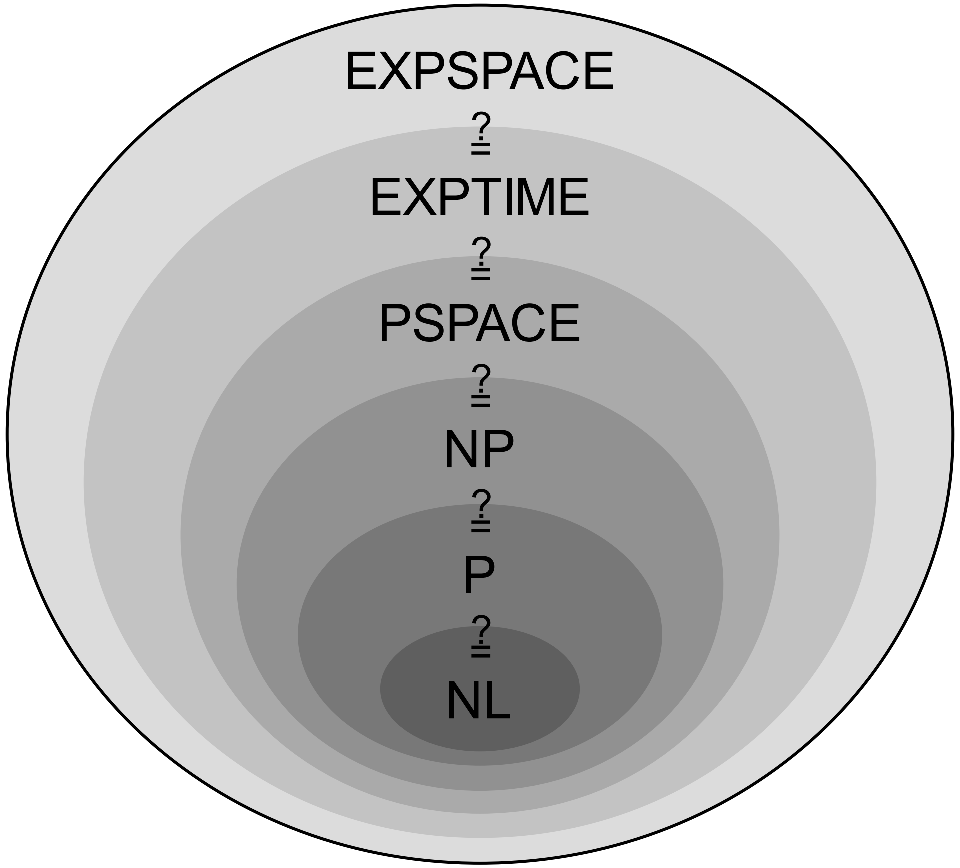
Nopeat algoritmit ja laskennallinen vaativuus teoria

Reijo Jaakkola
Tampere University

Laskennallinen vaativuus teoria

- Tavoite: ymmärtää laskennallisten ongelmien vaatimia resursseja
- Tietojenkäsittelytieteen lisäksi sovelluksia matematiikassa, fysiikassa, taloustieteessä, …

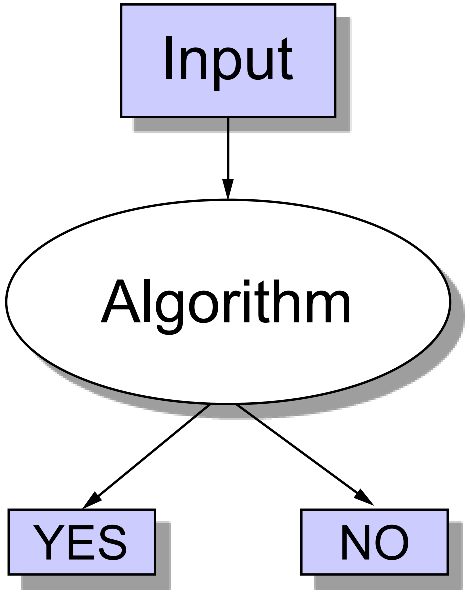


Mikä on laskennallinen ongelma?

Laskennallisessa ongelmassa P on kaksi osaa:

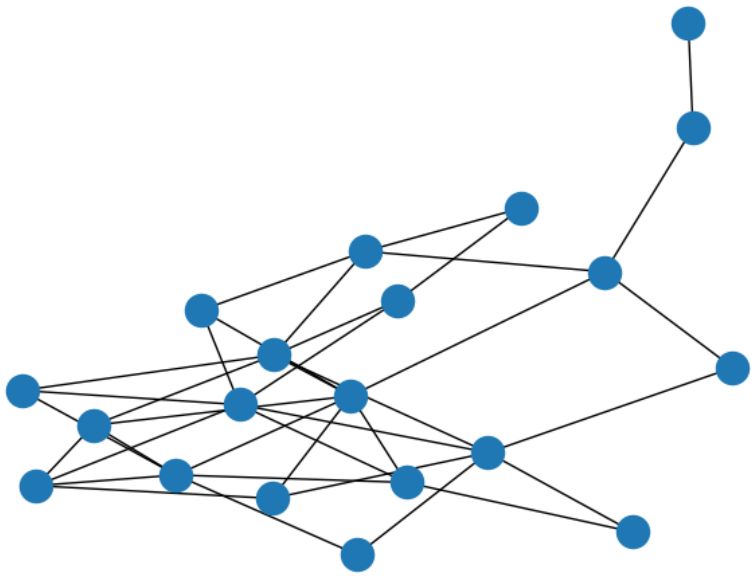
1. joukko syötteitä I (yleensä bittijonoja)
2. osajoukko $L \subseteq I$

- Algoritmi A ratkaisee ongelman P jos jokaisella $x \in I$ pätee, että A hyväksyy x joss $x \in L$



Esimerkkejä

- Onko annettu graafi yhtenäinen?
- Onko yhtälöryhmällä ratkaisuja?
- Onko annettu luku alkuluku?



$$x_1 + x_2 = 1$$
$$x_3 + x_1 = 0$$
$$x_2 + x_3 = 0$$
$$x_1, x_2, x_3 \in \{0,1\}$$

Tärkeä resurssi: aika

- Jos syötteen koko on n, niin kuinka kauan algoritmilla A kuluu ongelman ratkaisemisessa?

- Periaatteessa riippuu laskennan mallista (ja implementoinnista)

Asymptoottinen analyysi

$f(n) = O(g(n))$
joss on olemassa $C > 0$ ja n_0 s.e. $f(n) \leq Cg(n)$ kunhan $n \geq n_0$

- Esim. $10 = O(1)$ ja $22.4n^2 + 10.1n + 2 = O(n^2)$
- Algoritmin ajoajan asymptoottinen analyysi abstrahoi turhia (?) yksityiskohtia pois

Esimerkki

0 1 6 7 2 3 4 2 2 3 9 2 2 9 0 1 2 3 4 8 5 7 3 2 3 9 4 9 1 2 3 9 4 3 2

Syöte on lista $(x_1, \dots, x_n)$ jossa on n lukua.

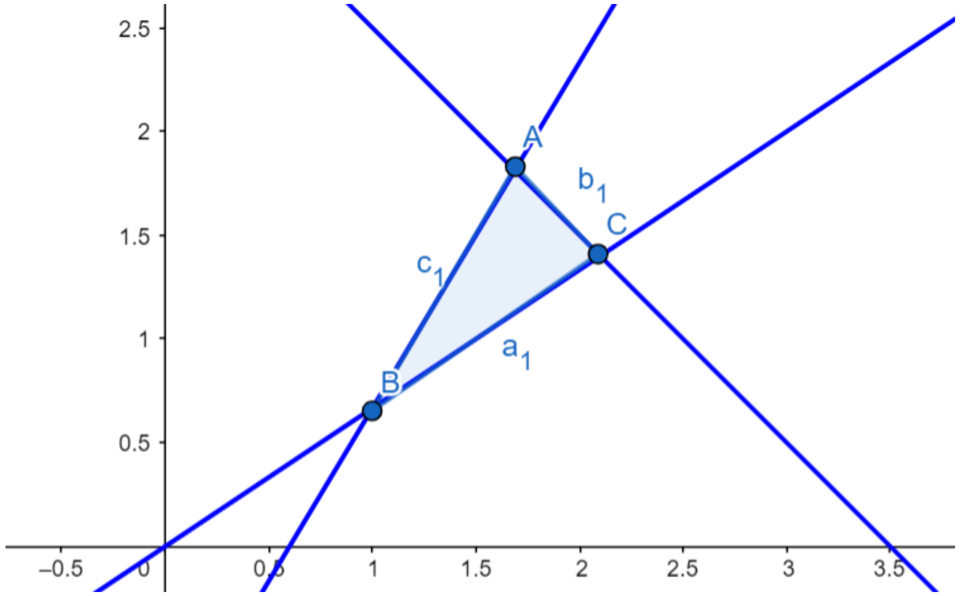
1. Algoritmi joka tarkistaa onko $x_1 = 0$: ajoaika $O(1)$
2. Algoritmi joka tarkistaa että $x_i = 0$ kaikilla i: ajoaika $O(n)$

Cobham-Edmonds teesi

- Algoritmi A on nopea (tai tehokas) jos on olemassa vakio $k > 0$ s.e. A vaatii $O(n^k)$ aikaa kaikilla syötteillä joiden koko on n.

$f(n) = O(g(n))$ joss on olemassa $C > 0$ ja n_0 s.e. $f(n) \leq Cg(n)$ kunhan $n \geq n_0$

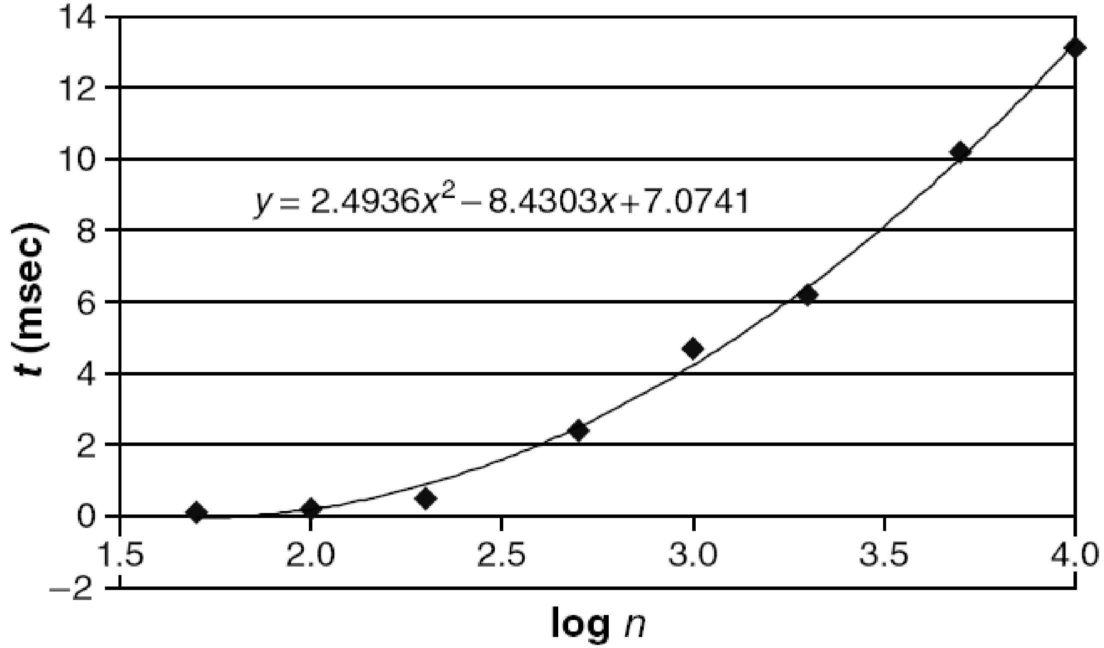
- Graafin yhtenäisyys
- Alkuluku tarkistus
- Lineaarinen ohjelmointi
- Hyperbolisten ryhmien sanaongelma
- Täydellinen paritus
- jne...



Cobham–Edmonds teesin ongelmia

- $O(n^{500})$-aikainen algoritmi on nopea ja $O\left(n^{\log(\log(\log(n)))}\right)$-aikainen algoritmi on hidas…

- Keskitytään "worst-case" ajoaikaan..



Onko olemassa vaikeita ongelmia?

- On, koska on olemassa ongelmia joita ei voi ratkaista millään algoritmilla… (Church & Turing)
- On, koska on olemassa ongelmia jotka vaativat eksponenttiaalisesti aikaa! (Hartmanis & Stearns)

Lause (Hartmanis & Stearn): On olemassa laskennallinen ongelma jonka voi ratkaista $O(2^n)$-aikainen algoritmi, mutta jota ei voi ratkaista mikään nopea, eli $O(n^k)$-aikainen, algoritmi.

"Todistus": Olkoon $(A_x)_{x\in\{0,1\}^*}$ lista **kaikista** algoritmeista. Määritellään uusi algoritmi B seuraavasti: millä tahansa syötteellä x jonka koko on n se "simuloi" algoritmia A_x syötteellä x 2^n-askelta ja hyväksyy joss A_x ei hyväksy x.

$f(n) = o(g(n))$ joss kaikille $\epsilon > 0$ on olemassa n_0 s.e. $f(n) \leq \epsilon g(n)$ kunhan $n \geq n_0$

Lause (Hartmanis & Stearn): Oletetaan, että $f(n) \log f(n) = o(g(n))$. Tällöin on olemassa ongelma jonka voi ratkaista $g(n)$-aikaisella algoritmilla, mutta ei $f(n)$-aikaisella algoritmilla.

- Kaikilla $k \geq 1$ pätee, että $n^k \log n^k = kn^k \log n = o(n^{k+1})$, joten jokaiselle $k \geq 1$ on olemassa ongelma jonka voi ratkaista $O(n^{k+1})$-aikaisella algoritmilla, mutta jota ei voi ratkaista $O(n^k)$-aikaisella algoritmilla.

Ketä kiinnostaa?

- Vaikka B ratkaisema ongelma ei itsessään ole luonnollinen, voimme koittaa redusoida sen johonkin luonnolliseen ongelmaan…

Ongelmien $P_1 = (I_1, L_1)$ ja $P_2 = (I_1, L_2)$ välinen nopea reduktio on kuvaus

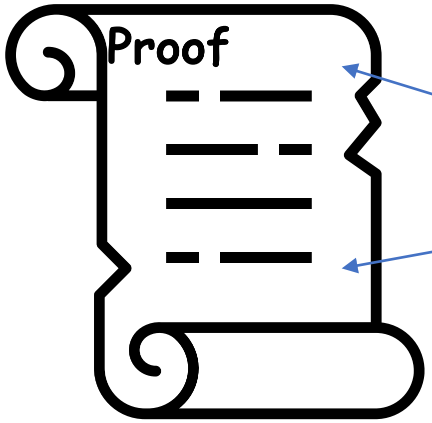
$$f: I_1 \rightarrow I_2$$

s.e. f on nopea ja $x \in L_1$ joss $f(x) \in L_2$

NP

- Monilla "käytännön" laskennallisilla ongelmilla on seuraava ominaisuus: niiden ratkaisut voidaan tarkistaa nopeasti (= NP)

- B ratkaisemaa ongelmaa ei voi (tod. näk.) redusoida tälläiseen ongelmaan (NP vs EXP)



NP-täydelliset ongelmat

- Käyttämällä reduktioita on osoittamaan, että monet ongelmat NP:ssä ovat "täydellisiä"
- P vs NP on kysymys siitä voidaanko NP täydellistä ongelmaa ratkaista nopeasti

- **Satisfiability**: the boolean satisfiability problem for formulas in conjunctive normal form (often referred to as SAT)
  - **0–1 integer programming** (A variation in which only the restrictions must be satisfied, with no optimization)
  - **Clique** (see also independent set problem)
    - **Set packing**
    - **Vertex cover**
      - **Set covering**
      - **Feedback node set**
      - **Feedback arc set**
      - **Directed Hamilton circuit** (Karp's name, now usually called **Directed Hamiltonian cycle**)
        - **Undirected Hamilton circuit** (Karp's name, now usually called **Undirected Hamiltonian cycle**)
  - **Satisfiability with at most 3 literals per clause** (equivalent to 3-SAT)
    - **Chromatic number** (also called the Graph Coloring Problem)
      - **Clique cover**
      - **Exact cover**
        - **Hitting set**
        - **Steiner tree**
        - **3-dimensional matching**
        - **Knapsack** (Karp's definition of Knapsack is closer to Subset sum)
          - **Job sequencing**
          - **Partition**
            - **Max cut**

Yhteenveto

- Laskennallisessa vaativuus teoriassa algoritmi on nopea jos jokaisella n-kokoisella syötteellä sen ajoaika on $O(n^k)$
- On olemassa laskennallisia ongelmia joita ei voida ratkaista nopeasti, mutta jotka voidaan ratkaista eksponentiaalisessa ajassa
- Tuhansia käytännöllisiä laskennallisia ongelmia joihin ei uskota löytyvän nopeita algoritmeja niiden ratkaisemiseen, mutta ei tätä ei olla pystytty vielä todistamaan (P vs NP)

Kiitos! :)