

8.2 Luokat L ja NL

- Aikavaativuusanalyysissä ei ole järkevää tarkastella *alilineaarisia* vaativuusluokkia, koska koko syötettä ei kyetä lukemaan alle lineaarisessa ajassa
- Sen sijaan koko syötettä ei välttämättä tarvitse tallettaa, joten on mielekästä tarkastella alilineaarisia tilavaativuusluokkia
- Laskentamallia tulee toki muuttaa
- Olkoon Turingin koneella työnauhan lisäksi erillinen syötenauha, jolle ei voida kirjoittaa mitään
- Vain varsinaiselta työnauhalla käytetty tila lasketaan koneen vaatimaksi tilaksi

- Luokka **L** on deterministisellä Turingin koneella logaritmisessa tilassa ratkaistavien kielten luokka

$$L = \text{DSPACE}(\log n)$$
- **NL** on vastaava luokka epädeterministisille koneille
- Aiemmin tarkastelimme kielen $\{0^k 1^k \mid k \geq 0\}$ tunnistavaa Turingin konetta, joka vaatii lineaarisen tilan
- Annetun merkkijonon kuulumisen kieleen voidaan ratkaista jo logaritmisessa tilassa
- Riittää laskea nollien lukumäärä ja ykkösten lukumäärä (sekä tehdä tarpeelliset tarkistukset). Lukumäärän binääriesitys on korkeintaan logaritminen syöteen pituuden suhteen



- Aiemmin osoitimme, että ongelma PATH: "Onko G :ssä suunnattu polku s :stä t :hen?" kuuluu P :hen
- Aiempi algoritmi vaatii pahimmassa tapauksessa lineaarisen tilan
- Determinististä logaritmisessa tilassa toimivaa algoritmia ei tunneta ongelmalle PATH
- Sen sijaan epädeterministinen logaritmisesta tilasta vaativa algoritmi on olemassa
- Lähtien solmusta s toista korkeintaan m askelta
 - Jos nykyisolmu on t , niin hyväksy
 - Talleta tieto nykyisestä solmusta työhauhalle (log-tila)
 - Epädeterministisesti valitse yksi seuraajista



- Pienillä tilavaativuuksilla $f(n)$ suhde aikavaativuuteen $2^{O(f(n))}$ ei enää välttämättä päde
- Esimerkiksi vakiotilan $O(1)$ vaativa Turingin kone voi vaatia lineaarisen $O(n)$ askelten lukumäärän
- Näissä tapauksissa aikavaativuudella on asymptoottinen yläraja $n2^{O(f(n))}$
- Kun $f(n) \geq \log n$, niin $n2^{O(f(n))} = 2^{O(f(n))}$
- Myös Savitchin lause pätee sellaisenaan kun $f(n) \geq \log n$
- PATH siis kuuluu NL :ään, muttei luultavasti L :ään
- Itse asiassa ei tunneta yhtään luokan NL ongelmaa, jonka voitaisiin todistaa olevan L :n ulkopuolella



- $L \stackrel{?}{=} NL$ on siis vastaava kysymys kuin $P \stackrel{?}{=} NP$
- Voidaan määritellä NL -täydelliset kielet, luokan NL kaikkein vaikeimpina kielinä
- Polynomisen palautus ei kuitenkaan kelpaa määrittelyyn, koska kaikki luokan NL ongelmat ovat polynomisessa ajassa ratkeavia
- Täten kaikki muut ongelmat paitsi \emptyset ja Σ^* ovat polynomisesti toisiinsa palautettavissa
- Sen sijaan käytetään logaritmisesta tilasta palautuvuutta \leq_L
- Funktio on laskettavissa logaritmisessa tilassa, jos sen arvon laskeva Turingin kone käyttää työnauhallaan vain $O(\log n)$ tilan



Lause 8.23 Jos $A \leq_L B$ ja $B \in L$, niin $A \in L$. \square

Korollaari 8.24 Jos jokin NL -täydellinen kieli on L :ssä, niin $L = NL$. \square

Lause 8.25 $PATH$ on NL -täydellinen kieli. \square

Korollaari 8.26 $NL \subseteq P$.

Todistus. Lauseen 8.25 perusteella mille tahansa luokan NL kielelle A pätee $A \leq_L PATH$. Tilan $f(n)$ vaativa Turingin kone toimii ajassa $n2^{O(f(n))}$, joten logaritmisessa tilassa toimiva palautusfunktion laskeva kone toimii myös polynomisessa ajassa.

Koska $A \leq_m^p PATH$ ja $PATH \in P$, niin lauseen 7.14 perusteella myös $A \in P$. \square

9. Approksimointi

- Tarkastellaan ongelmaa, jossa on annettu
 - Perusjoukko U , jossa on m alkioita
 - Perusjoukon osajoukkojen kokoelma $S = \{S_1, \dots, S_n\}$ s.e. $US = U$
- Tavoitteena on löytää *osapeite* $S' \subseteq S$,
 $US' = U$,
jossa on mahdollisimman vähän osajoukkoja
- Tämä ongelma on **minimi joukkopeite** (Minimum Set Cover, minSC)
- Yksi vanhimmista ja eniten tutkituista kombinatorisista optimointiongelmistä

- Ongelman päätösversio
 - Annettuna: perusjoukko U , peite S ja kokonaisluku k
 - Kysymys: onko U :lla osapeitettä $S' \subseteq S$ s.e. $|S'| \leq k$?

Lause 9.1 *Minimi joukkopeite -ongelman päätösversio on NP-täydellinen.*

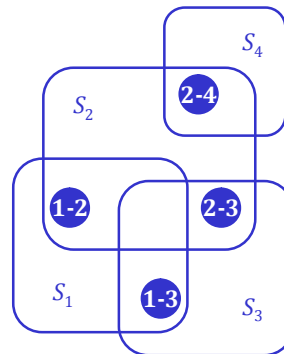
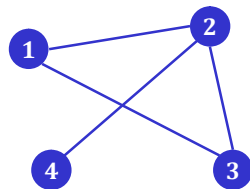
Todistus. Selvästi $\text{minSC} \in \text{NP}$: Arvataan annetusta peitteestä S k :n osajoukon osapeite S' ja tarkistetaan deterministisesti polynomisessa ajassa ratkaisu.

Polynominen palautus $VC \leq_m^p \text{minSC}$ on helppo muodostaa. Olk. $\langle G, k \rangle$ solmupeiteongelman tapaus, missä $G = (V, E)$. Valitaan kuvaus f :

$$f(\langle (V, E), k \rangle) = \langle E, V_B, k \rangle,$$

missä V_B on verkon G solmuihin liittyvien kaarten muodostama kokoelma. Siis jokaista $v \in V$ vastaa joukko $\{e \in E \mid e = (v, w)\}$.

Selvästi f on polynomisessa ajassa laskettavissa ja on palautus. \square



- minSC siis on laskennallisesti vaativa ongelma, emme tunne sille polynomi aikaista ratkaisualgoritmia
- Pyrimme löytämään polynomisessa ajassa toimivan algoritmin,
 - jonka tuottama ratkaisu ei välttämättä ole paras mahdollinen (optimaalinen), mutta
 - jonka voidaan osoittaa aina olevan korkeintaan syötteen pituudesta riippuvan funktion verran optimaalista ratkaisua huonompi
- Tällaista algoritmia kutsutaan **aprosimointialgoritmiksi**
- Merkitään, että **Opt** on optimaalisen ja **App** aproksimointi-algoritmin tuottaman ratkaisun kustannus

- Koska minSC on minimointiongelma, niin $App/Opt \geq 1$
- Mitä lähempänä arvoa 1 tämä suhdeluku on, sen paremmin tuotettu ratkaisu aproksimoi optimaalista ratkaisua
- Aproksimointialgoritmit siis vaaditaan, että osamäärää rajoittaa syötteen pituuden n funktio

$$\frac{App}{Opt} \leq \rho(n)$$

- $\rho(n)$ on algoritmin **aprosimointisuhde**
 - Algoritmia nimitetään $\rho(n)$ -aprosimointialgoritmiksi
- Parhaimmillaan aproksimointisuhde ei riipu laisinkaan syötteen pituudesta n , vaan on vakio

- Tarkastellaan seuraavaa algoritmia solmupeiteongelmalle
- Osoitamme, että se on 2-aproksimoinalgoritmi ongelmalle

Syöte: Suuntaamaton verkko $G = (V, E)$

Tuloste: Solmupeite C

1. $C \leftarrow \emptyset$;

2. $E' \leftarrow E$;

3. **while** $E' \neq \emptyset$ **do**

a. olkoon (u, v) mv. joukon E' kaari;

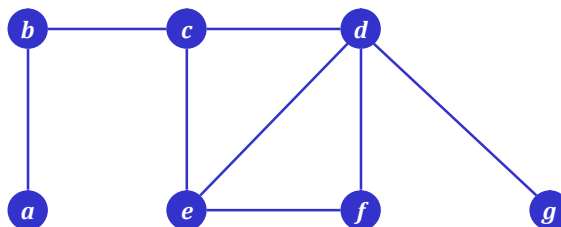
b. $C \leftarrow C \cup \{u, v\}$;

c. Poista E' :stä kaikki solmuihin u ja v liittyvät kaaret;

4. **od**;

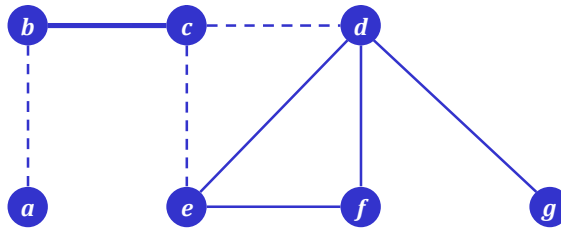
5. **return** C ;

Ensimmäinen satunnainen kaaren valinta: (b, c)

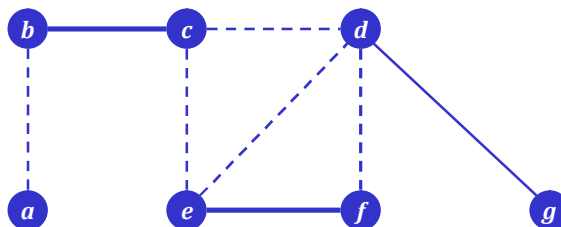




Poistetaan solmuihin b ja c liittyvät muut kaaret

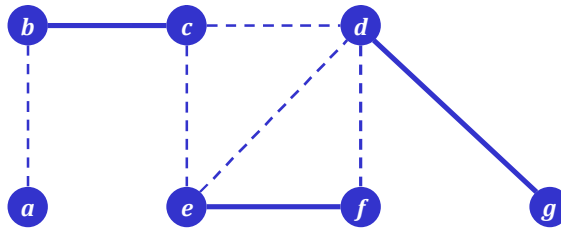


Seuraava satunnainen valinta: (e, f) ja sen solmuihin liittyvien muiden kaarten poisto



Valittavana on enää (d, g)

Saadaan siis 6:n solmun peite,
kun optimaalisessa on 3 solmua (esim. b, d, e)



Lause 9.2 Edellinen algoritmi on polynomiainainen 2-
apksimointialgoritmi solmupeiteongelmalle.

Todistus. Algoritmin aikavaativuus, käyttäen vieruslistaesitystä verkolle, on $O(V + E)$, joten se on polynomiainainen. On selvää, että algoritmin palauttama solmujoukko C on solmupeite Verkon G kaarille, koska solmuja lisätään C :hen rivin 3 silmukassa kunnes kaikki kaaret on peitetty.

Olkoon A algoritmin rivillä 3a valitsemien kaarten joukko. Sen sisältämien kaarten peittämiseksi missä tahansa solmupeitteessä — erityisesti myös optimaalisessa solmupeitteessä — on oltava ainakin kunkin joukon A kaaren toinen pää.

Koska joukon A sisältämien kaarten päätepisteet ovat algoritmin toiminnan perusteella toisistaan poikkeavat, niin $|A|$ on alaraja minkä tahansa solmupeitteen koolle.

Erityisesti

$$\text{Opt} \geq |A|.$$

Toisaalta yo. algoritmi valitsee rivillä 3a aina kaaren, jonka kumpikaan pää ei vielä ole joukossa C . Täten

$$\text{App} = |C| = 2|A|.$$

Edelliset yhdistämällä, saamme

$$\text{App} = 2|A| \leq 2 \text{Opt},$$

joten

$$\text{App}/\text{Opt} \leq 2.$$

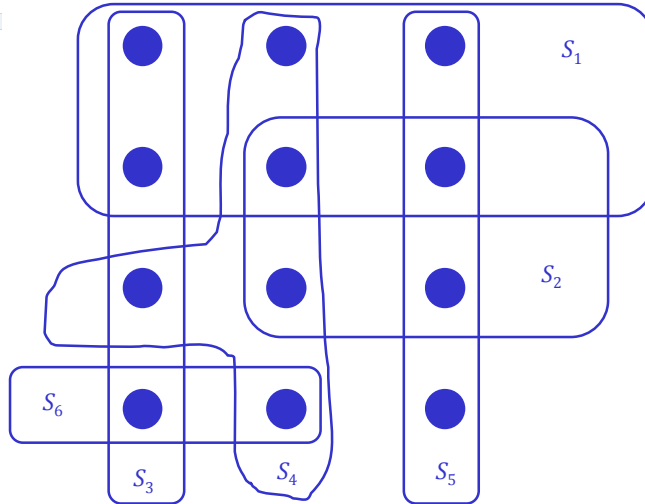
□

- Myös joukkopeiteongelmalla on yksinkertainen ahne approksimointialgoritmi
- Tällä eikä millään muullakaan polynomiaikaisella deterministisellä algoritmilla ei kuitenkaan voida saavuttaa vakioapproksimoituvuutta

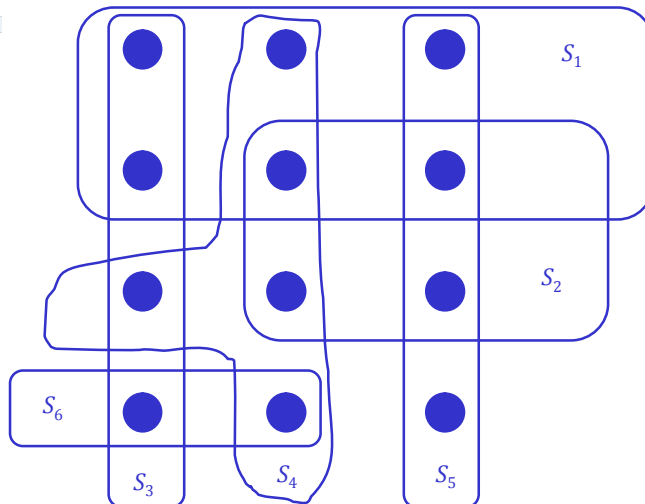
Syöte: Perusjoukko U ja sen peite S

Tuloste: Joukkopeite C

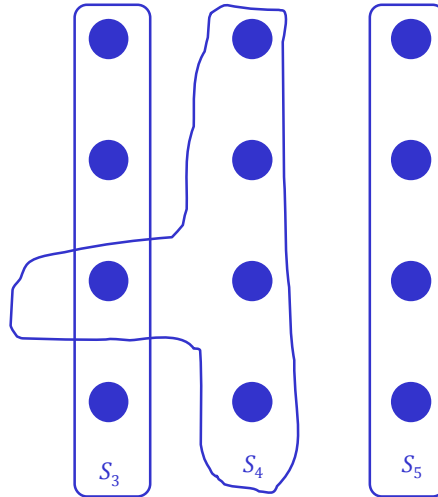
1. $X \leftarrow U; C \leftarrow \emptyset;$
2. **while** $X \neq \emptyset$ **do**
 - a. valitse $S \in S$ s.e. $|S' \cap X|$ maksimoituu;
 - b. $X \leftarrow X \setminus S;$
 - c. $C \leftarrow C \cup \{S\};$
3. **od;**
4. **return** $C;$



Ahne: 4 osajoukkoa



Optimaalinen: 3 osajoukkoa



- Ahne algoritmi on helppo toteuttaa syötteen pituuden $|U|$ ja $|S|$ suhteen polynomisessa ajassa
 - Rivin 2 silmukkaa suoritetaan korkeintaan $\min(|U|, |S|)$ kertaa ja silmukan rungon suoritus on helposti toteutettavissa ajassa $O(|U| \cdot |S|)$
 - Kaikkiaan siis vaativuus on $O(|U| \cdot |S| \min(|U|, |S|))$
 - Myös lineaariaikainen toteutus on mahdollinen
- Algoritmin palauttama kokoelma C on selvästi joukkopeite, koska rivin 2 silmukkaa suoritetaan kunnes peittämättömiä alkioita ei enää ole

- Ahneen algoritmin palauttaman joukkopeitteen kustannuksen suhtauttamiseksi, asetetaan kullekin valitulle joukolle kustannus 1
- Olkoon S_i ahneen algoritmin i :ntenä valitsema joukko
- S_i :n kustannus jaetaan tasan kaikkien siihen sisältyvien alkioiden kesken, jotka tulevat nyt ensi kerran peitettyä
- Merk. c_u on alkiolle $u \in U$ laskutettu kustannus
- Kullekin alkiolle laskutetaan kustannusta vain kerran, kun se ensi kerran peitetään
- Jos u tulee ensi kerran peitettyä joukolla S_i , niin sille laskutettu kustannus on

$$c_u = \frac{1}{|S_i \setminus (S_1 \cup S_2 \cup \dots \cup S_{i-1})|}$$

- Kullekin ahneen algoritmin valitsemalle joukolle annetaan kustannus 1, joten

$$\text{App} = |C| = \sum_{u \in U} c_u$$

- Optimaalisen peitteen C^* kustannus puolestaan on

$$\sum_{S' \in C^*} \sum_{u \in S'} c_u$$

- Koska jokainen $u \in U$ kuuluu vähintään yhteen $S' \in C^*$, niin

$$\sum_{S' \in C^*} \sum_{u \in S'} c_u \geq \sum_{u \in U} c_u$$

- Edelliset yhdistämällä, seuraa

$$\text{App} \leq \sum_{S' \in C^*} \sum_{u \in S'} c_u$$

- Merkitään $H(k)$:lla k :ttä harmonista lukua

$$H(k) = \sum_{j=1}^k \frac{1}{j} = 1 + \frac{1}{2} + \dots + \frac{1}{k}$$

- Määritellään $H(0) = 0$
- Seuraavaksi osoitamme, että mille tahansa $S' \in S$ pätee

$$\sum_{u \in S'} c_u \leq H(|S'|)$$

- Edellisen epäyhtälön perusteella tällöin

$$\begin{aligned} \text{App} &\leq \sum_{S' \in C^*} H(|S'|) \\ &\leq |C^*| \cdot H(\max\{|S'|: S' \in S\}) \\ &\text{Opt} \cdot H(\max\{|S'|: S' \in S\}) \end{aligned}$$

Lemma 9.3 *Kaikilla $S' \in S$ pätee*

$$\sum_{u \in S'} c_u \leq H(|S'|)$$

Todistus. Olkoon $S' \in S$ mv. ja $i = 1, 2, \dots, |C|$. Olkoon edelleen

$$n_i = |S' \setminus (S_1 \cup S_2 \cup \dots \cup S_i)|$$

niiden S' :n alkioden lukumäärä, joita ei ole vielä peitetty kun ahne algoritmi on valinnut joukot S_1, S_2, \dots, S_i peitteeseen.

Asetetaan $n_0 = |S'|$.

Olkoon k pienin indeksi s.e. $n_k = 0$, eli jokainen S' :n alkiu kuuluu vähintään yhteen joukoista S_1, S_2, \dots, S_k .

Tällöin $n_{i-1} \geq n_i$ ja S_i , $i = 1, 2, \dots, k$, peittää ensimmäisen kerran $n_{i-1} - n_i$ alkiota.

Nyt

$$\sum_{u \in S'} c_u = \sum_{i=1}^k (n_{i-1} - n_i) \frac{1}{|S_i \setminus (S_1 \cup \dots \cup S_{i-1})|}.$$

Koska S_i on ahneesti valittu joukko, niin se peittää vähintään niin monta alkioa kuin joukko S' (tai muutoin S' olisi pitänyt valita). Näin ollen

$$|S_i \setminus (S_1 \cup \dots \cup S_{i-1})| \geq |S' \setminus (S_1 \cup \dots \cup S_{i-1})| = n_{i-1}$$

jonka perusteella

$$\sum_{u \in S'} c_u \leq \sum_{i=1}^k (n_{i-1} - n_i) \frac{1}{n_{i-1}}.$$

$$\begin{aligned} \sum_{u \in S'} c_u &\leq \sum_{i=1}^k (n_{i-1} - n_i) \frac{1}{n_{i-1}} \\ &= \sum_{i=1}^k \sum_{j=n_i+1}^{n_{i-1}} \frac{1}{n_{i-1}} \\ &\leq \sum_{i=1}^k \sum_{j=n_i+1}^{n_{i-1}} \frac{1}{j}, \end{aligned}$$

koska $j \leq n_{i-1}$. Edelleen

$$\begin{aligned} &= \sum_{i=1}^k \left(\sum_{j=1}^{n_{i-1}} \frac{1}{j} - \sum_{j=1}^{n_i} \frac{1}{j} \right) \\ &= \sum_{i=1}^k (H(n_{i-1}) - H(n_i)) \\ &= H(n_0) - H(n_k), \end{aligned}$$

sillä summan muut termit kumoavat toisensa.

Koska valinnan perusteella $n_k = 0$ ja $H(0) = 0$, niin edelleen

$$\begin{aligned}
 &= H(n_0) - H(0) \\
 &= H(n_0) \\
 &= H(|S'|)
 \end{aligned}$$

joten olemme todistaneet lemmän. \square

- Harmoniselle luvulle $H(k)$ pätee $\ln k < H(k) \leq \ln k + 1$
- Täten edellä olleista seuraa:

Lause 9.4 *Joukkopeiteongelman ahneelle algoritmille pätee*

$$\frac{\text{App}}{\text{Opt}} \leq H(\max\{|S'| : S' \in S\}) \leq \ln |U| + 1$$

- Joissain sovelluksissa $\max\{|S'| : S' \in S\}$ on pieni vakio
- Tällöin ahneen algoritmin antama vastaus on vain pienen vakion päässä optimaalisesta
- Erityisesti, jos osajoukoilla S' on yläraja d koolle, niin $\text{App}/\text{Opt} \leq H(d)$
- Esimerkiksi kun solmupeite-ongelmassa solmujen aste on korkeintaan 3, niin
 - ahneen joukkopeite -algoritmin palauttama vastaus ei ole kuin korkeintaan $H(3) = 11/6 < 2$ kertaa niin suuri kuin optimaalinen peite



- Feige, 1996: yksikään polynomiainen algoritmi ei voi approksimoida minSC-ongelmaa tarkkuudella $(1-\varepsilon)\ln m$, millä tahansa $\varepsilon > 0$, jollei $NP \subseteq DTIME(n^{\log \log n})$
- Täten ahnetta algoritmia oleellisesti parempaa approksimointialgoritmia ei voida löytää
- Slavík, 1996: Ahneen algoritmin approksimointisuhteen tarkempi yläraja on:
$$\ln m - \ln \ln m + \Theta(1)$$
- Itse asiassa tämä on myös alaraja ahneen algoritmin approksimointisuhteelle
- Asymptoottisesti $\ln m - \ln \ln m + \Theta(1)$ on siis ahneen algoritmin täsmällinen approksimointisuhte