

JOHDATUS TEKÖÄLYYN

TEEMU ROOS

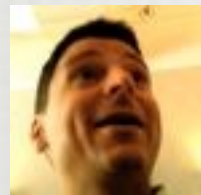


HELSINGIN YLIOPISTO

KURSSIN PERUSTIEDOT

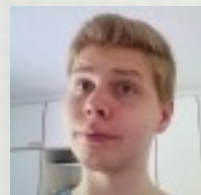
- * VALINNAINEN AINEOPINTOTASOINEN KURSSI, 5 OP
- * PERIODI 3: 18.1.2016-6.3.2016 (7 VIIKKOA+KOE)
- * LUENNOT (CK112):
MA 14-16, TI 14-16
- * LASKUHARJOITUKSET:
RYHMÄ 1: TI 12-14 (TEEMU)
RYHMÄ 2: TO 14-16 (ELIAS)
RYHMÄ 3: PE 12-14 (ELIAS)
RYHMÄ 4?
- * KURSSIKOE TO 10.3. KLO 16-18:30

TIIMI



TEEMU.ROOS@CS.HELSEINKI.FI

IRC: teemuroos HUONE: A322



ELIASJÄÄSAARI

IRC:

IRC: #johtek

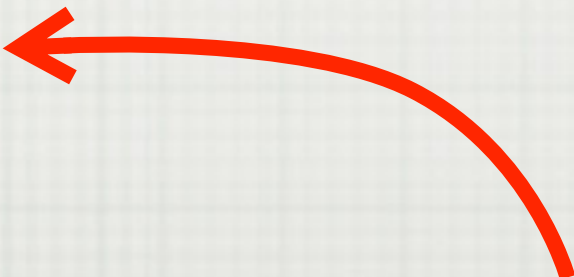
ESITIETOVAATIMUKSET

- * TIETORAKENTEET-KURSSI
- * JOHDATUS YLIOPISTOMATEMATIIKKAAN -KURSSI
- * TODENNÄKÖISYYSLASKENNAN KURSSISTA ON 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 20 PISTETTÄ
- * KURSSIKOE MAX 40 PISTETTÄ
- * HYVÄKSYMISRAJA N. 30 PISTETTÄ

VIELÄ PARI JUTTUA

- * ERILAINEN KUIN TYYPILLINEN AI-KURSSI:
 - VARHAISEMMASSA VAIHEESSA OPINTOJA
 - VÄHEMMÄN MATEMATIIKKA (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. KONEOPPIMINEN JA PÄÄTTELY EPÄVARMUUDEN
VALLITESSA

"MODERN AI"

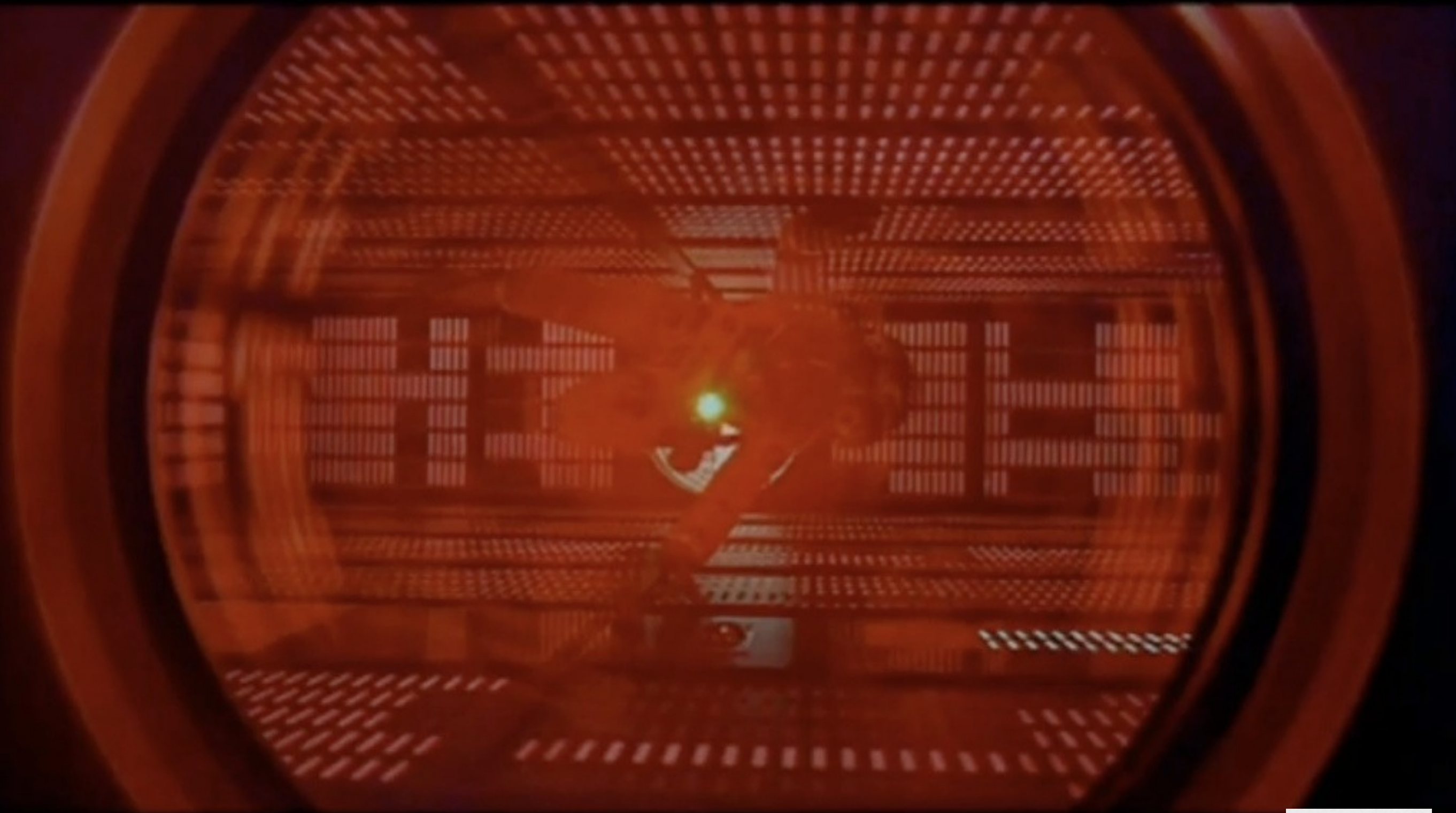
5. LUONNOLLISEN KIELEN KÄSITTELY

6. ROBOTIIKKA

KESKUSTELUA

* TEKOÄLY KULTTUURISSA

- * _____ R2-D2 _____
- * _____ GLADOS (PORTAL) _____
- * _____ GOOGLE AUTO _____
- * _____ SKYNET _____
- * _____ DATA _____





KESKUSTELUA

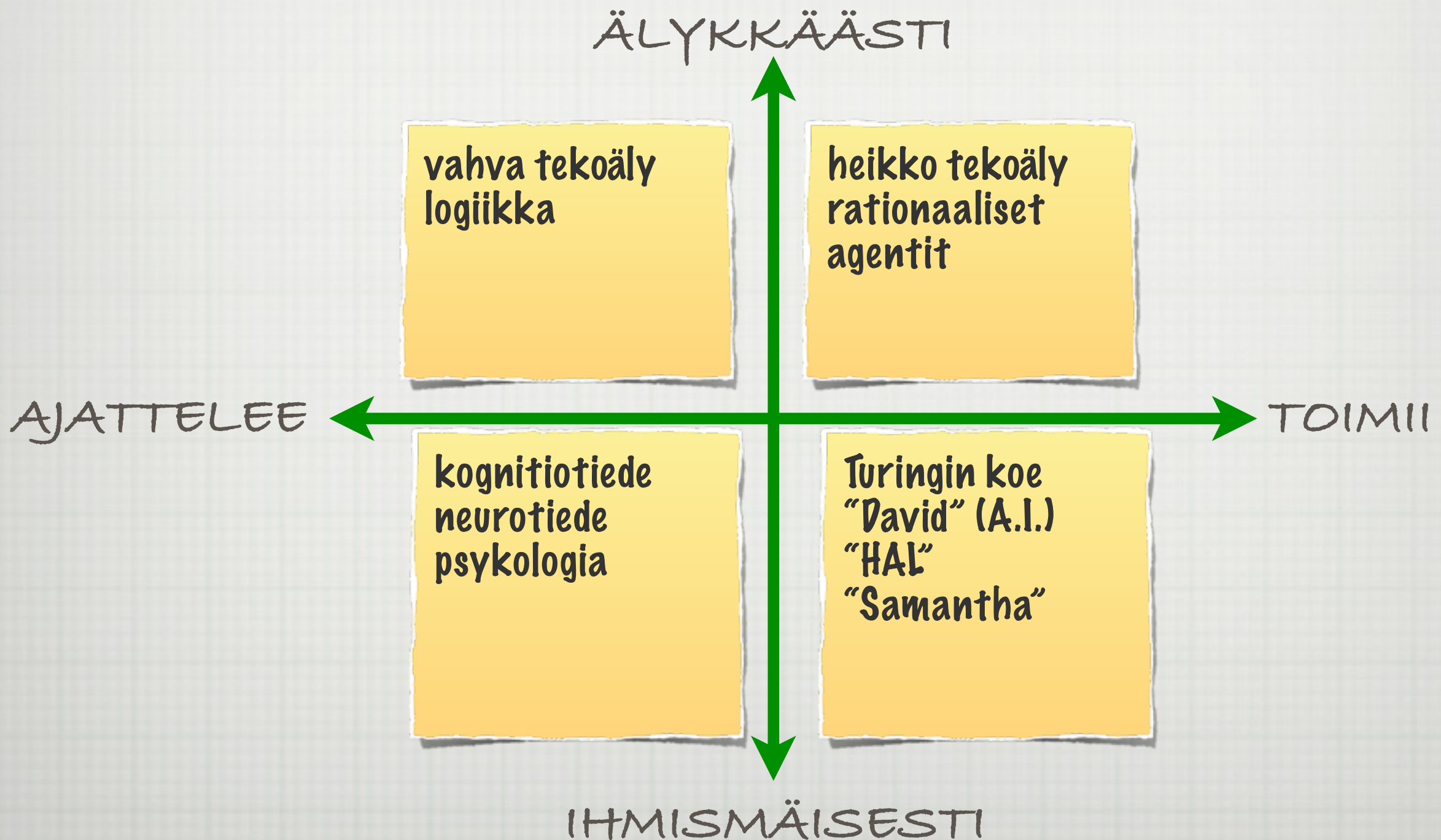
✱ MITÄ KAIKKEA SAMANTHA^(*) OSAA?

*) TAI MUU ELOKUVAN TEKOÄLY

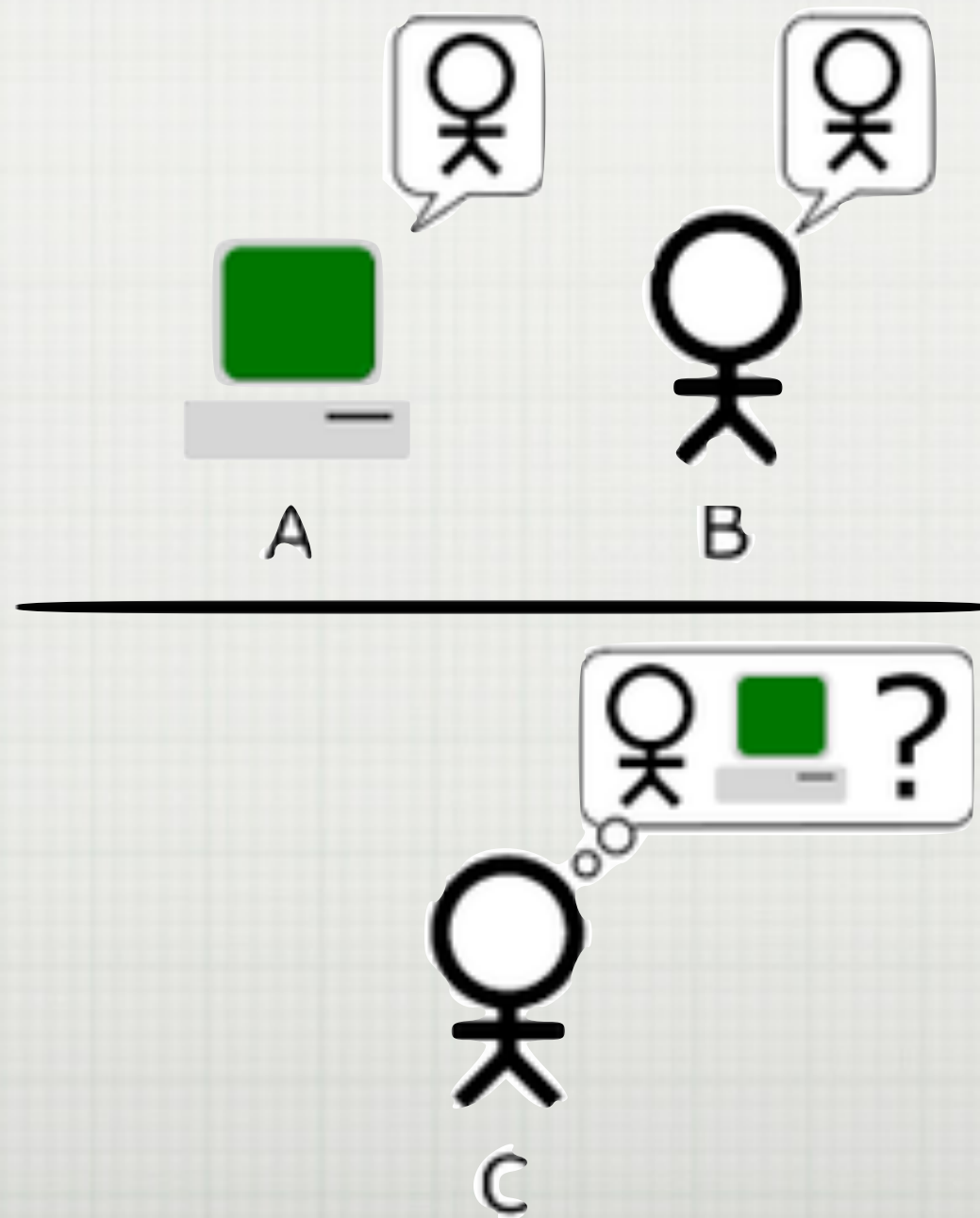
✱ MITÄ NÄISTÄ EI OSATA VIELÄ TOTEUTTAA?

✱ ONKO SAMANTHALLA TIETOISUUS?

TEKOÄLYN FILOSOFIAA



TOIMII IHMISMÄISESTI: TURINGIN TESTI





Are you good?

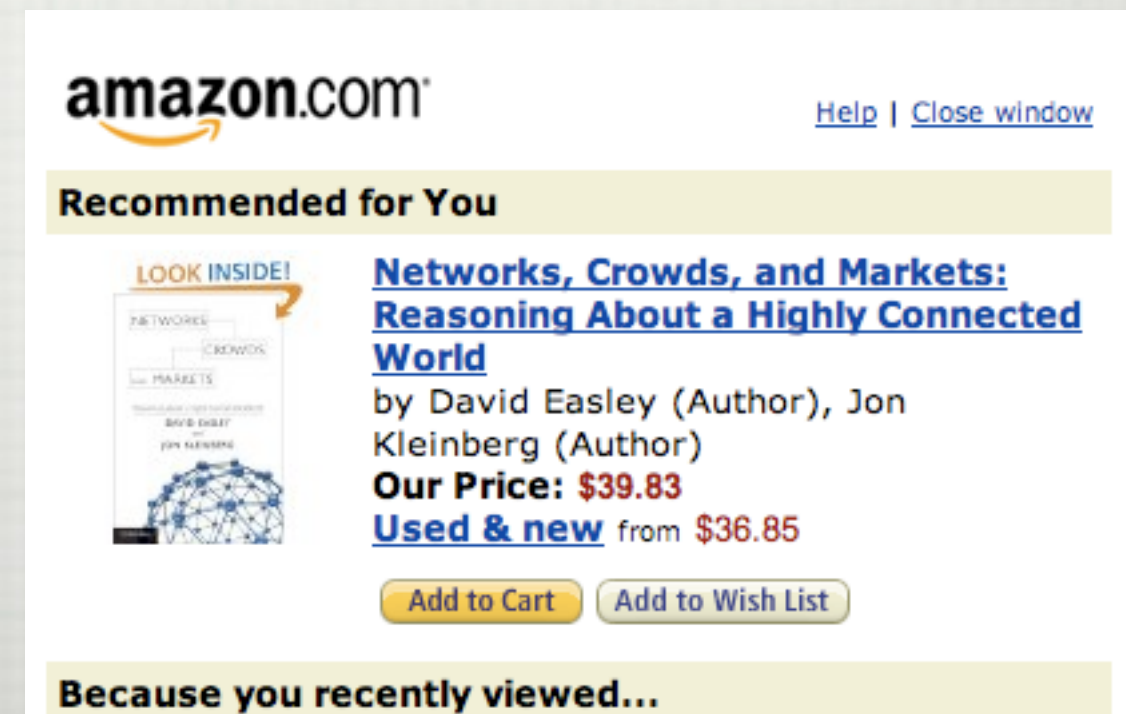
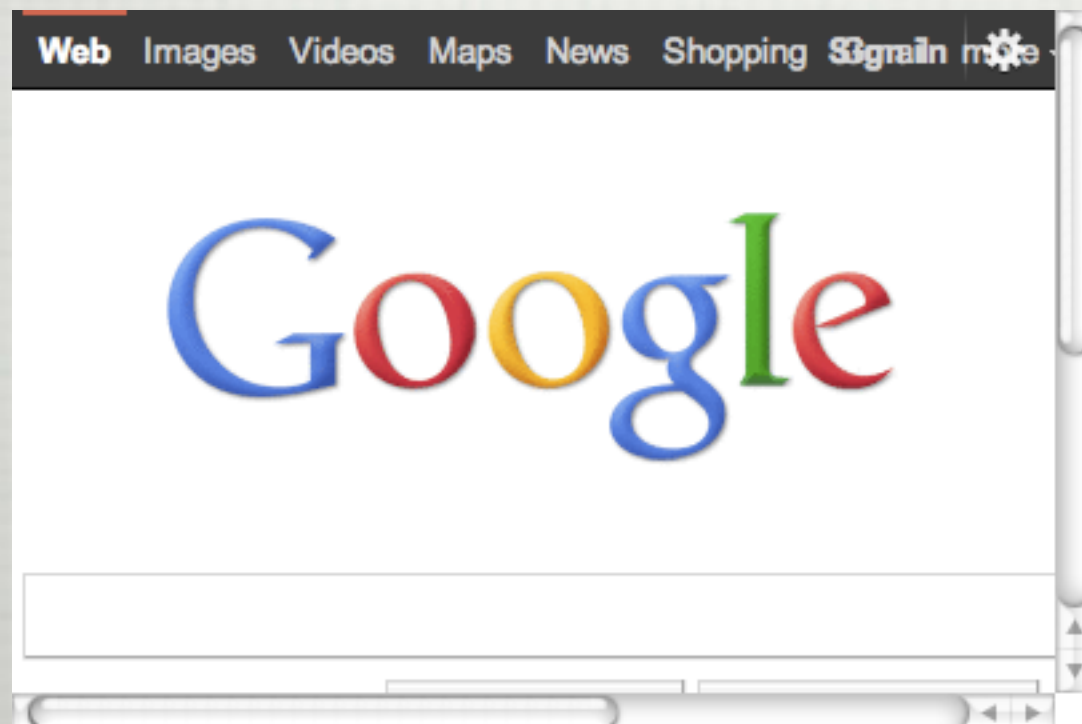
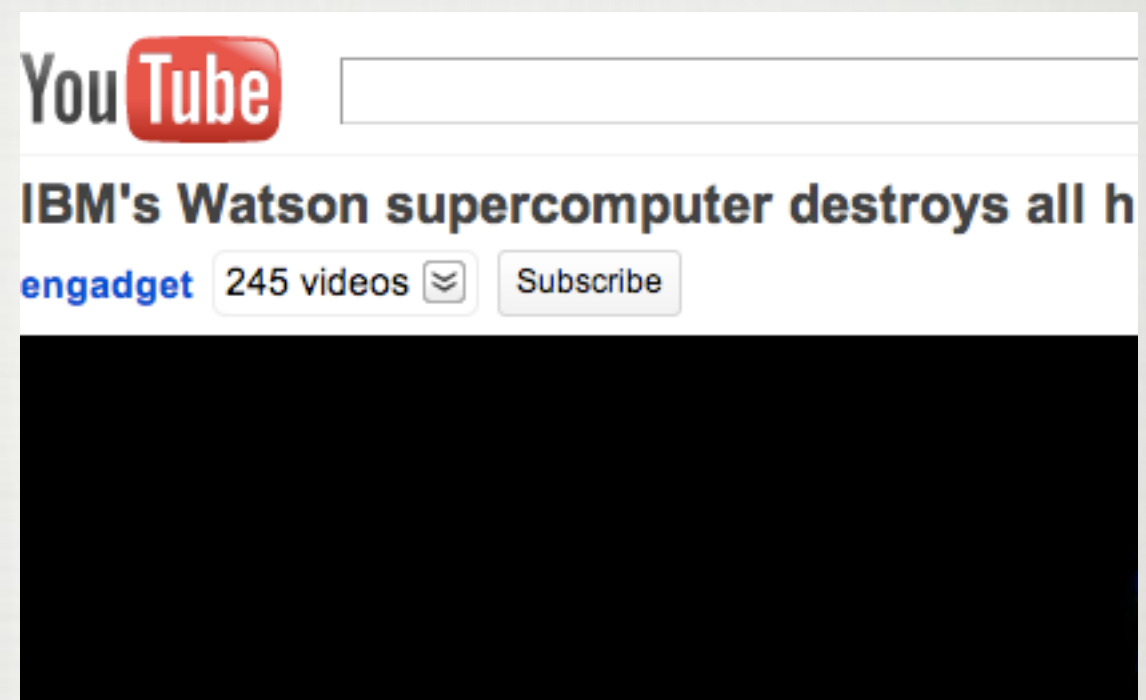
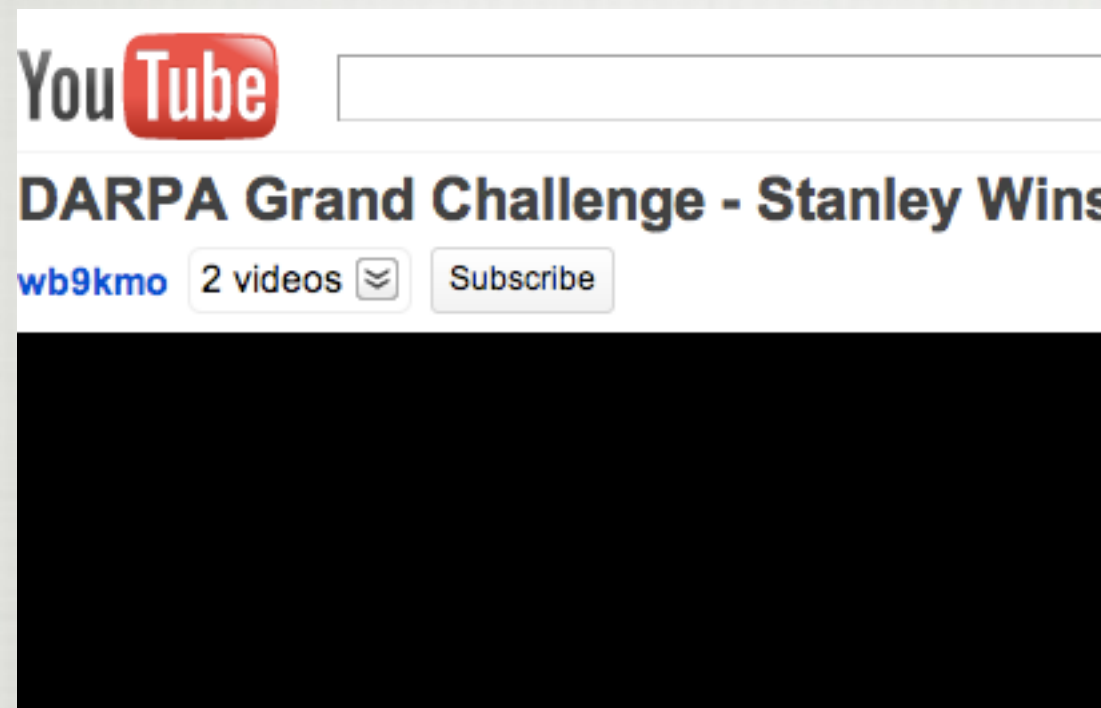
Artificial Intelligence
A Modern Approach

KIINALAINEN HUONE

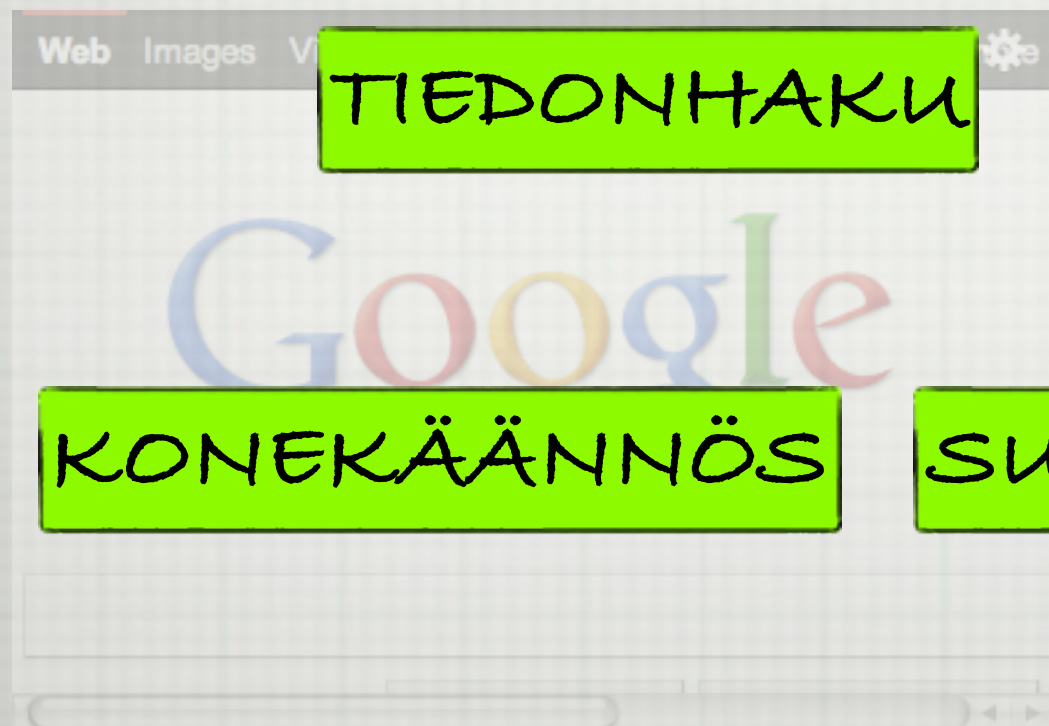
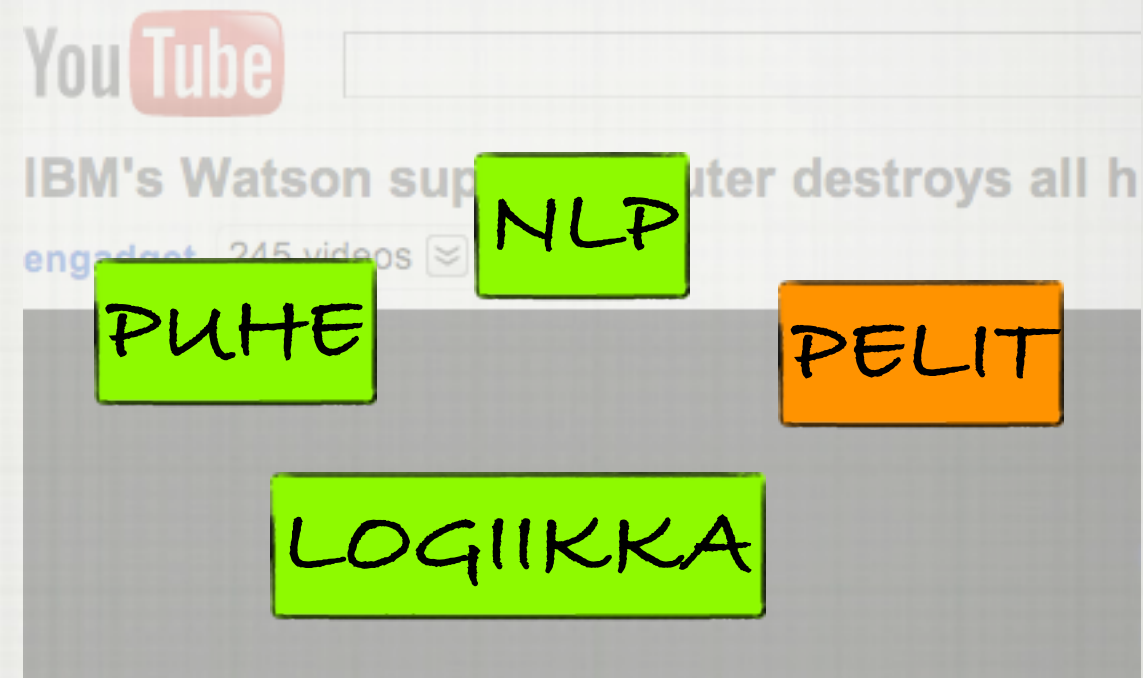
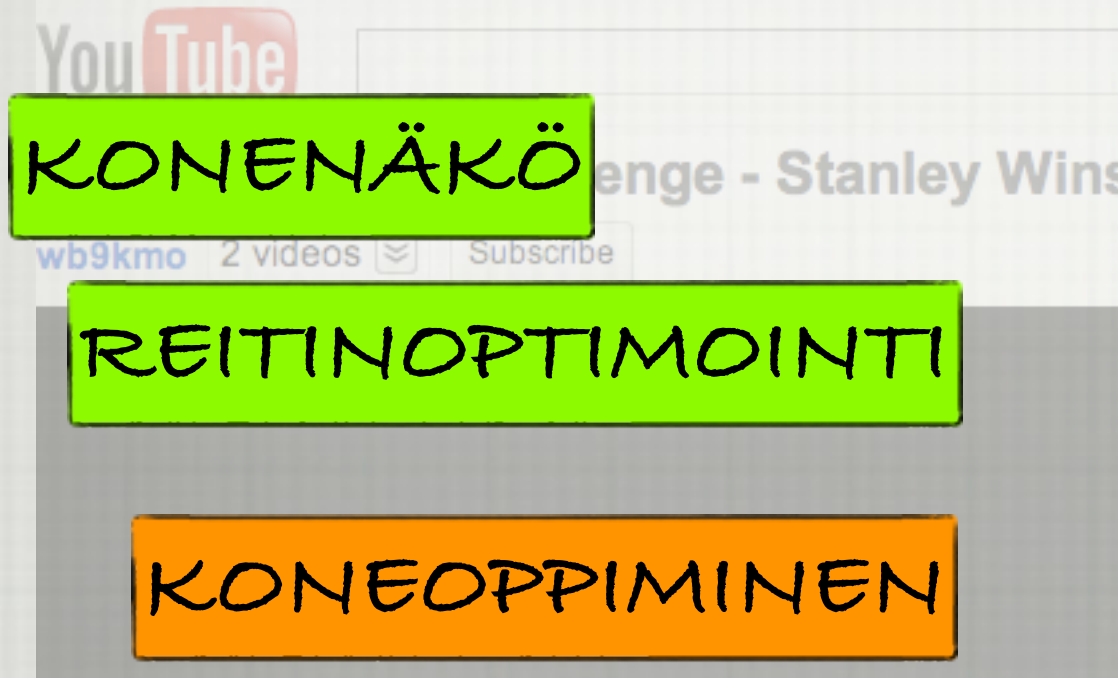
- * VOIKO TOIMIA
ÄLYKKÄÄSTI ILMAN
ETTÄ AJATTELEE?
- * TIETOISUUS?



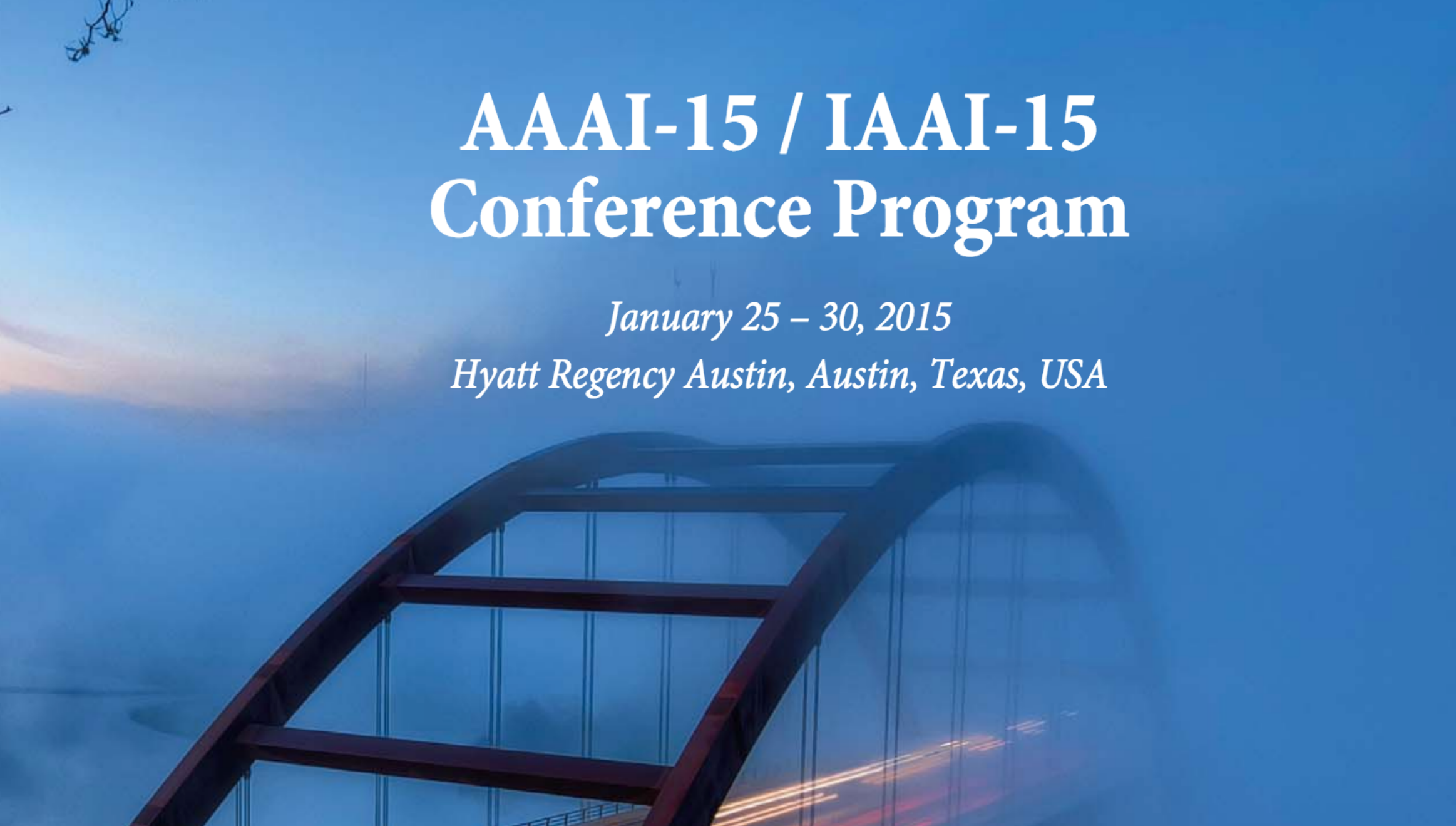
MITÄ TEKOÄLY OIKEASTI ON?



MITÄ TEKOÄLY OIKEASTI ON?



MITÄ TEKÖÄLY OIKEASTI ON?

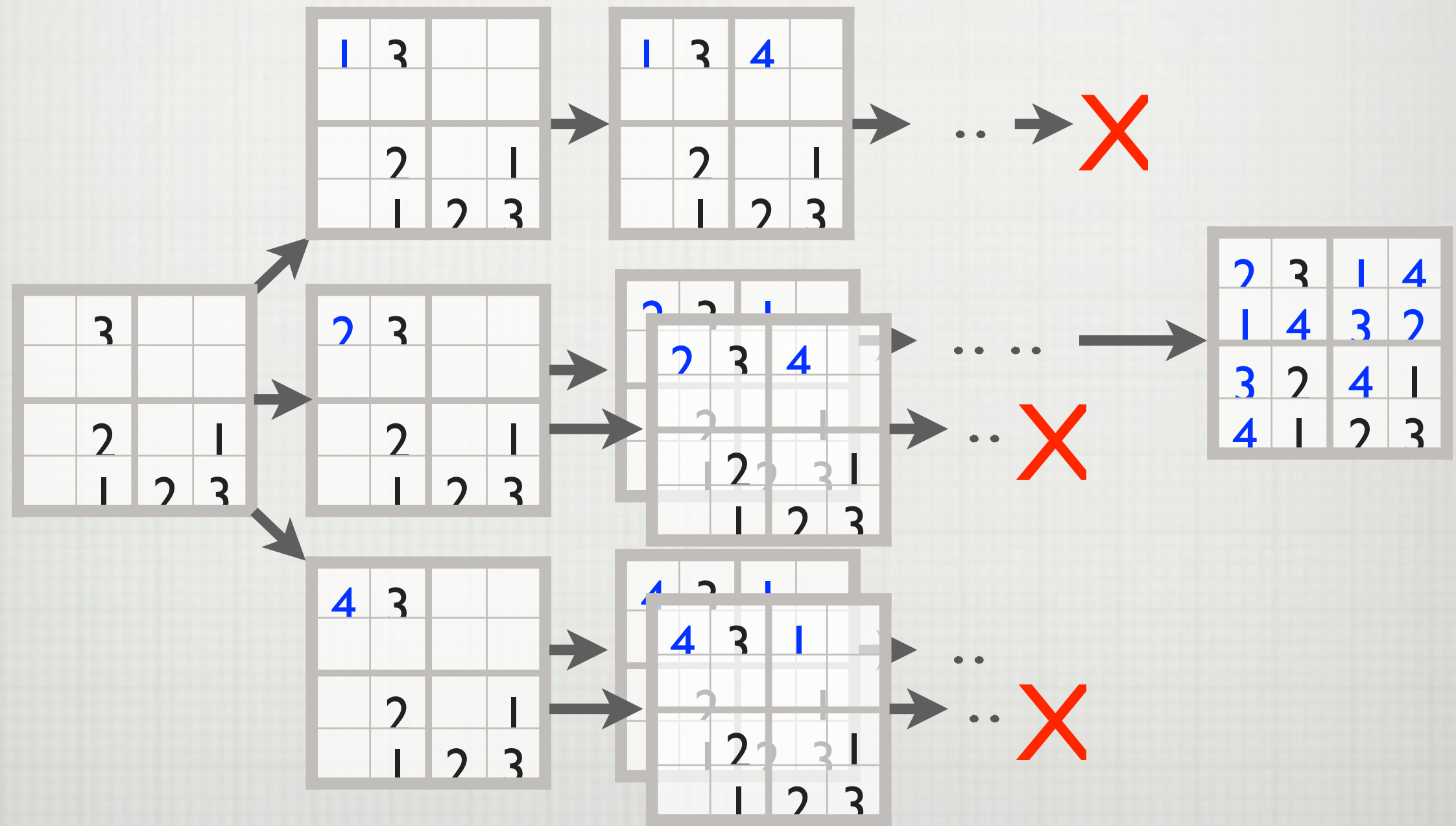


AAAI-15 / IAAI-15 Conference Program

January 25 – 30, 2015

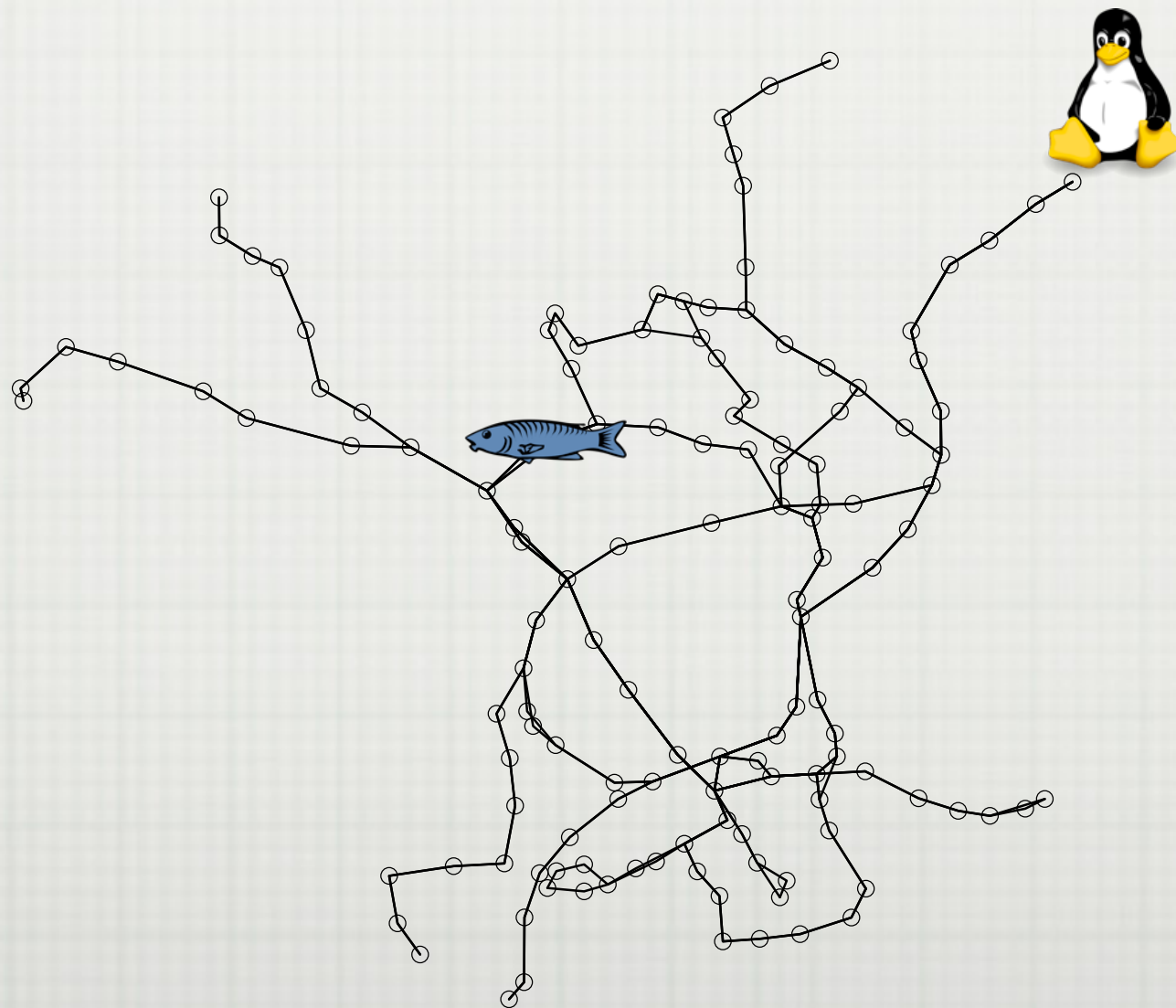
Hyatt Regency Austin, Austin, Texas, USA

ETSINTÄ



ETSINTÄ

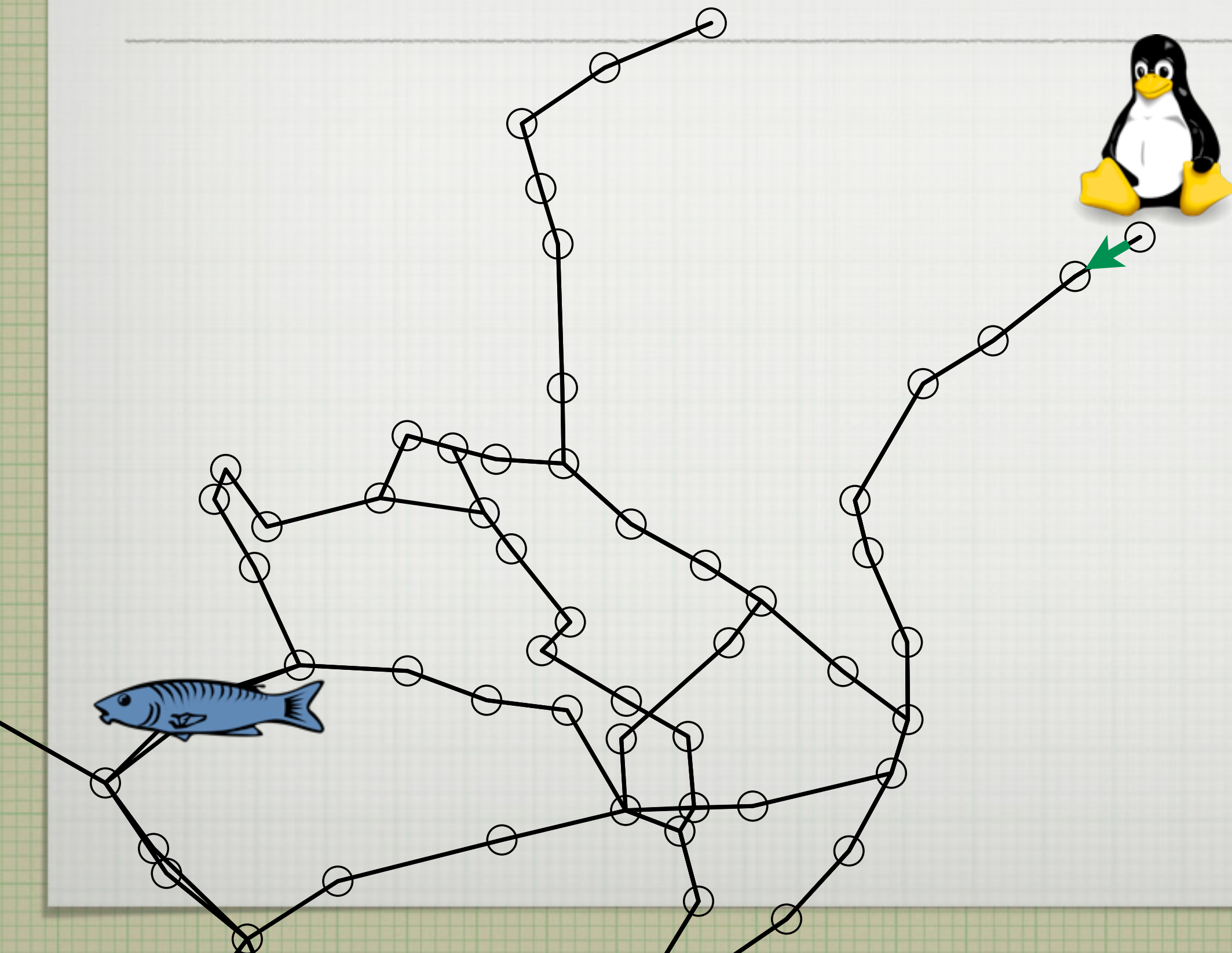
LEVEYSSUUNTAINEN HAKU



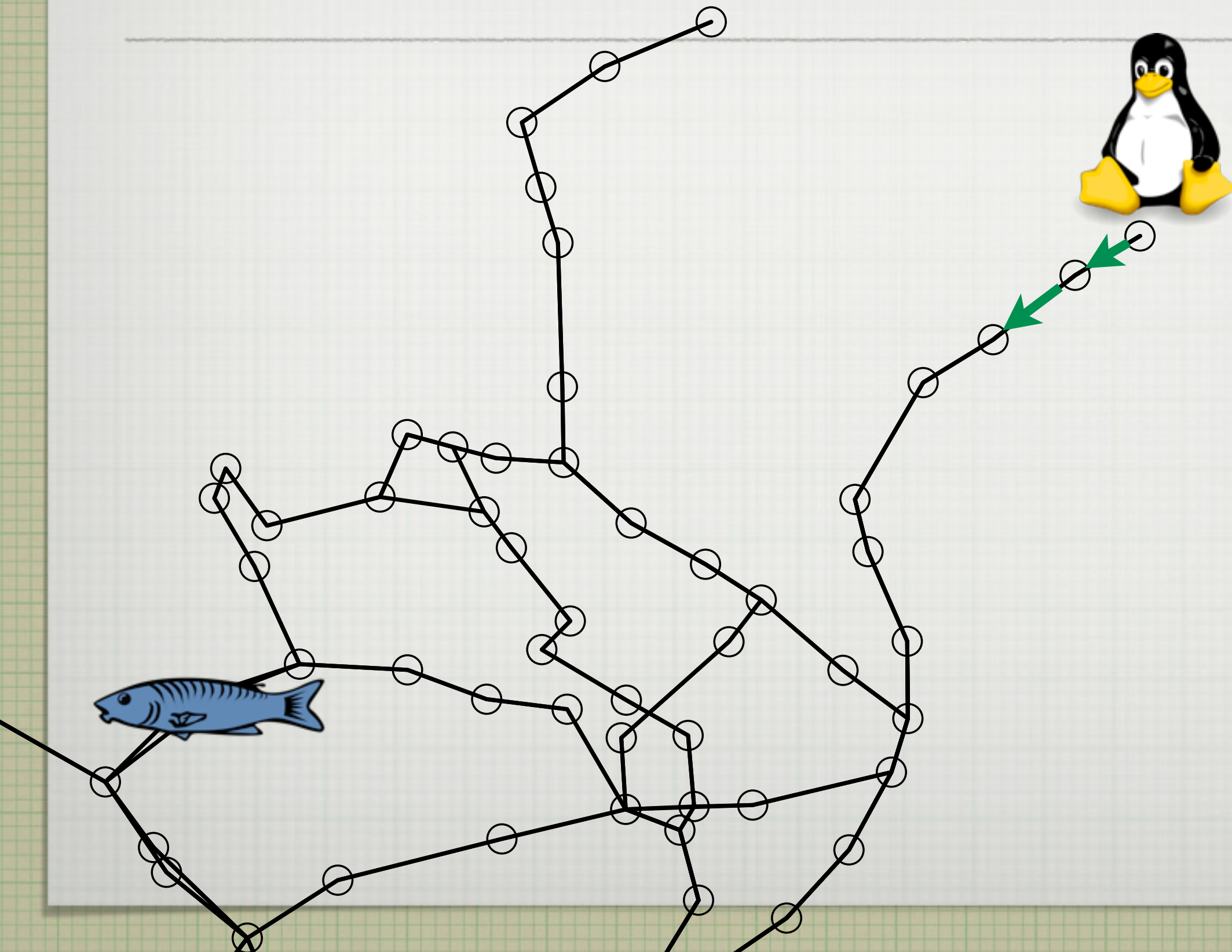
ETSINTÄ



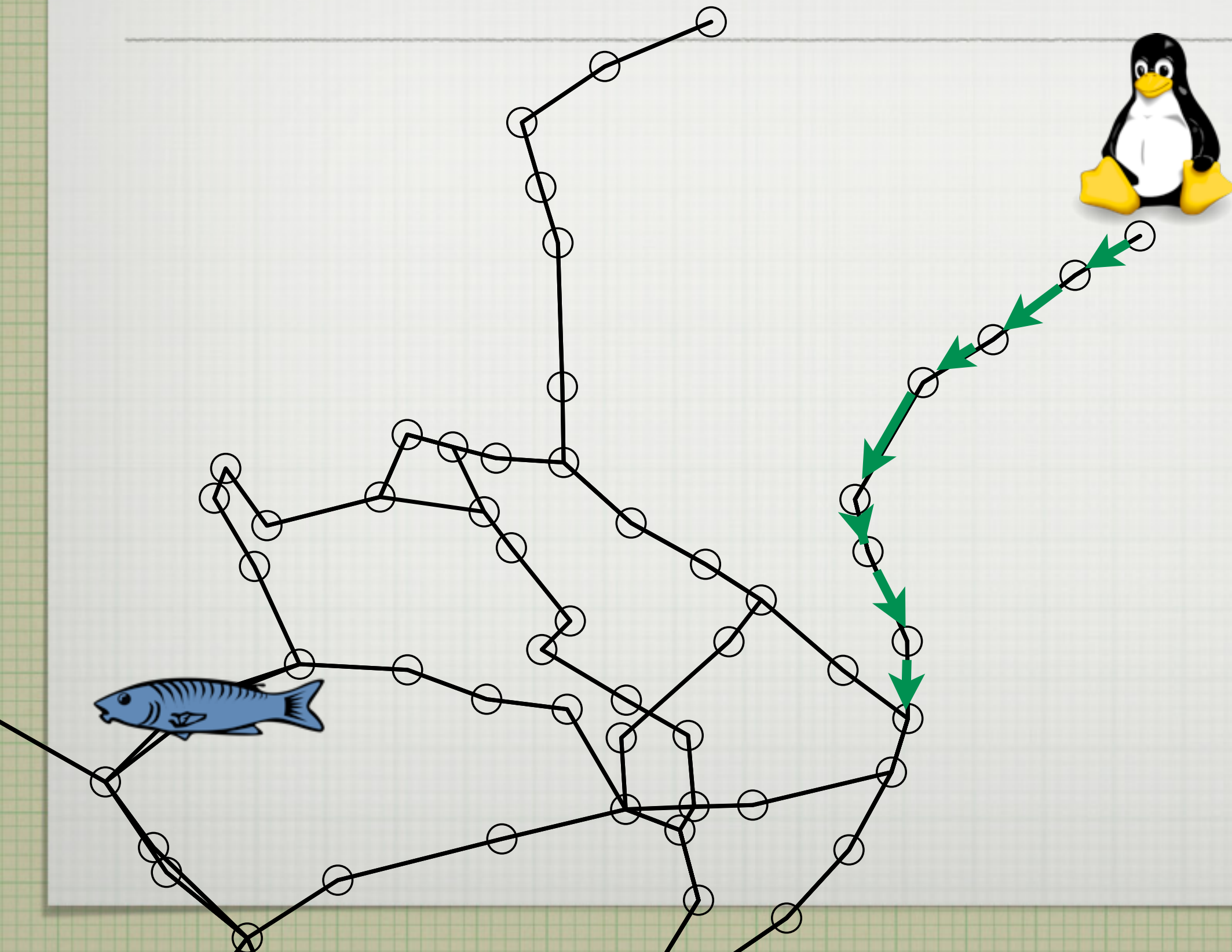
ETSINTÄ



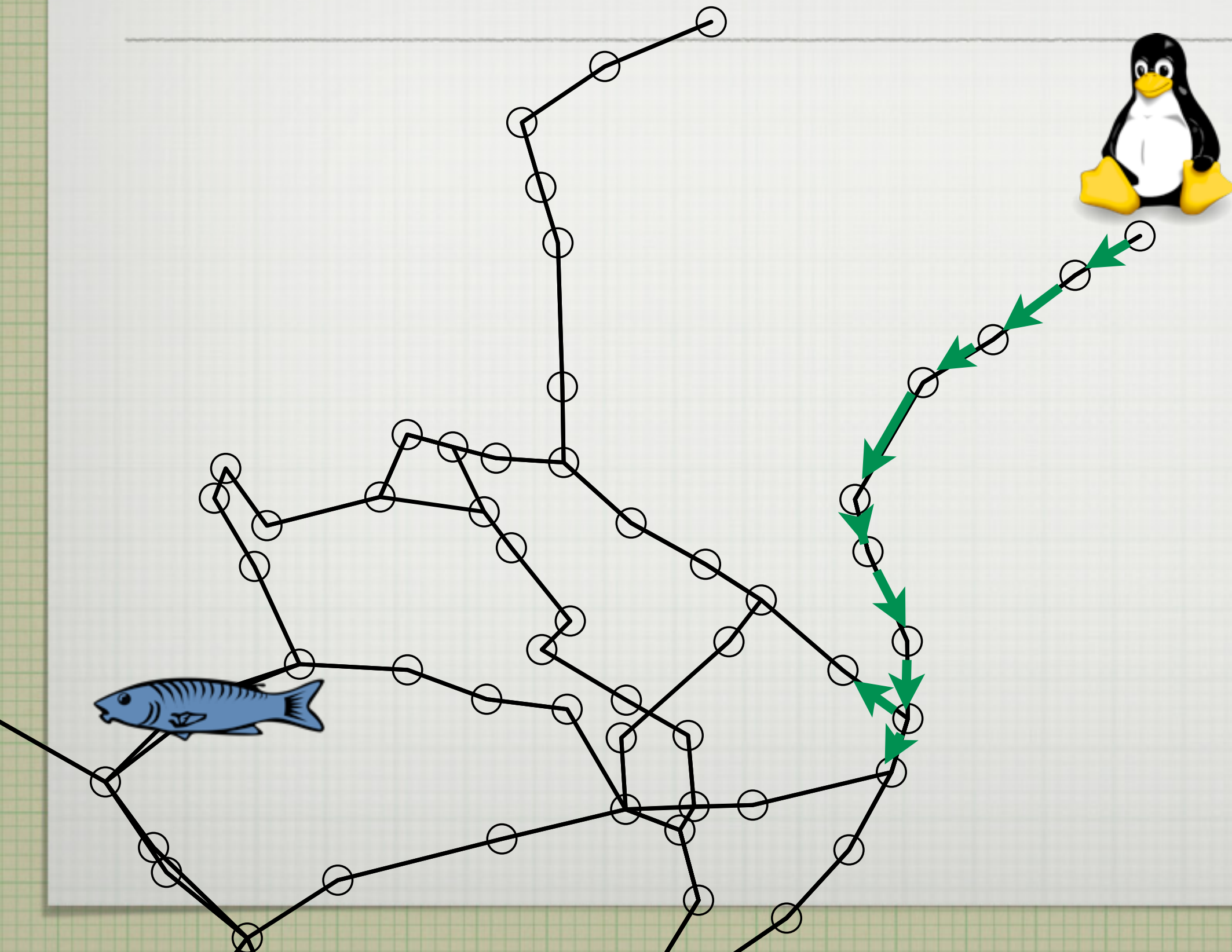
ETSINTÄ



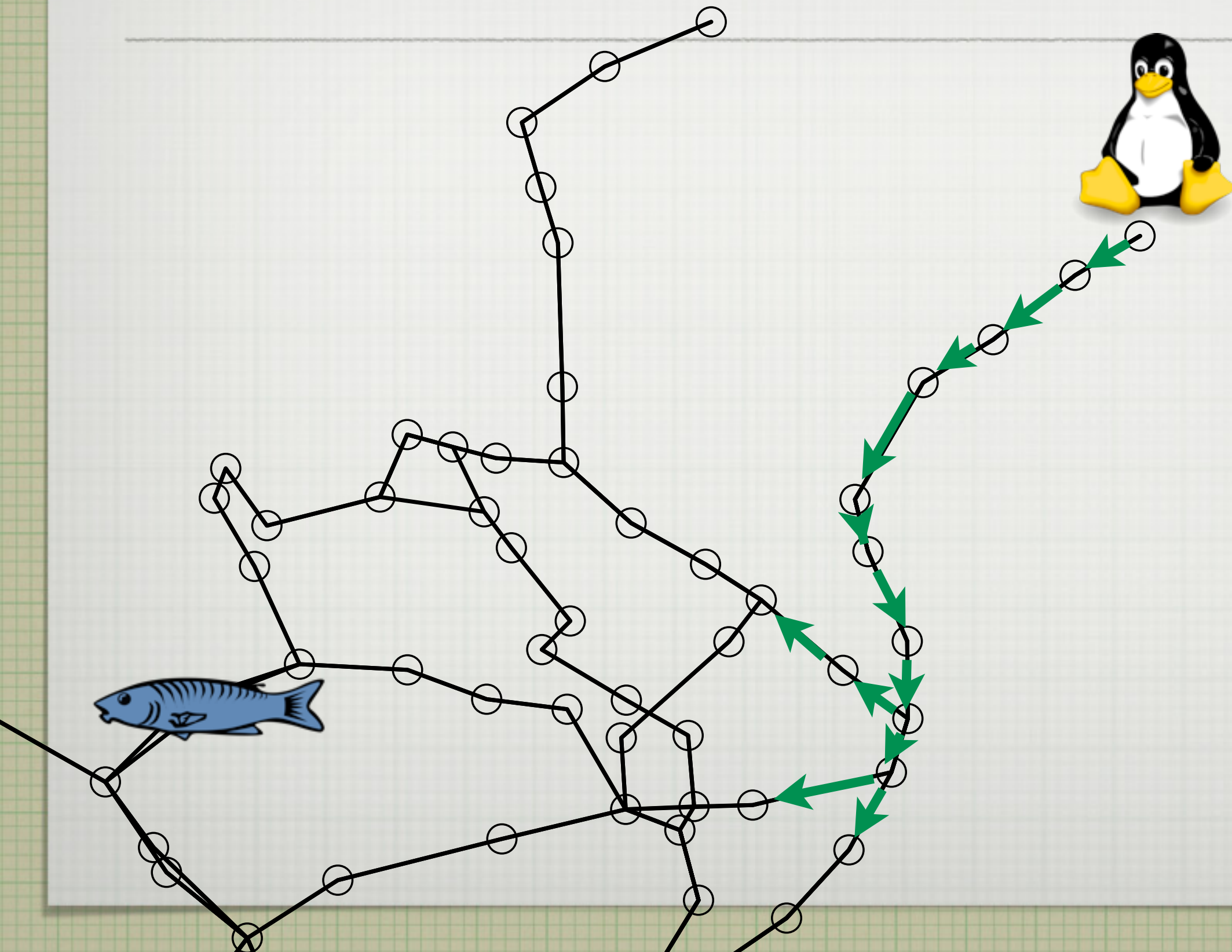
ETSINTÄ



ETSINTÄ

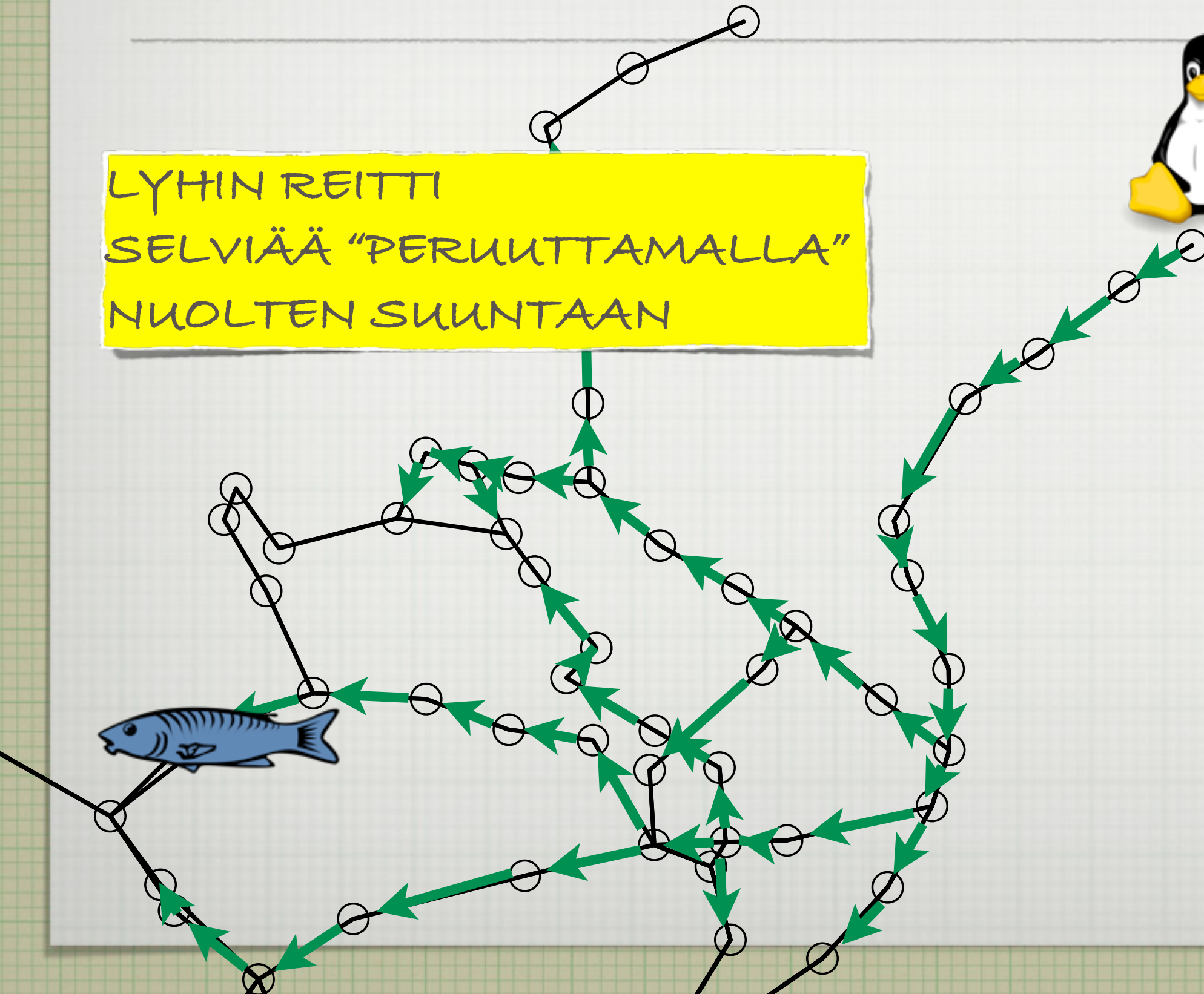


ETSINTÄ



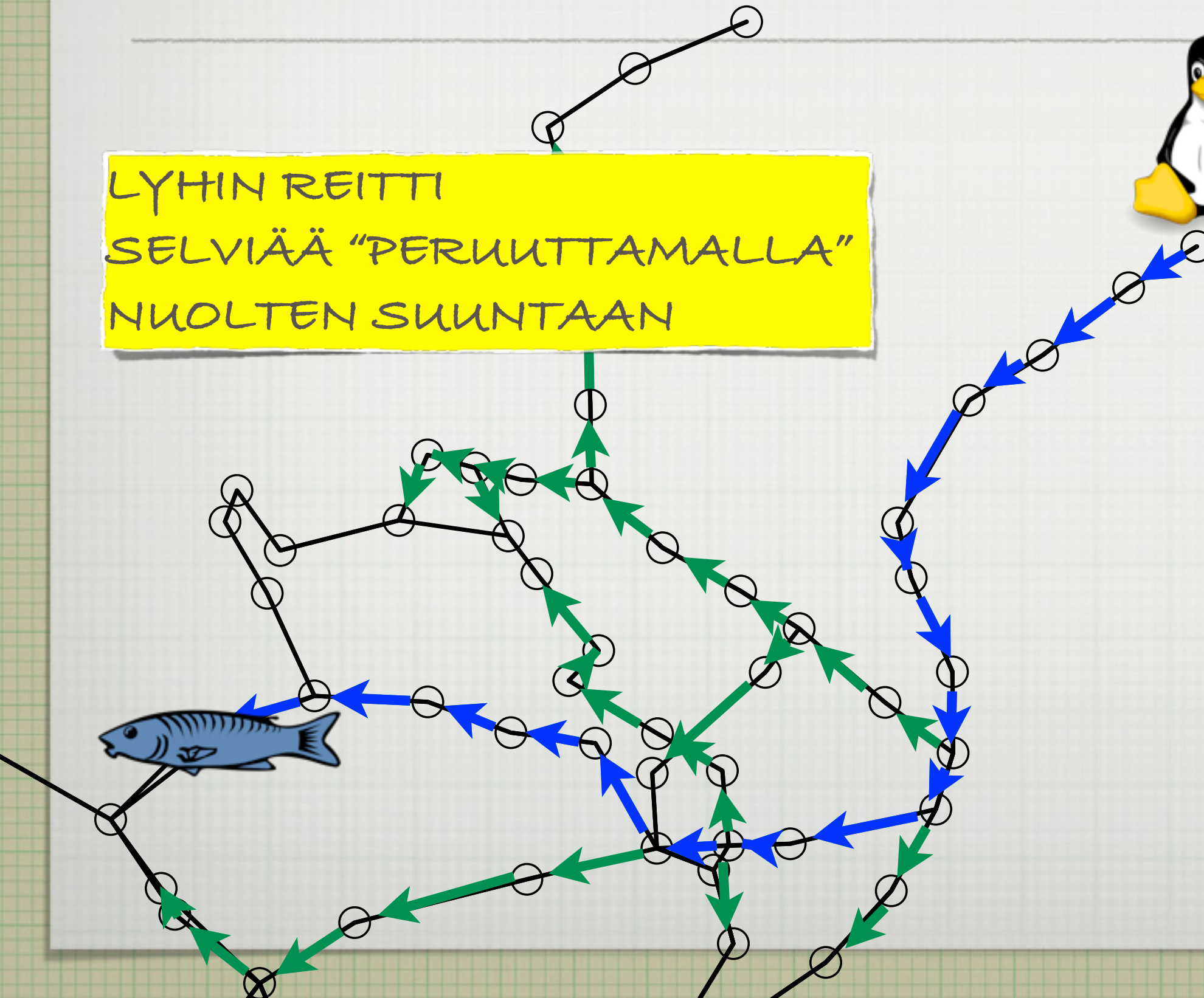
ETSINTÄ

LYHIN REITTI
SELVIÄÄ "PERUUTTAMALLA"
NUOLTEN SUUNTAAN



ETSINTÄ

LYHIN REITTI
SELVIÄÄ "PERUUTTAMALLA"
NUOLTEN SUUNTAAN



ETSINTÄ

ETSINTÄ(Alkusolmu)

Solmulista = [Alkusolmu]

Käsiteltyt = []

while Solmulista not empty

 Solmu = EKA(Solmulista)

 Solmulista = LOPUT(Solmulista)

 if Solmu not in Käsiteltyt

 Käsiteltyt = Käsiteltyt + [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 – Käsiteltyt – 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]

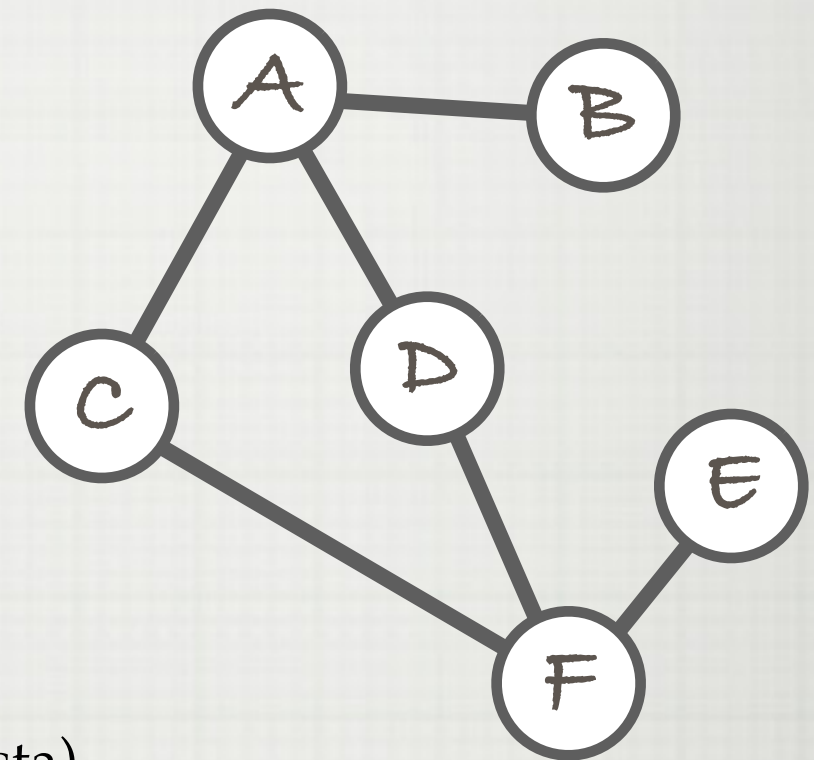
 if MAALI(Solmu) return("ratkaisu", Solmu)

 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

[A]

ETSINTÄ(Alkusolmu)

Solmulista = [Alkusolmu]

Käsitellyt = []

while Solmulista not empty

 Solmu = EKA(Solmulista)

 Solmulista = LOPUT(Solmulista)

 if Solmu not in Käsitellyt

 Käsitellyt = Käsitellyt + [Solmu]

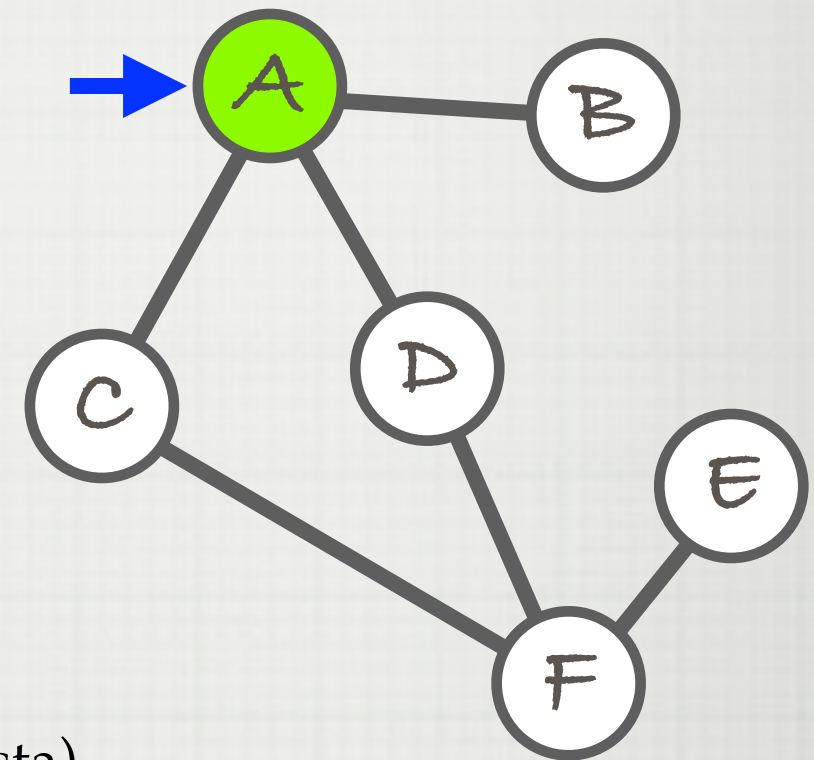
 if MAALI(Solmu) return("ratkaisu", Solmu)

 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]

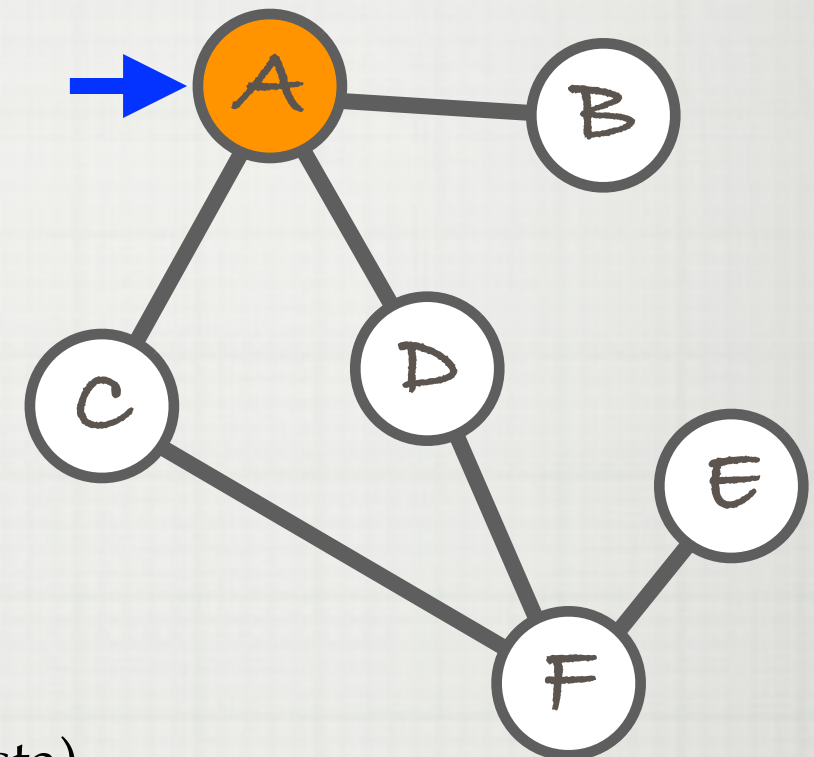
 if MAALI(Solmu) return("ratkaisu", Solmu)

 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]

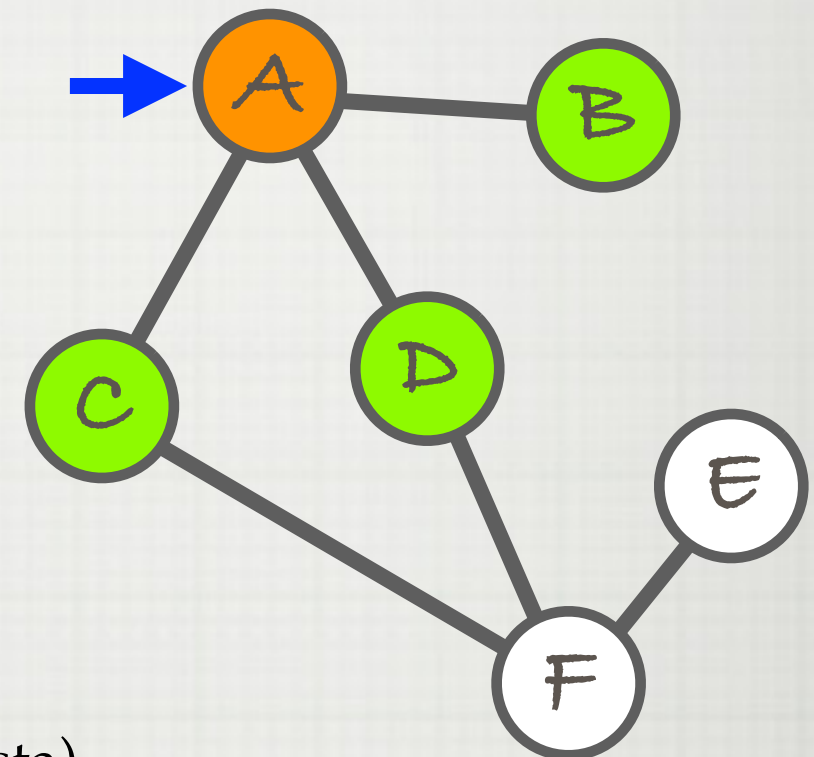
 if MAALI(Solmu) return("ratkaisu", Solmu)

 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]

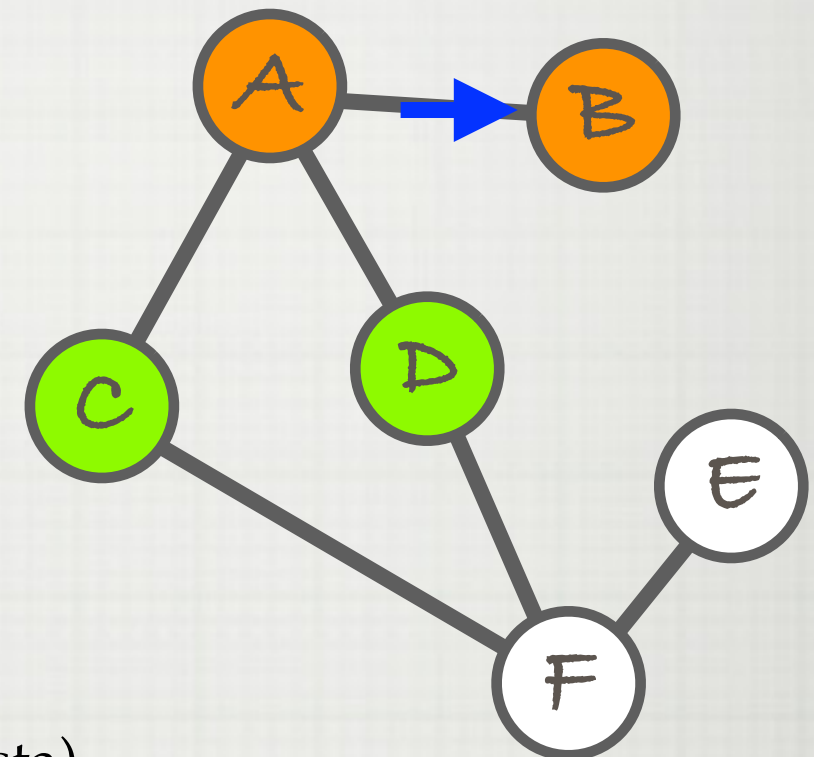
 if MAALI(Solmu) return("ratkaisu", Solmu)

 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]

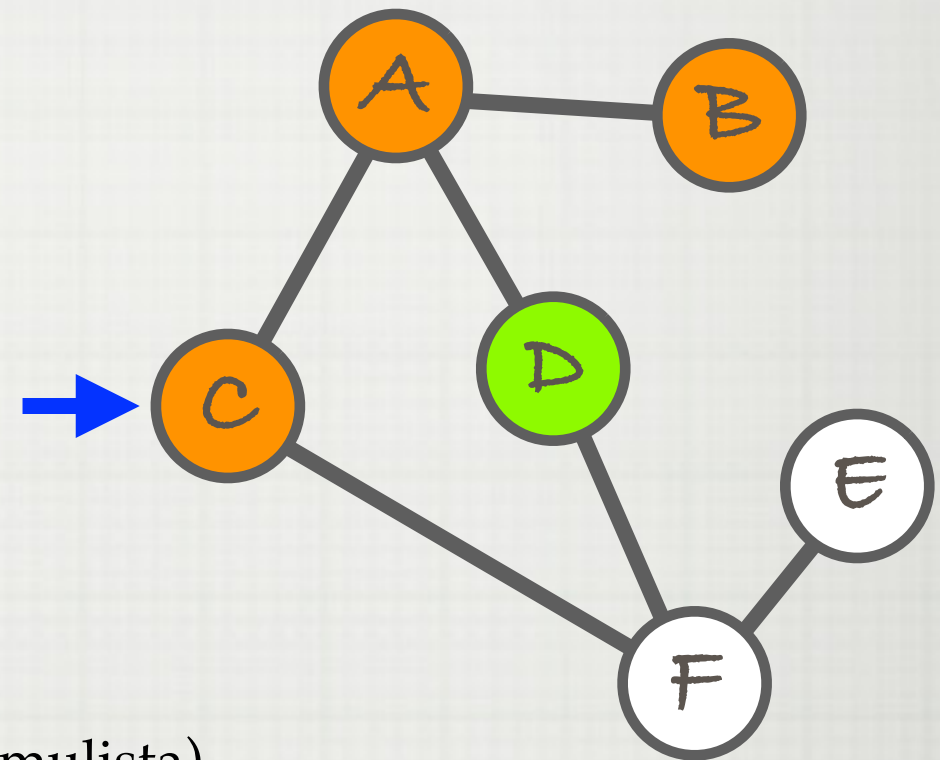
 if MAALI(Solmu) return("ratkaisu", Solmu)

 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]

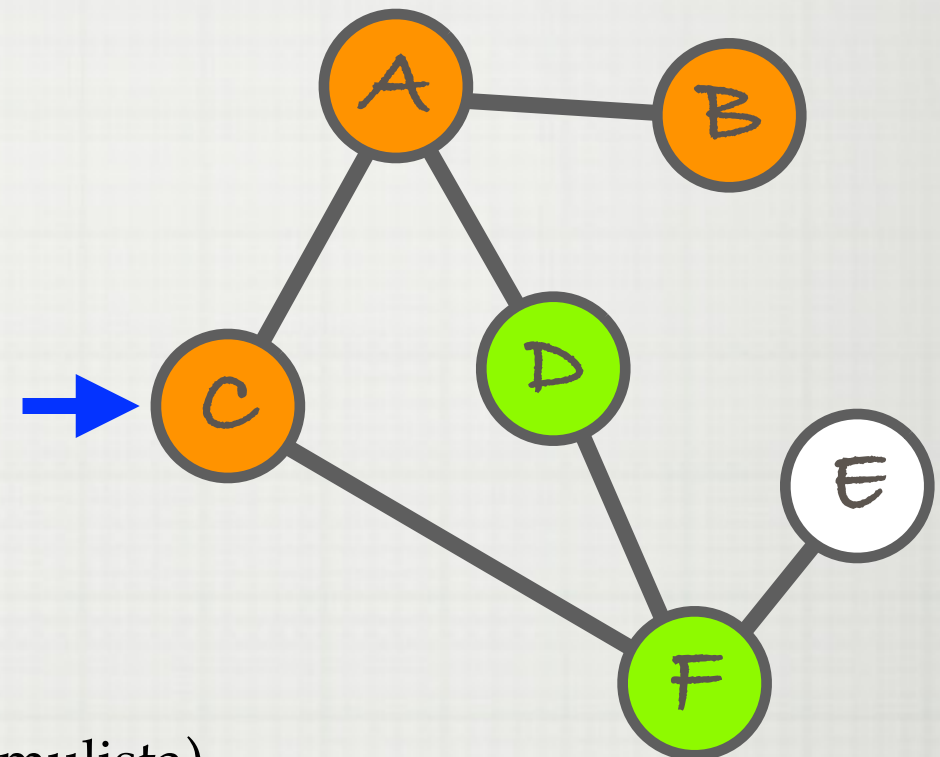
 if MAALI(Solmu) return("ratkaisu", Solmu)

 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]

Käsiteltyt = []

while Solmulista not empty

 Solmu = EKA(Solmulista)

 Solmulista = LOPUT(Solmulista)

 if Solmu not in Käsiteltyt

 Käsiteltyt = Käsiteltyt + [Solmu]

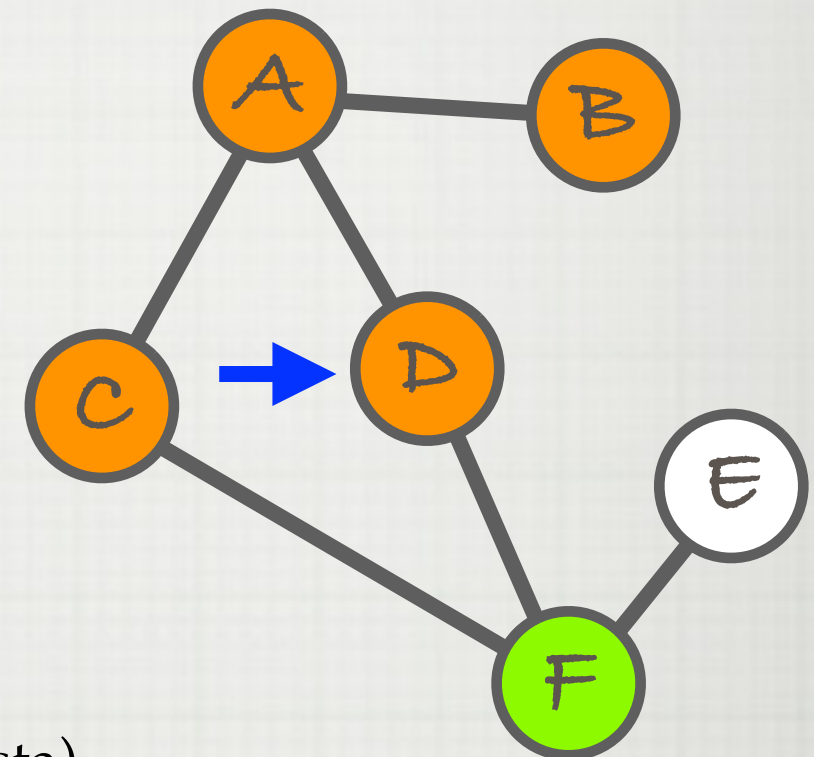
 if MAALI(Solmu) return("ratkaisu", Solmu)

 Solmulista = LISÄÄ(NAAPURIT(Solmu), Solmulista)

 end if

end while

return("ei ratkaisua")



LISÄÄ(Naapurit, Solmulista)

LEVEYSSUUNTAINEN HAKU:

Uudet = Naapurilista – Käsiteltyt – 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]

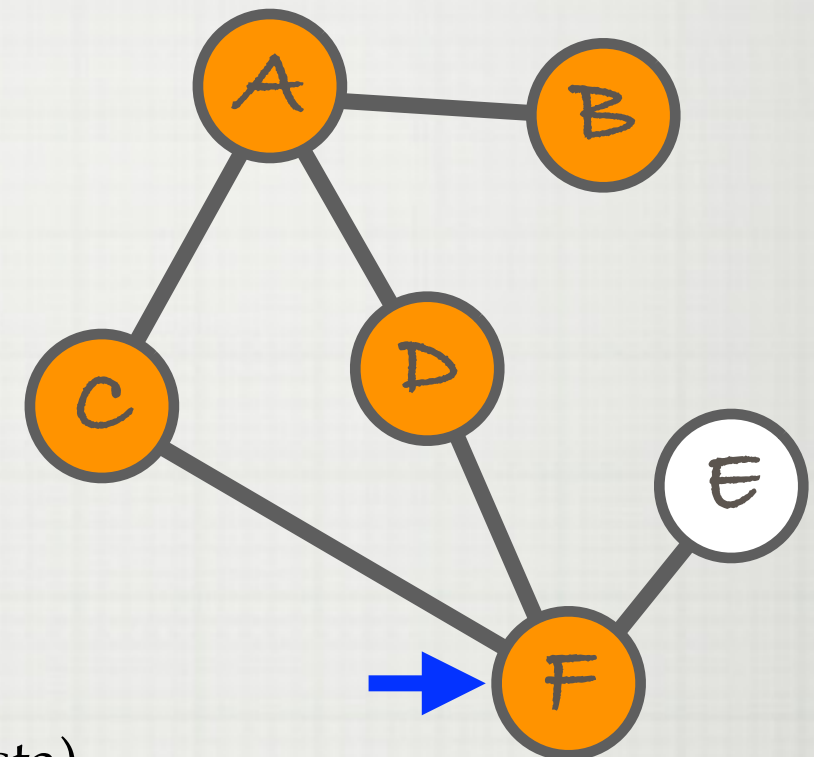
 if MAALI(Solmu) return("ratkaisu", Solmu)

 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]

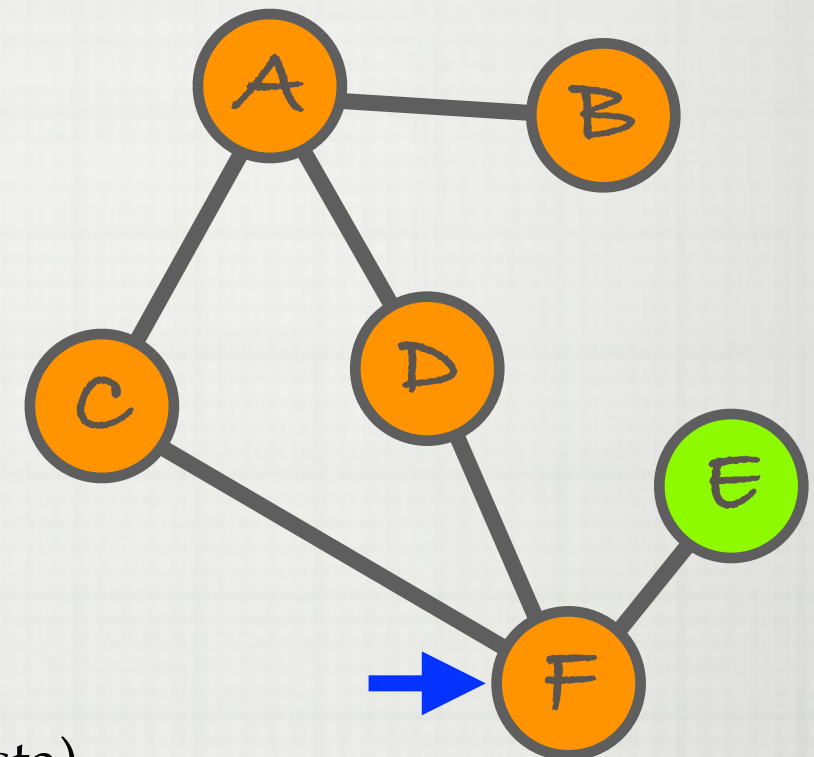
 if MAALI(Solmu) return("ratkaisu", Solmu)

 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]

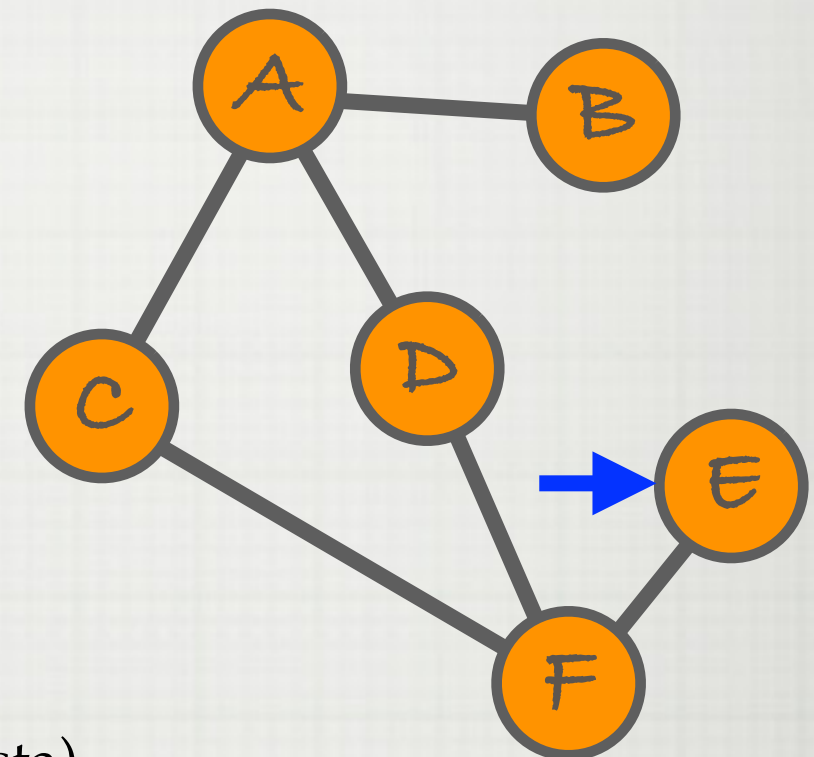
 if MAALI(Solmu) return("ratkaisu", Solmu)

 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]

Käsiteltyt = []

while Solmulista not empty

 Solmu = EKA(Solmulista)

 Solmulista = LOPUT(Solmulista)

 if Solmu not in Käsiteltyt

 Käsiteltyt = Käsiteltyt + [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 – Käsiteltyt – Solmulista

return(PERÄKKÄIN(Solmulista, Uudet))

LISÄÄ([a,f],[d]) => [d,a,f]

ETSINTÄ

ETSINTÄ(Alkusolmu)

Solmulista = [Alkusolmu]

Käsiteltyt = []

while Solmulista not empty

 Solmu = EKA(Solmulista)

 Solmulista = LOPUT(Solmulista)

 if Solmu not in Käsiteltyt

 Käsiteltyt = Käsiteltyt + [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 – Käsiteltyt

return(PERÄKKÄIN(Uudet, Solmulista))

LISÄÄ([a,f],[d]) => [a,f,d]

ETSINTÄ ONGELMANRATKAISUNA



ETSINTÄ ONGELMANRATKAISUNA

- * KOLME KANNIBAALIA JA KOLME LÄHETYSSAARNAAJAA HALUAA YLITTÄÄ JOEN VENEELLÄ, JOHON MAHTUU VAIN KAKSI HENKILÖÄ.
- * JOS JOMMALLA KUMMALLA RANNALLA ON ENEMMÄN KANNIBAALIA KUIN LÄHETYSSAARNAAJIA (MUTTA KUITENKIN VÄHINTÄÄN YKSI LÄHETYSSAARNAAJA), KANNIBAALIT SYÖVÄT HEIDÄT.
- * MITEN JOKI SAADAAN YLITETTYÄ ILMAN, ETTÄ KETÄÄN SYÖDÄÄN?
- * VOIT KOKEILLA KLIKKAAamalla 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			
	2		1
	1	2	3

SUDOKU

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

SUDOKU

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

SUDOKU

1	3	4	2
2	?		
	2		1
	1	2	3

SUDOKU

1	3	4	2
4			
	2		1
	1	2	3

SUDOKU

1	3	4	2
4	?		
	2		1
	1	2	3

SUDOKU

1	3	4	2
?			
	2		1
	1	2	3

SUDOKU

1	3	4	?
	2		1
	1	2	3

SUDOKU

1	3	?	
	2		1
	1	2	3

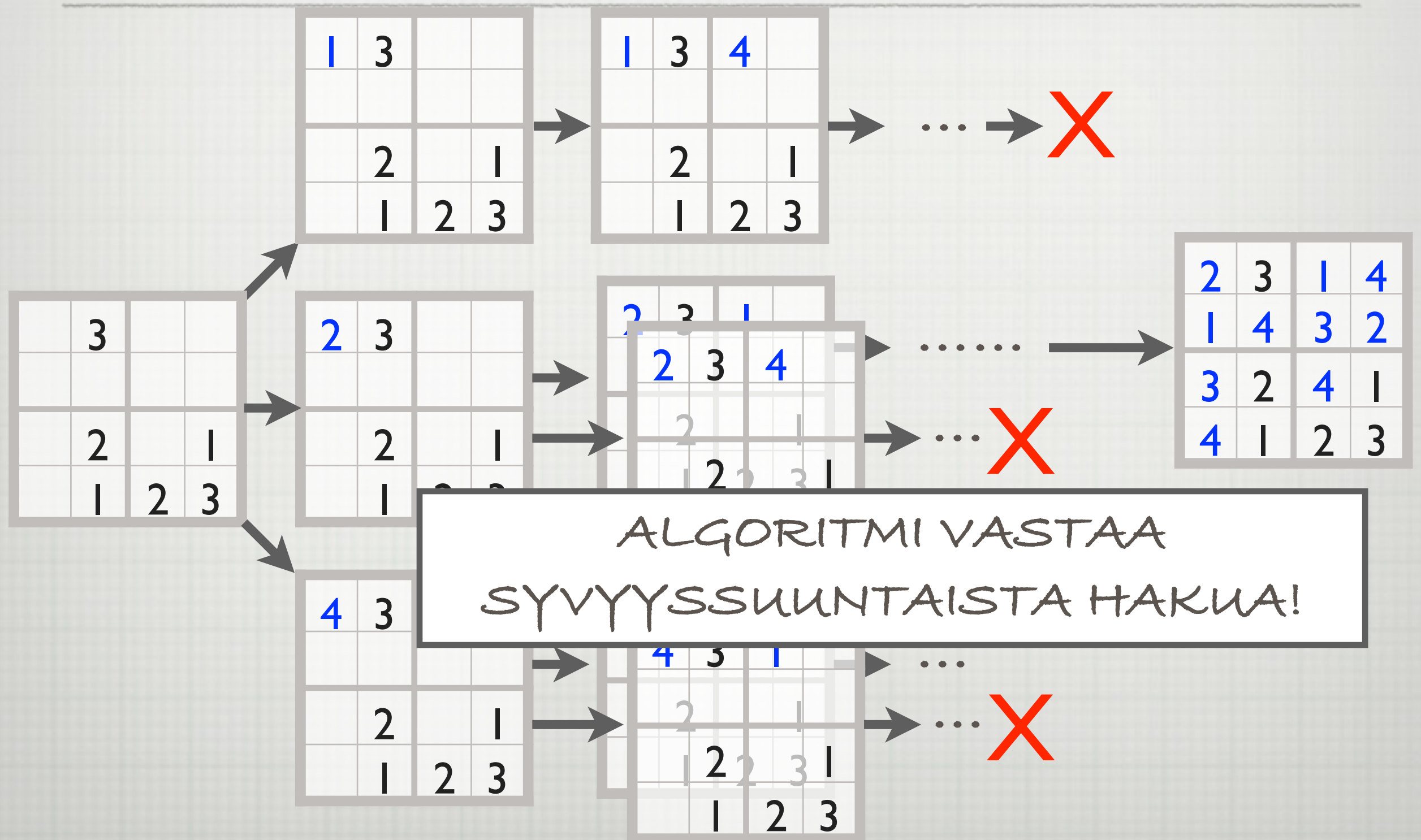
SUDOKU

2	3		
	2		1
	1	2	3

SUDOKU

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

SUDOKU



PARAS-ENSIN-HAKU

ETSINTÄ(Alkusolmu)

Solmulista = [Alkusolmu]

Käsiteltyt = []

while Solmulista not empty

 Solmu = EKA(Solmulista)

 Solmulista = LOPUT(Solmulista)

 if Solmu not in Käsiteltyt

 Käsiteltyt = Käsiteltyt + [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),(c,3),(b,3),(f,5),(a,5)]

HEURISTIIKAT

- * **KUSTANNUUSARVIO:** $f(N)$
 - ARVIO LÄHTÖSOLMUSTA SOLMUN N KAUTTA MAALISOLMUUN KULKEVAN POLUN KUSTANNUKSESTA
- * **"HEURISTIIKKA":** $h(N)$
 - ARVIO KUSTANNUKSESTA SOLMUSTA N MAALISOLMUUN
- * **POLKUKUSTANNUKSE:** $g(N)$
 - KUSTANNUKSE 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")

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

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),(c,3),(b,3),(f,5),(a,5)]

A*

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



Reittiopas

[Taskuversio](#) • [På svenska](#) • [In English](#) • [Slangi](#) • [По-русски](#)

[Palaute](#) • [Ohjeet](#) • [FAQ](#)

[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	
36	05	06	07	08	09	10	11	
37	12	13	14	15	16	17	18	
38	19	20	21	22	23	24	25	
39	26	27	28	29	30	01	02	
40	03	04	05	06	07	08	09	

Syyskuu
2011

Lokakuu

Hae

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.](#)

Omat reitit

Ei omia reittejä ([ohje omien reittien tallentamiseen](#)).

Omat paikat

Ei omia paikkoja ([ohje omien paikkojen tallentamiseen](#)).

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

HUOMENNA

- ✱ PELIT
- ✱ CASE: SHAKKI
- ✱ MINIMAX JA ALPHA-BETA-KARSINTA