Slide 1

## Tik-79.240 Laskennallisen vaativuuden erikoiskurssi (3 ov)
## syksy 1998

**Tilaisuudet:** tiistaisin klo 16–19, sali TB353

**Vastaava opettaja :** Dos. Ilkka Niemelä,
puh. 451 3290, e-mail: Ilkka.Niemela@hut.fi.

**Assistentti:** DI Tommi Junttila,
puh. 451 2896, e-mail: Tommi.Junttila@hut.fi.

**Kotisivu:**
`http://saturn.hut.fi/html/Tik-79.240.html`

Slide 2

### Tavoitteet

- Keskeisimmät vaativuusluokat (P, NP, PSPACE, NC, polynominen hierarkia, ...) ja niihin liittyvät menetelmät, joilla ongelmien vaativuusanalyysia voidaan tehdä.
- Rinnakkainen laskenta
- Satunnaistettu laskenta
- Kryptografia

**Materiaali:** C. Papadimitriou, Computational Complexity, Addison-Wesley, 1994. Ch. 1–3, 7–20



Slide 3

### Kurssin suorittaminen

- seminaariesitelmät
- kotilaskut
  1 kotilasku/esitelmä; arvostellaan asteikolla 0-2

Arvosana: määräytyy lasketuista kotilaskuista saatujen pisteiden perusteella (ei tenttiä)

| Arvosana | alaraja |
|---|---|
| 1 | 50 % |
| 2 | 57 % |
| 3 | 64 % |
| 4 | 71 % |
| 5 | 78 % |

Slide 4

### Järjestelyt

1. Vuorossa oleva esitelmöijä valmistelee esitelmän ja vast. opettaja valitsee ko. luvun tehtävistä esitelmään liittyvän kotitehtävän.

2. Pidetään esitelmä (n. 45min) ja ilmoitetaan kotitehtävä, joka esille kotisivulle (esim. Syrjänen/2.8.7).

3. Muut tekevät tehtävän ja palauttavat sen assistentille viimeistään 2 viikon kuluttua esitelmästä pidettävässä seminaaritilaisuudessa.

4. Esitelmöijä tarkastaa tehtävän ja ilmoittaa tulokset assistentille 3. kohdan takarajasta viimeistään 1 viikon kuluttua olevassa seminaaritilaisuudessa. Tulokset tulevat nähtäville kotisivulle.

Slide 5

## Ongelma vs. algoritmi

Kurssi painottuu *ongelmien* (ei algoritmien) analysointiin.

- Ongelma: ääretön joukko ilmentymiä, joista esitetään kysymys
- Päätösongelma: Kysymykseen kyllä/ei vastaus

**Esimerkki.** REACHABILITY-ongelma
Ilmentymä: Graafi $(V, E)$ ja solmut $v, u \in V$.
Kysymys: Löytyy graafista polku solmusta $v$ solmuun $u$?

Slide 6

Algoritmi REACHABILITY-ongelmalle:

$S := \{v\}$;
Merkitään solmu $v$;
**while** $S \neq \{\}$ **do**
    Valitaan solmu $i$ ja poistetaan se joukosta $S$;
    Kaikille kaarille $(i, j) \in E$:
        jos $j$ on merkitsemättä, merkitään se ja lisätään joukkoon $S$
**endwhile**
Jos $u$ on merkitty, palautetaan vastaus "kyllä"
    muutoin palautetaan vastaus "ei"



Slide 7

Kuinka tehokas algoritmi on?

- Ohjelmointikieli?
- Tietokonearkkitehtuuri?
- Graafin esitys?
- Joukon $S$ esitys?

Tietyillä oletuksilla algoritmi tarvitsee $\mathcal{O}(|E|)$ askelta.

Slide 8

**Funktioiden kasvuvauhti**

Olkoon $f$ ja $g$: $\mathbf{N} \mapsto \mathbf{N}$. Merkitään

- $f(n) = \mathcal{O}(g(n))$, jos on olemassa luonnolliset luvut $c$ ja $n_0$ siten, että kaikilla $n \geq n_0$, $f(n) \leq c \cdot g(n)$
- $f(n) = \Omega(g(n))$, jos $g(n) = \mathcal{O}(f(n))$
- $f(n) = \Theta(g(n))$, jos $g(n) = \mathcal{O}(f(n))$ ja $f(n) = \mathcal{O}(g(n))$.

**Esimerkki.** Jos $p(n)$ on polynomi, jonka asteluku on $d$, $p(n) = \Theta(n^d)$.
Jos $c > 1$ on kokonaisluku ja $p(n)$ on polynomi, $p(n) = \mathcal{O}(c^n)$ mutta $p(n) \neq \Omega(c^n)$.

**Slide 9**

Ongelmien analysointia varten tehdään yksinkertaistuksia:

- Laskennan malli: Turingin kone
- Ongelma tehokkaasti ratkaistavissa: polynominen aikavaativuus ($\mathcal{O}(n^d)$)
- Ei tehokasta ratkaisua (intractable): aikavaativuus eksponentiaalinen
- Tarkastellaan huonoimman tapauksen vaativuutta.

**Slide 10**

Kritiikkiä:

- Polynomiset algoritmit eivät ole aina käytännössä tehokkaita ja toisaalta käytännössä tehokkaille algoritmeille ei voida aina taata polynomista vaativuutta. ($n^{80}$ vs $2^{\frac{n}{100}}$)
- Keskimääräisen tapauksen analyysi tärkeämpi kuin huonoimman tapauksen.

☞ Mutta käyttämällä tätä yksinkertaistusta saadaan käyttökelpoinen teoria, joka kertoo käytännön kannalta merkittäviä asioita laskennasta ja ongelmien vaativuudesta.



**Slide 11**

## Esimerkkiongelmia

MAX FLOW-ongelma:
Ilmentymä: Verkko $N = (V, E, s, t, c)$, missä $(V, E)$ on (suunnattu) graafi, $s, t \in V$, solmuun $s$ ei tule kaaria eikä solmusta $t$ lähde kaaria ja $c$ on funktio, joka antaa kaarien kapasiteetit (luonnollisia lukuja).
Kysymys: Mikä on verkon vuon maksimiarvo?

Vuo on funktio $f$, joka antaa kullekin kaarelle $(i, j)$ luonnollisen luvun $f(i, j) \leq c(i, j)$ siten, että kaikille muille solmuille paitsi $s, t$ sisään tulevan vuon summa on ulos menevän vuon summa.
Verkon vuon arvo on solmusta $s$ lähtevän vuon summa.

**Slide 12**

- MAX FLOW on optimointiongelma.
- MAX FLOW(D)-päätösongelma: onko olemassa annettua lukua $K$ suurempaa vuon arvoa?
- MAX FLOW ja MAX FLOW(D) suurin piirtein yhtä vaikeita.
- MAX FLOW-ongelma on malliesimerkki tapauksesta, jossa ensin oli ongelmana löytää jokin algoritmi, jolla on polynominen aikavaativuus.

  Kun "eksponenttiaalisuusraja" murtui, seurasi sarja parannuksia, jossa aikavaativuus laski $\mathcal{O}(n^5), \ldots, \mathcal{O}(n^3), \ldots$

**Slide 13**

MATCHING-ongelma:

Ilmentymä: Bipartiitti graafi $B = (U, V, E)$, jossa $U = \{u_1, \ldots, u_n\}$, $V = \{v_1, \ldots, v_n\}$, $E \subseteq U \times V$.

Kysymys: Onko olemassa joukko $M \subseteq E$, jossa on $n$ alkiota ja jokaiselle kaariparille $(u, v), (u', v') \in M$, $u \neq u'$ ja $v \neq v'$.

**Slide 14**

- Voidaan ratkaista **palauttamalla** (redusoimalla) MAX FLOW-ongelmaan:
  Graafille $B = (U, V, E)$ muodostetaan verkko $(V \cup U \cup \{s, t\}, E', s, t, c)$, jossa $E' = E \cup \{(s, u) \mid u \in U\} \cup \{(v, t) \mid v \in V\}$ ja kaikkien kaarien kapasiteetti 1.
- Nyt graafille $B$ löytyy sovitus (matching), joss verkon vuolla on arvo $n$.
- Tehokas algoritmi vuon laskemiselle antaa tehokkaan algoritmin MATCHING-ongelmalle:
  palautus toteutettavissa tehokkaasti (polynomisessa ajassa).



**Slide 15**

TSP-ongelma:

Ilmentymä: $n$ kaupunkia $1, \ldots, n$ ja kullekin kaupunkiparille $(i, j)$ luonnollinen luku $d_{ij}$ (etäisyys), jolle pätee $d_{ij} = d_{ji}$.

Kysymys: Mikä on lyhin kierros, joka käy jokaisessa kaupungissa täsmälleen kerran ja palaa alkupisteeseensä (eli permutaatio $\pi$ kaupungeista siten, että

$$\sum_{i=1}^{n} d_{\pi(i)\pi(i+1)}$$

on pienin (missä $\pi(n+1) = \pi(1)$).

Päätösongelma TSP(D): Onko kierrosta, jonka pituus on korkeintaan $B$?

**Slide 16**

- Suoraviivainen algoritmi: käydään kaikki kierrokset läpi
  Ei käytännöllinen; $\mathcal{O}(n!)$ kierrosta (10! = 3 628 800)
- TSP-ongelmaa tutkittu paljon, mutta tämän hetken parhaillakin menetelmillä huonoimman tapauksen aikavaativuus eksponentiaalinen.
- Konjektuuri: TSP-ongelmalle ei ole olemassa polynomista ratkaisualgoritmia.
  Kytkeytyy tiivisti erääseen keskeisimmistä tietojenkäsittelyn auki olevista ongelmista:

$$\mathrm{P} = \mathrm{NP}?$$