2 1
1 2 3
4 3 1
2 1
2 1 3
1 2 3
..
.. X
2 3 1 4
1 4 3 2
3 2 4 1
4 1 2 3

ETSINTÄ



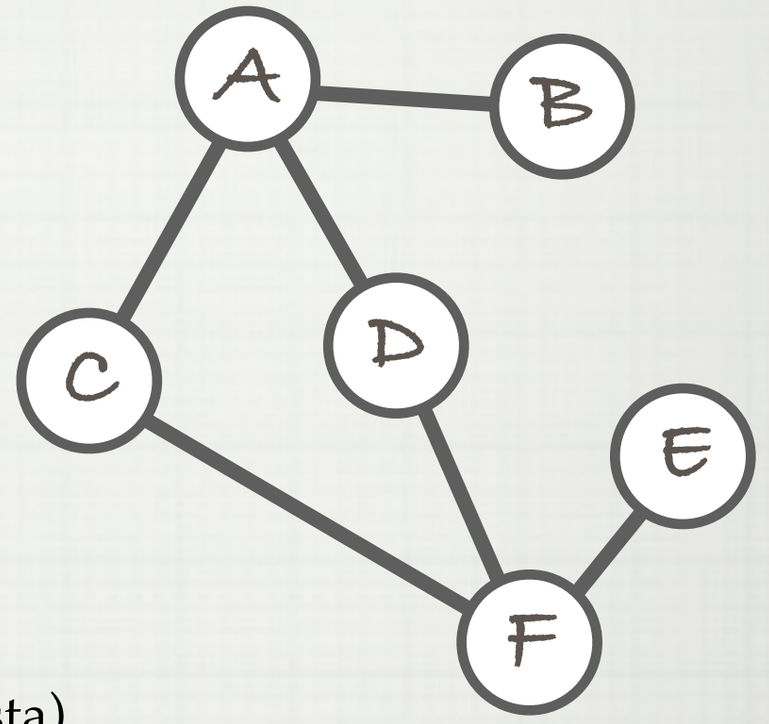
ETSINTÄ

LYHIN REITTI
SELVIÄÄ "PERUUTTAMALLA"
NUOLTEN SUUNTAAN

ETSINTÄ

ETSINTÄ(Alkusolmu)

```
Solmulista = [Alkusolmu]
Käsitellyt = [ ]
while Solmulista not empty
    Solmu = EKA(Solmulista)
    Solmulista = LOPUT(Solmulista)
    if Solmu not in Käsitellyt
        Käsitellyt = Käsitellyt + [Solmu]
        if MAALI(Solmu) return("ratkaisu", Solmu)
        Solmulista = LISÄÄ(NAAPURIT(Solmu),Solmulista)
    end if
end while
return("ei ratkaisua")
```



LISÄÄ(Naapurit, Solmulista)

LEVEYSSUUNTAINEN HAKU:

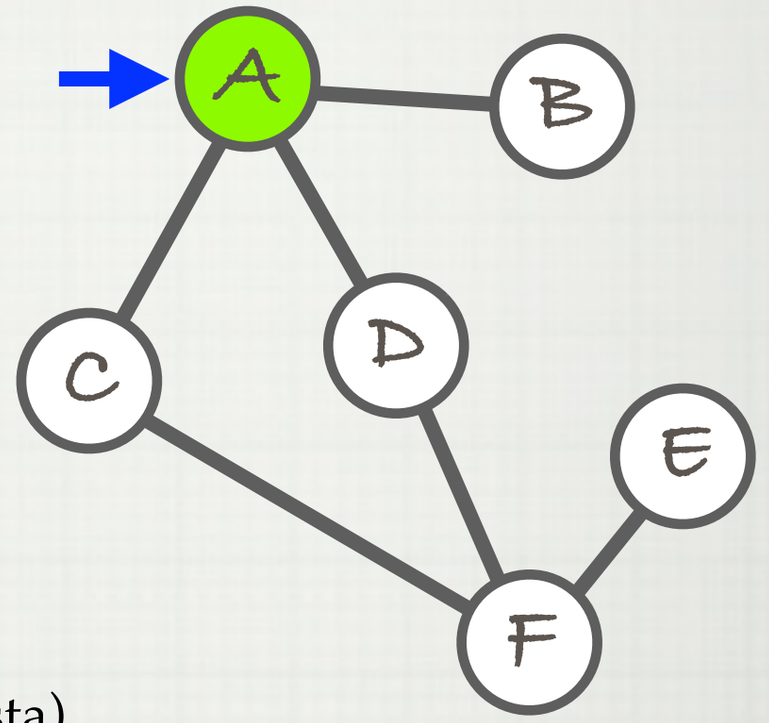
```
Uudet = Naapurilista – Käsitellyt – Solmulista
return(PERÄKKÄIN(Solmulista, Uudet))
# LISÄÄ([a,f],[d]) => [d,a,f]
```

ETSINTÄ

[A]

ETSINTÄ(Alkusolmu)

```
Solmulista = [Alkusolmu]
Käsitellyt = [ ]
while Solmulista not empty
    Solmu = EKA(Solmulista)
    Solmulista = LOPUT(Solmulista)
    if Solmu not in Käsitellyt
        Käsitellyt = Käsitellyt + [Solmu]
        if MAALI(Solmu) return("ratkaisu", Solmu)
        Solmulista = LISÄÄ(NAAPURIT(Solmu),Solmulista)
    end if
end while
return("ei ratkaisua")
```



LISÄÄ(Naapurit, Solmulista)

LEVEYSSUUNTAINEN HAKU:

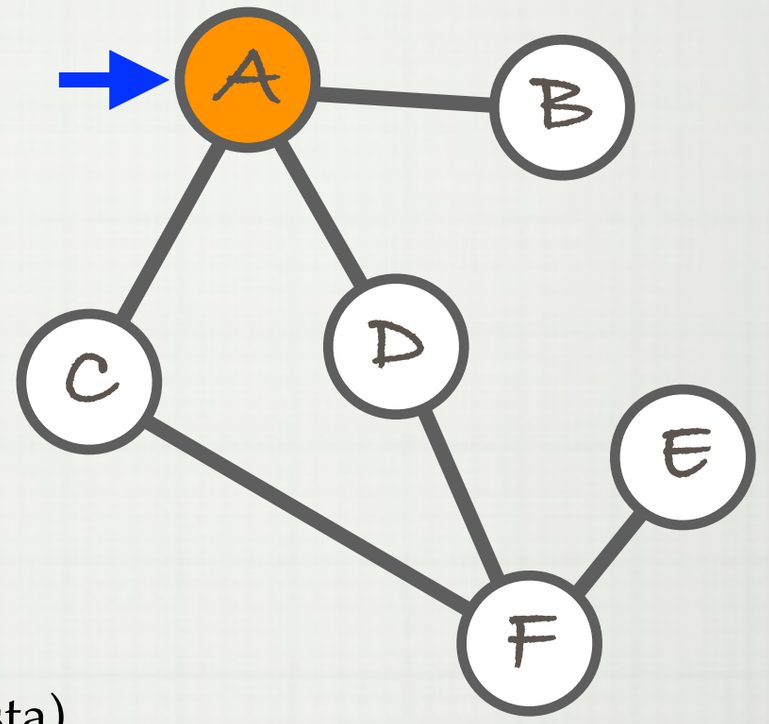
```
Uudet = Naapurilista – Käsitellyt – Solmulista
return(PERÄKKÄIN(Solmulista, Uudet))
# LISÄÄ([a,f],[d]) => [d,a,f]
```

ETSINTÄ

[]

ETSINTÄ(Alkusolmu)

```
Solmulista = [Alkusolmu]
Käsitellyt = [ ]
while Solmulista not empty
    Solmu = EKA(Solmulista)
    Solmulista = LOPUT(Solmulista)
    if Solmu not in Käsitellyt
        Käsitellyt = Käsitellyt + [Solmu]
        if MAALI(Solmu) return("ratkaisu", Solmu)
        Solmulista = LISÄÄ(NAAPURIT(Solmu),Solmulista)
    end if
end while
return("ei ratkaisua")
```



LISÄÄ(Naapurit, Solmulista)

LEVEYSSUUNTAINEN HAKU:

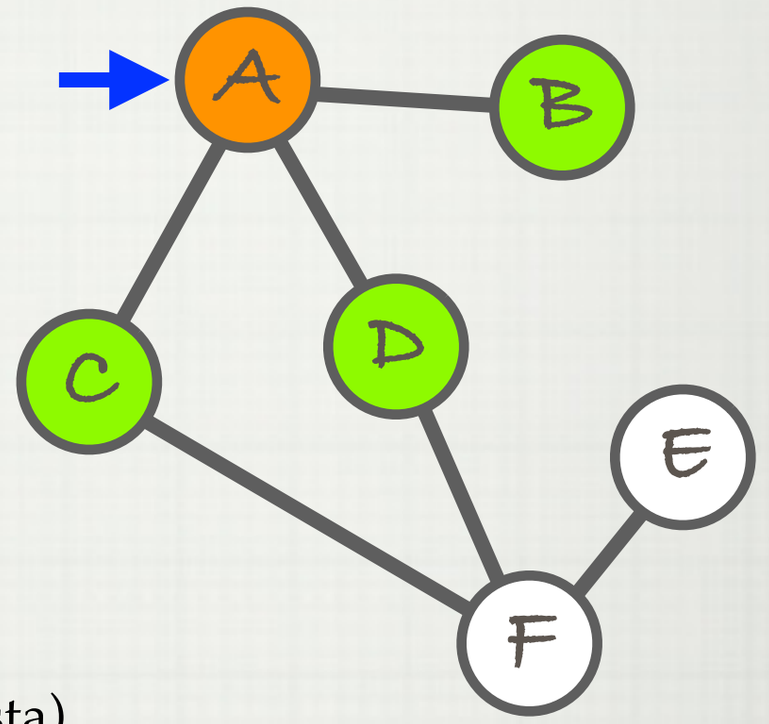
```
Uudet = Naapurilista – Käsitellyt – Solmulista
return(PERÄKKÄIN(Solmulista, Uudet))
# LISÄÄ([a,f],[d]) => [d,a,f]
```

ETSINTÄ

[B,C,D]

ETSINTÄ(Alkusolmu)

Solmulista = [Alkusolmu]
Käsitellyt = [ ]
while Solmulista not empty
    Solmu = EKA(Solmulista)
    Solmulista = LOPUT(Solmulista)
    if Solmu not in Käsitellyt
        Käsitellyt = Käsitellyt + [Solmu]
        if MAALI(Solmu) return(“ratkaisu”, Solmu)
        Solmulista = LISÄÄ(NAAPURIT(Solmu),Solmulista)
    end if
end while
return(“ei ratkaisua”)



LISÄÄ(Naapurit, Solmulista)

LEVEYSSUUNTAINEN HAKU:

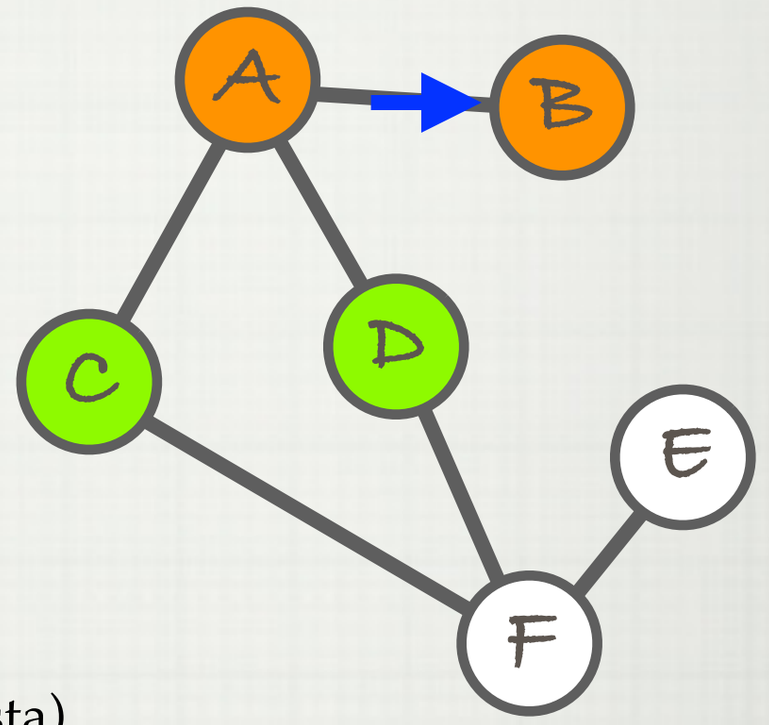
Uudet = Naapurilista – Käsitellyt – Solmulista
return(PERÄKKÄIN(Solmulista, Uudet))
# LISÄÄ([a,f],[d]) => [d,~~a~~,f]

ETSINTÄ

[C,D]

ETSINTÄ(Alkusolmu)

Solmulista = [Alkusolmu]
Käsitellyt = [ ]
while Solmulista not empty
    Solmu = EKA(Solmulista)
    Solmulista = LOPUT(Solmulista)
    if Solmu not in Käsitellyt
        Käsitellyt = Käsitellyt + [Solmu]
        if MAALI(Solmu) return(“ratkaisu”, Solmu)
        Solmulista = LISÄÄ(NAAPURIT(Solmu),Solmulista)
    end if
end while
return(“ei ratkaisua”)



LISÄÄ(Naapurit, Solmulista)

LEVEYSSUUNTAINEN HAKU:

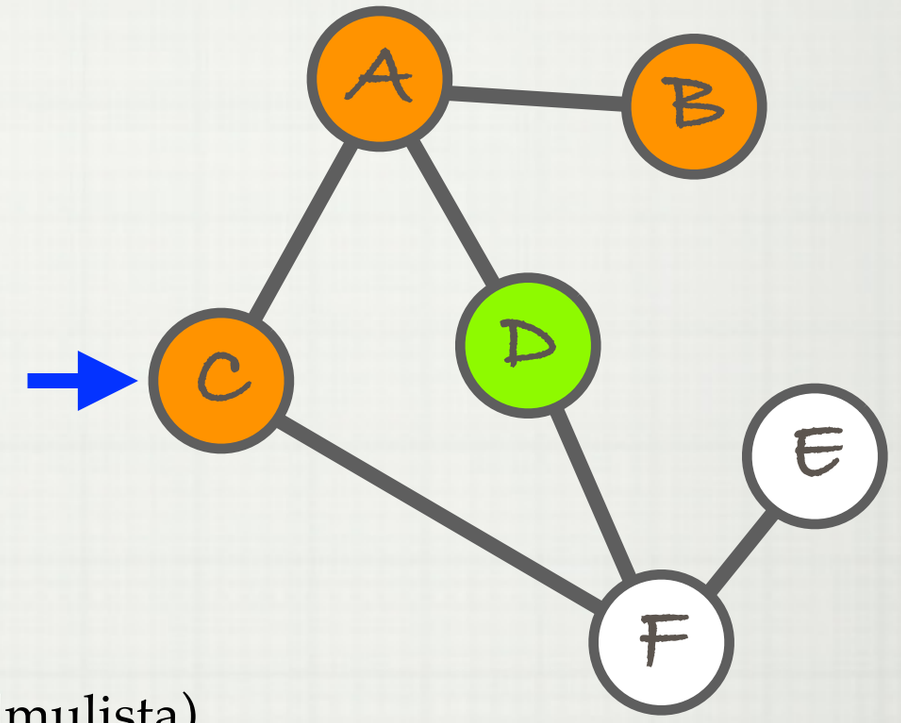
Uudet = Naapurilista – Käsitellyt – Solmulista
return(PERÄKKÄIN(Solmulista, Uudet))
# LISÄÄ([a,f],[d]) => [d,~~a~~,f]

ETSINTÄ

[D]

ETSINTÄ(Alkusolmu)

```
Solmulista = [Alkusolmu]
Käsitellyt = [ ]
while Solmulista not empty
    Solmu = EKA(Solmulista)
    Solmulista = LOPUT(Solmulista)
    if Solmu not in Käsitellyt
        Käsitellyt = Käsitellyt + [Solmu]
        if MAALI(Solmu) return("ratkaisu", Solmu)
        Solmulista = LISÄÄ(NAAPURIT(Solmu),Solmulista)
    end if
end while
return("ei ratkaisua")
```



LISÄÄ(Naapurit, Solmulista)

LEVEYSSUUNTAINEN HAKU:

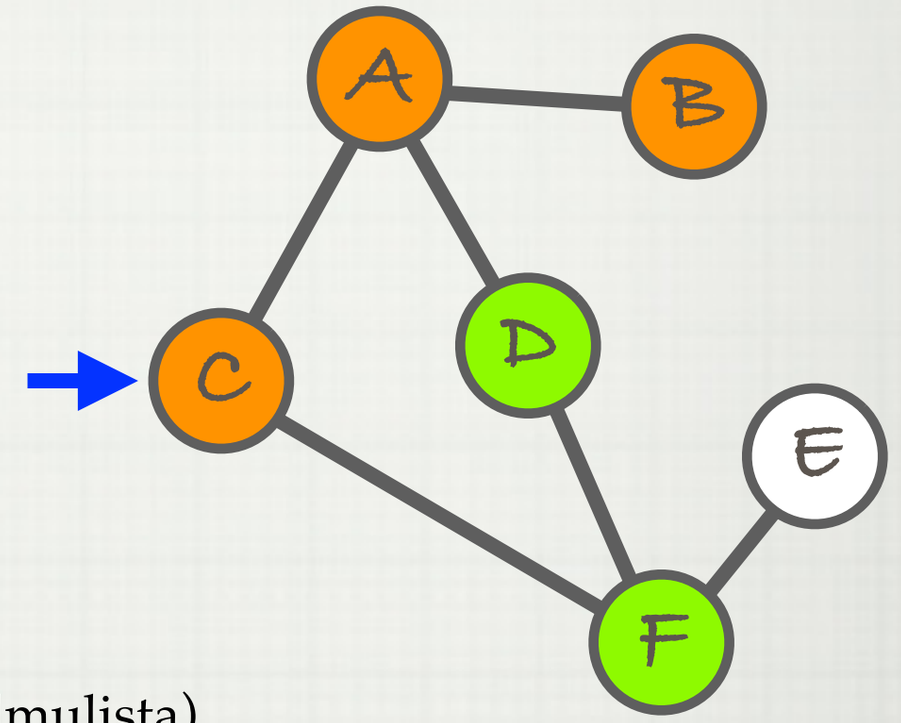
```
Uudet = Naapurilista – Käsitellyt – Solmulista
return(PERÄKKÄIN(Solmulista, Uudet))
# LISÄÄ([a,f],[d]) => [d,~~a~~,f]
```

ETSINTÄ

[D,F]

ETSINTÄ(Alkusolmu)

```
Solmulista = [Alkusolmu]
Käsitellyt = [ ]
while Solmulista not empty
    Solmu = EKA(Solmulista)
    Solmulista = LOPUT(Solmulista)
    if Solmu not in Käsitellyt
        Käsitellyt = Käsitellyt + [Solmu]
        if MAALI(Solmu) return("ratkaisu", Solmu)
        Solmulista = LISÄÄ(NAAPURIT(Solmu),Solmulista)
    end if
end while
return("ei ratkaisua")
```



LISÄÄ(Naapurit, Solmulista)

LEVEYSSUUNTAINEN HAKU:

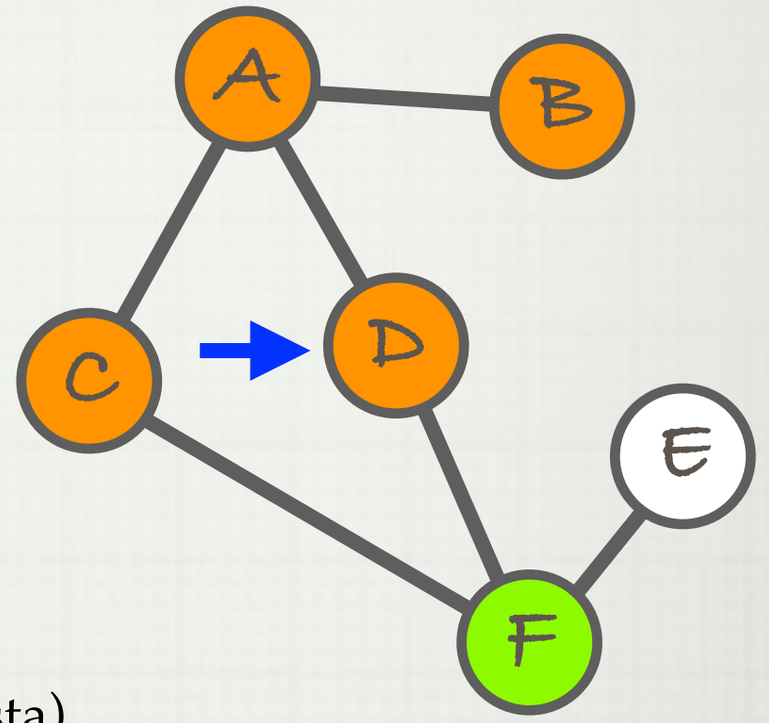
```
Uudet = Naapurilista – Käsitellyt – Solmulista
return(PERÄKKÄIN(Solmulista, Uudet))
# LISÄÄ([a,f],[d]) => [d,a,f]
```

2. ETSINTÄ JA PELIT

[F]

ETSINTÄ(Alkusolmu)

```
Solmulista = [Alkusolmu]
Käsitellyt = [ ]
while Solmulista not empty
    Solmu = EKA(Solmulista)
    Solmulista = LOPUT(Solmulista)
    if Solmu not in Käsitellyt
        Käsitellyt = Käsitellyt + [Solmu]
        if MAALI(Solmu) return("ratkaisu", Solmu)
        Solmulista = LISÄÄ(NAAPURIT(Solmu),Solmulista)
    end if
end while
return("ei ratkaisua")
```



LISÄÄ(Naapurit, Solmulista)

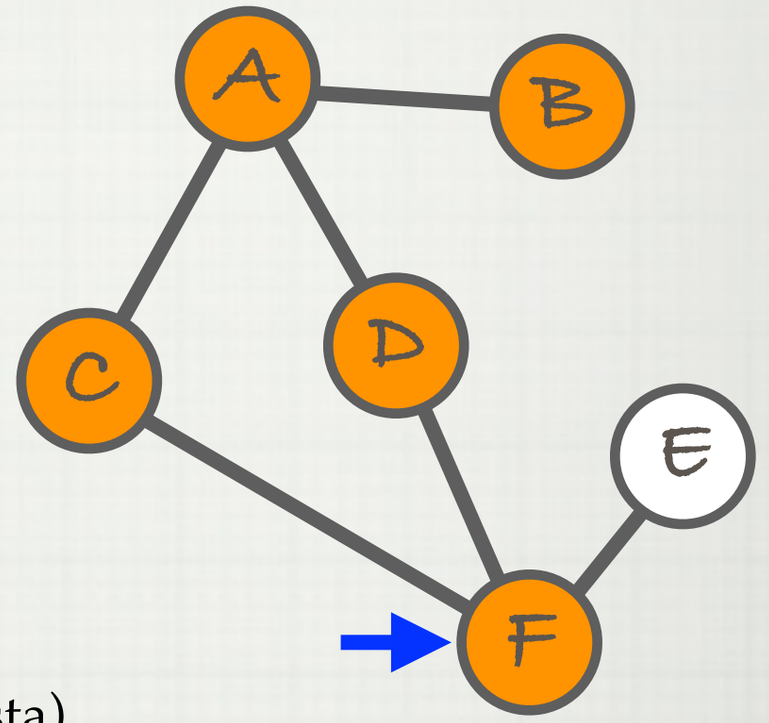
LEVEYSSUUNTAINEN HAKU:

```
Uudet = Naapurilista – Käsitellyt – Solmulista
return(PERÄKKÄIN(Solmulista, Uudet))
# LISÄÄ([a,f],[d]) => [d,a,f]
```

ETSINTÄ

ETSINTÄ(Alkusolmu)

```
Solmulista = [Alkusolmu]
Käsitellyt = [ ]
while Solmulista not empty
    Solmu = EKA(Solmulista)
    Solmulista = LOPUT(Solmulista)
    if Solmu not in Käsitellyt
        Käsitellyt = Käsitellyt + [Solmu]
        if MAALI(Solmu) return("ratkaisu", Solmu)
        Solmulista = LISÄÄ(NAAPURIT(Solmu),Solmulista)
    end if
end while
return("ei ratkaisua")
```



LISÄÄ(Naapurit, Solmulista)

LEVEYSSUUNTAINEN HAKU:

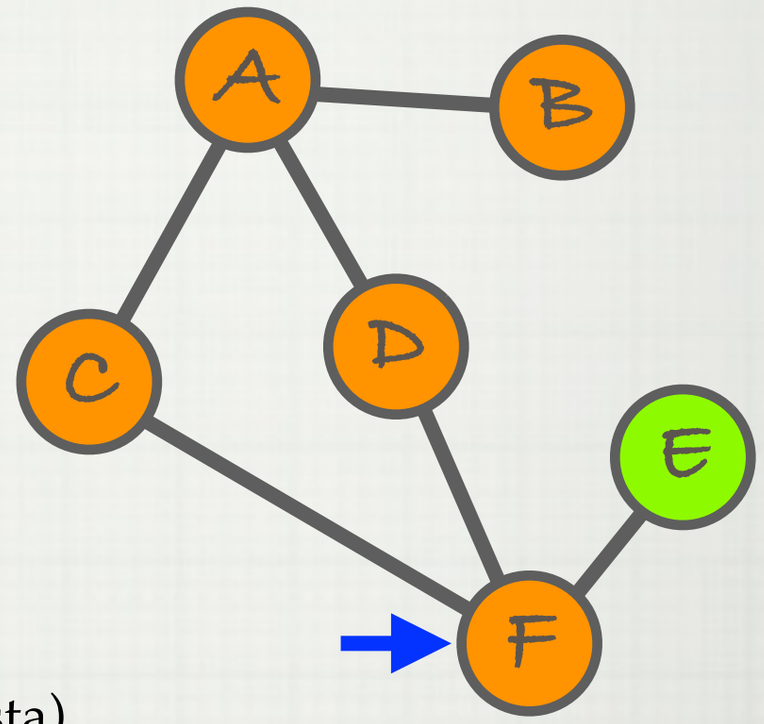
```
Uudet = Naapurilista – Käsitellyt – Solmulista
return(PERÄKKÄIN(Solmulista, Uudet))
# LISÄÄ([a,f],[d]) => [d,a,f]
```

ETSINTÄ

[E]

ETSINTÄ(Alkusolmu)

```
Solmulista = [Alkusolmu]
Käsitellyt = [ ]
while Solmulista not empty
    Solmu = EKA(Solmulista)
    Solmulista = LOPUT(Solmulista)
    if Solmu not in Käsitellyt
        Käsitellyt = Käsitellyt + [Solmu]
        if MAALI(Solmu) return("ratkaisu", Solmu)
        Solmulista = LISÄÄ(NAAPURIT(Solmu),Solmulista)
    end if
end while
return("ei ratkaisua")
```



LISÄÄ(Naapurit, Solmulista)

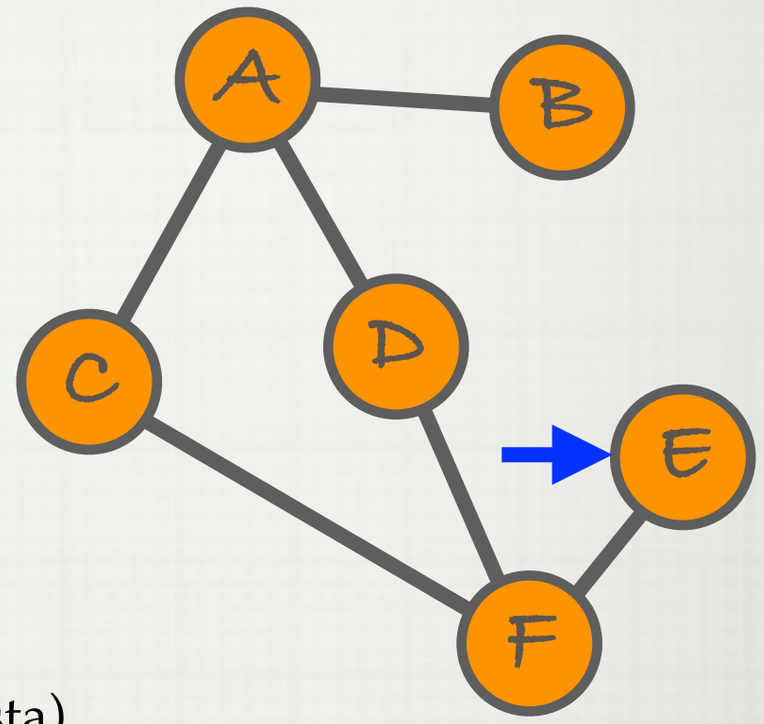
LEVEYSSUUNTAINEN HAKU:

```
Uudet = Naapurilista – Käsitellyt – Solmulista
return(PERÄKKÄIN(Solmulista, Uudet))
# LISÄÄ([a,f],[d]) => [d,a,f]
```

ETSINTÄ

ETSINTÄ(Alkusolmu)

```
Solmulista = [Alkusolmu]
Käsitellyt = [ ]
while Solmulista not empty
    Solmu = EKA(Solmulista)
    Solmulista = LOPUT(Solmulista)
    if Solmu not in Käsitellyt
        Käsitellyt = Käsitellyt + [Solmu]
        if MAALI(Solmu) return("ratkaisu", Solmu)
        Solmulista = LISÄÄ(NAAPURIT(Solmu),Solmulista)
    end if
end while
return("ei ratkaisua")
```



LISÄÄ(Naapurit, Solmulista)

LEVEYSSUUNTAINEN HAKU:

```
Uudet = Naapurilista – Käsitellyt – Solmulista
return(PERÄKKÄIN(Solmulista, Uudet))
# LISÄÄ([a,f],[d]) => [d,a,f]
```

ETSINTÄ

ETSINTÄ(Alkusolmu)

```
Solmulista = [Alkusolmu]
Käsitellyt = [ ]
while Solmulista not empty
    Solmu = EKA(Solmulista)
    Solmulista = LOPUT(Solmulista)
    if Solmu not in Käsitellyt
        Käsitellyt = Käsitellyt + [Solmu]
        if MAALI(Solmu) return(“ratkaisu”, Solmu)
        Solmulista = LISÄÄ(NAAPURIT(Solmu),Solmulista)
    end if
end while
return(“ei ratkaisua”)
```

JONO (“FIRST-IN-FIRST-OUT”)

LISÄÄ(Naapurit, Solmulista) LEVEYSSUUNTAINEN HAKU:

```
Uudet = Naapurilista – Käsitellyt – Solmulista
return(PERÄKKÄIN(Solmulista, Uudet))
# LISÄÄ([a,f],[d]) => [d,a,f]
```

ETSINTÄ

ETSINTÄ(Alkusolmu)

```
Solmulista = [Alkusolmu]
Käsitellyt = [ ]
while Solmulista not empty
    Solmu = EKA(Solmulista)
    Solmulista = LOPUT(Solmulista)
    if Solmu not in Käsitellyt
        Käsitellyt = Käsitellyt + [Solmu]
        if MAALI(Solmu) return(“ratkaisu”, Solmu)
        Solmulista = LISÄÄ(NAAPURIT(Solmu),Solmulista)
    end if
end while
return(“ei ratkaisua”)
```

PINO (“LAST-IN-FIRST-OUT”)

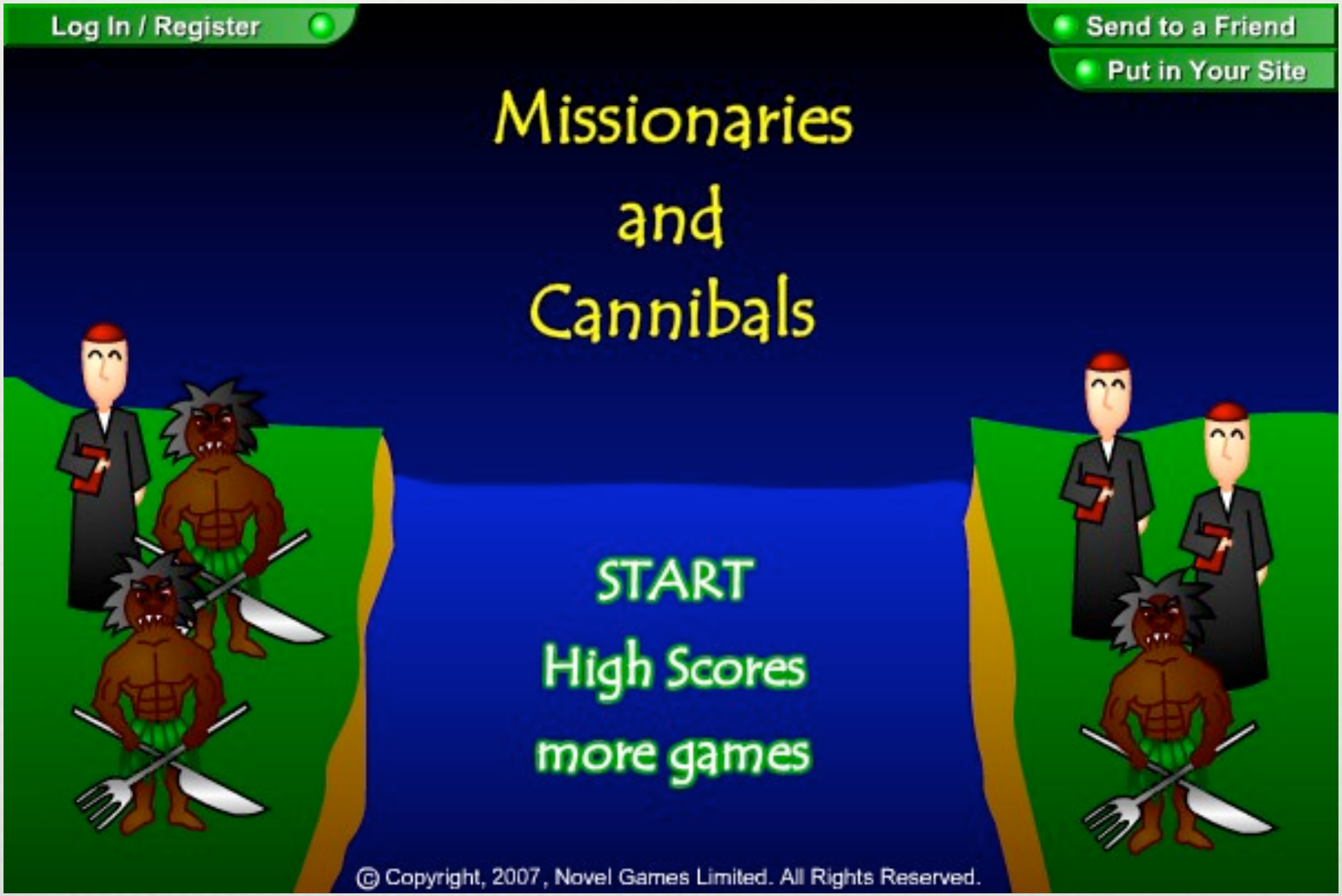
LISÄÄ(Naapurit, Solmulista)

SYVYYSSUUNTAINEN HAKU:

```
Uudet = Naapurilista – Käsitellyt
return(PERÄKKÄIN(Uudet, Solmulista))
# LISÄÄ([a,f],[d]) => [a,f,d]
```

ETSINTÄ ONGELMANRATKAISUNA

ETSINTÄ ONGELMANRATKAISUNA



ETSINTÄ ONGELMANRATKAISUNA

- KOLME KANNIBAALIA JA KOLME LÄHETYSSAARNAAJAA HALUAA YLITTÄÄ JOEN VENEELLÄ, JOHON MAHTUU VAIN KAKSI HENKILÖÄ.

- JOS JOMMALLA KUMMALLA RANNALLA ON ENEMMÄN KANNIBAALEJA KUIN LÄHETYSSAARNAAJIA (MUTTA KUITENKIN VÄHINTÄÄN YKSI LÄHETYSSAARNAAJA), KANNIBAALIT SYÖVÄT HEIDÄT.

- MITEN JOKI SAADAAN YLITETTYÄ ILMAN, ETTÄ KETÄÄN SYÖDÄÄN?

- VOIT KOKEILLA KLIKKAAMALLA TÄSTÄ.

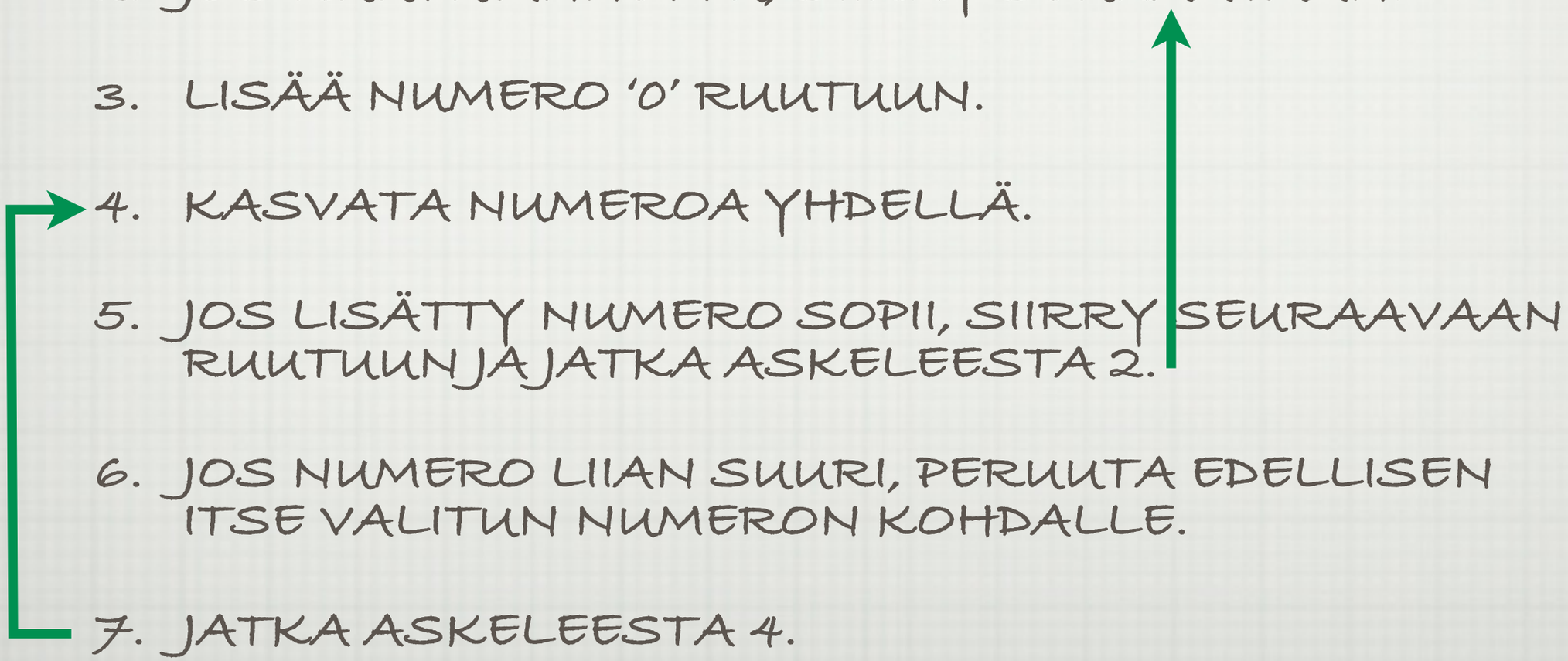
SUDOKU

	3		
	2		1
	1	2	3

SUDOKU

- YKSINKERTAINEN SUDOKU-ALGORITMI:

1. ALOITA VASEMMASTA YLÄKULMASTA.
2. JOS RUUTU ANNETTU, SIIRRY SEURAAVAAN.
3. LISÄÄ NUMERO '0' RUUTUUN.
4. KASVATA NUMEROA YHDELLÄ.
5. JOS LISÄTTY NUMERO SOPII, SIIRRY SEURAAVAAN RUUTUUN JA JATKA ASKELEESTA 2.
6. JOS NUMERO LIIAN SUURI, PERUUTA EDELLISEN ITSE VALITUN NUMERON KOHDALLE.
7. JATKA ASKELEESTA 4.



SUDOKU

	3		
	2		1
	1	2	3

SUDOKU

1	3		
	2		1
	1	2	3

SUDOKU

1	3	1	
	2		1
	1	2	3

SUDOKU

1	3	2	
	2		1
	1	2	3

SUDOKU

1	3	3	
	2		1
	1	2	3

SUDOKU

1	3	4	
	2		1
	1	2	3

SUDOKU

1	3	4	2
	2		1
	1	2	3

SUDOKU

1	3	4	2
2	4	1	?
	2		1
	1	2	3

SUDOKU

1	3	4	2
2	4	3	?
	2		1
	1	2	3

SUDOKU

1	3	4	2
2	4	?	
	2		1
	1	2	3

SUDOKU

1	3	4	2
4			
	2		1
	1	2	3

SUDOKU

2	3		
	2		1
	1	2	3

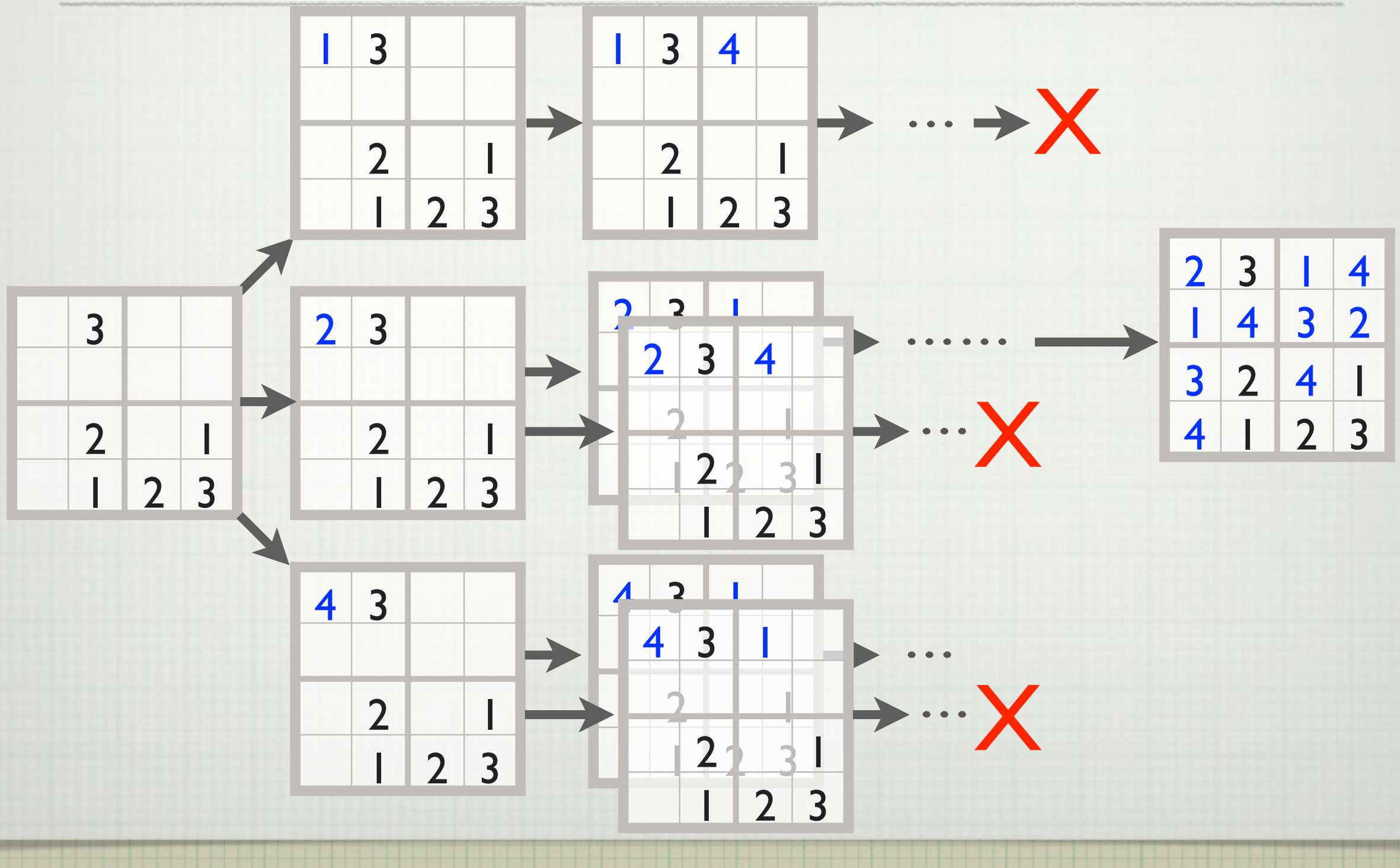
SUDOKU

2	3	1	
	2		1
	1	2	3

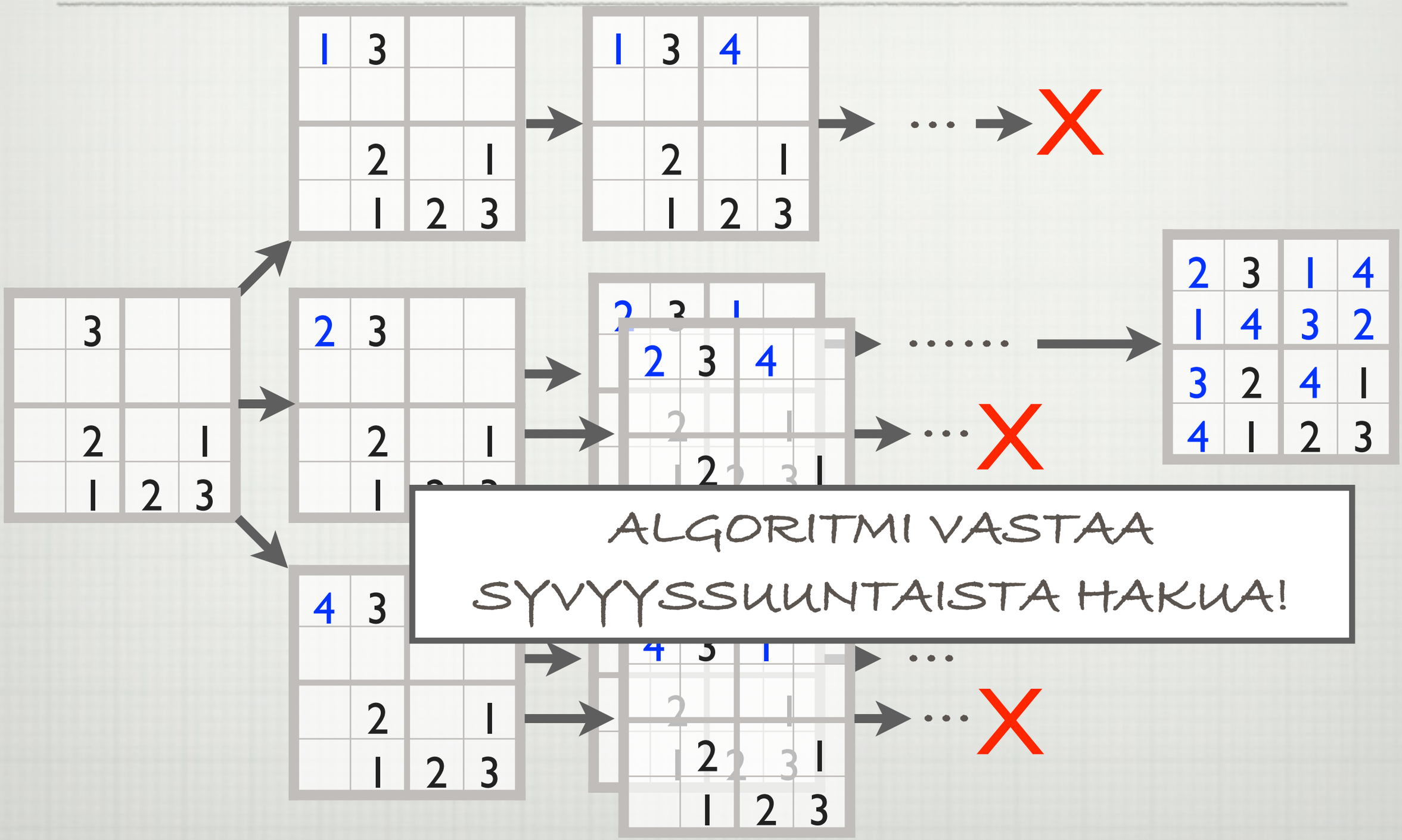
SUDOKU

2	3	1	4
1	4	3	2
3	2	4	1
4	1	2	3

SUDOKU


1 3
2
1
1 2 3
1 3 4
2
1
1 2 3
X
3
2
1
1 2 3
2 3
2
1
1 2 3
2 3 4
2
1
1 2 3
X
4 3
2
1
1 2 3
4 3 1
2
1
1 2 3
X
2 3 1 4
1 4 3 2
3 2 4 1
4 1 2 3

SUDOKU



PARAS-ENSIN–HAKU

ETSINTÄ(Alkusolmu)

```
Solmulista = [Alkusolmu]
Käsitellyt = [ ]
while Solmulista not empty
    Solmu = EKA(Solmulista)
    Solmulista = LOPUT(Solmulista)
    if Solmu not in Käsitellyt
        Käsitellyt = Käsitellyt + [Solmu]
        if MAALI(Solmu) return(“ratkaisu”, Solmu)
        Solmulista = LISÄÄ(NAAPURIT(Solmu),Solmulista)
    end if
end while
return(“ei ratkaisua”)
```

LISÄÄ(Solmulista1, Solmulista2)

PARAS-ENSIN -HAKU:

```
return(JÄRJESTÄ(Solmulista1, Solmulista2))
# [(a,5),(b,3),(c,1)], [(a,2),(c,3),(f,5)] => [(c,1),(a,2),(b,3),(f,5)]
```

HEURISTIIKAT

- KUSTANNUSARVIO: $f(N)$
  - ARVIO LÄHTÖSOLMUSTA SOLMUN N KAUTTA MAALISOLMUUN KULKEVAN POLUN KUSTANNUKSESTA

- "HEURISTIIKKA": $h(N)$
  - ARVIO KUSTANNUKSESTA SOLMUSTA N MAALISOLMUUN

- POLKUKUSTANNUS: $g(N)$
  - KUSTANNUS ALKUSOLMUSTA SOLMUUN N (RIIPPUU KULJETUSTA REITISTÄ)

$$f(N) = g(N) + h(N)$$

A*

ETSINTÄ(Alkusolmu)

```
Solmulista = [Alkusolmu]
Käsitellyt = [ ]
while Solmulista not empty
    Solmu = EKA(Solmulista)
    Solmulista = LOPUT(Solmulista)
    if Solmu not in Käsitellyt
        Käsitellyt = Käsitellyt + [Solmu]
        if MAALI(Solmu) return("ratkaisu", Solmu)
        Solmulista = LISÄÄ(NAAPURIT(Solmu),Solmulista)
    end if
end while
return("ei ratkaisua")
```

LISÄÄ(Solmulista1, Solmulista2)

```
return(JÄRJESTÄ(Solmulista1, Solmulista2))
# [(a,5),(b,3),(c,1)], [(a,2),(c,3),(f,5)] => [(c,1),(a,2),(b,3),(f,5)]
```

PARAS-ENSIN -HAKU:

A*-HAKU: $f(N) = g(N) + h(N)$

A*

- OLETUS: HEURISTIIKKA $h(N)$ ANTAA AINA ENINTÄÄN YHTÄ SUUREN ARVON KUIN TODELLINEN KUSTANNUS SOLMUSTA N MAALIIN.

- TÄLLÖIN A* TUOTTAA AINA OPTIMAALISEN RATKAISUN

- TODISTUKSEN IDEA:
  JONON EKAKSI EI VOI PÄÄSTÄ MAALISOLMU, JONKA POLKUKUSTANNUS ON SUUREMPI KUIN OPTIMAALISEN REITIN KUSTANNUS.

- JOS HEURISTIIKKA "HYVÄ", SUURIMMASSA OSASSA HUONOJA SOLMUJA EI KÄYDÄ OLLENKAAN.

REITTIOPAS


Taskuversio • På svenska • In English • Slangi • По-русски
Palaute • Ohjeet • FAQ
HSL HRT
Reittiopas
Reittiopas classic • Reittiopas API
HSL
Reittiopas
Omat lähdöt • Aikataulut • Linjaopas • Pyöräily ja kävely
Perushaku
Tarkennettu haku
Mistä
Kartta Tallenna Hakemisto
Mihin
Kartta Tallenna Hakemisto
Kello 22 : 29
Lähtöaika
Perillä
Pvm
Ma Ti Ke To Pe La Su
Syyskuu 2011
Lokakuu
Hae
Omat reitit
Ei omia reittejä (ohje omien reittien tallentamiseen).
Omat paikat
Ei omia paikkoja (ohje omien paikkojen tallentamiseen).
Poikkeusinfo
8.9.2011 17:35 - Sisäiset ja seutuliikenne myöhästyvät. Syy: ruuhka. Paikka: Mannerheimintie Arvioitu kesto: 17:31 - 24:00.
8.9.2011 17:32 - Sisäiset ja seutulinjat myöhästyvät. Syy: ruuhka. Paikka: Mannerheimintie. Arvioitu kesto: 17:23 - 24:00.
HSL /Poikkeusinfo → Mobiililaitteille →
Liikennetiedotteet
8.9.2011 Linjalla 11 lisävuoroja Kissojen yöhön 9.9.
8.9.2011 Martinkyläntien pysäkkien poikkeusjärjestelyt jatkuvat
7.9.2011 Bussille 14 reittimuutos Eirassa 12.9.
7.9.2011 Bussit ajavat poikkeusreittiä Keravan keskustassa 9. - 12.9.

REITTIOPAS

- TILA: (PYSÄKKI, KULUNUT AIKA)
- KUSTANNUSARVIO: (MATKA-AIKA (MIN))
- SIIRTYMÄT: (UUSI PYSÄKKI, VÄLIMATKA (MIN))
- TEHTÄVÄ: ETSI NOPEIN REITTI PYSÄKILTÄ A PYSÄKILLE B (EI KÄVELYÄ)
- MENETELMÄ: A*-HAKU