ALGORITMUSOK ÉS BONYOLULTSÁGELMÉLET

Matematika MSc hallgatók számára

9. Előadás

Előadó: Hajnal Péter
Jegyzetelő: Bogya Norbert 2011. Április 12.

Emlékeztető. Legyen $\vec{G}$ egy irányított gráf, a és z pedig ennek két csúcsa.

Az $\overrightarrow{\text{ELÉRHETŐSÉG}}$ nyelv azokat a $(\vec{G}, a, z)$ hármasokat tartalmazza, amelyekre teljesül, hogy létezik $\vec{az}$ irányított út $\vec{G}$-ben. Formálisan:

$$\omega = \ulcorner(\vec{G}, a, z)\urcorner \in \overrightarrow{\text{ELÉRHETŐSÉG}} \iff \exists \vec{az} \text{ irányított út } \vec{G}\text{-ben.}$$

$\overrightarrow{\text{ELÉRHETŐSÉG}} \in \mathcal{NL}$.

1. Immerman-Szelepcsényi tétel

1. Tétel (Immerman (1988)—Szelepcsényi (1987)).

$$\overrightarrow{ELÉRHETŐSÉG} \in co\ \mathcal{NL}.$$

Megjegyzés. A tétel kifejtve a következőket jelenti. Ha egy rögzített kódolás mellett az input $\omega = \ulcorner(\vec{G}, a, z)\urcorner$, akkor létezik olyan logaritmikus tárigényű nemdeterminisztikus T Turing-gép, amelynek

- ha nem létezik $\vec{az}$ irányított út, akkor van elfogadó futása,
- ha létezik $\vec{az}$ irányított út, akkor minden futása elvető.

Észrevétel. Neil Immerman és Róbert Szelepcsényi egymástól függetlenül igazolták a tételt 1987-ben, Szelepcsényi mindössze 20 éves volt ekkor. Az közismert bonyolultsági osztályok gyorsan kialakultak az 50-es, 60-as évek környékén, így ez a tétel viszonylag későinek mondható. A tétel — a bizonyításával együtt — egyszerűsége és fontossága miatt ma már standard tananyagnak számít.

Bizonyítás. A bizonyítás során mindig feltesszük, hogy van egy ω input, ami kódol egy $\vec{G}$ gráfot. Vezessük be a következő jelöléseket:

$$\begin{aligned} N_i &= \left\{ v \in V(\vec{G}) : a \vec{\sim}_{\leq i} v \right\}, \\ n_i &= |N_i|. \end{aligned}$$

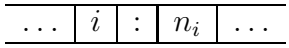
Tehát N_i jelenti azoknak a csúcsoknak a halmazát, melyek a-ból legfeljebb i lépésben irányítottan elérhetők, n_i pedig az N_i számosságát jelöli. Érdemes észrevenni, hogy $N_0 = \{a\}$, $N_1 = \{a\} \cup \{a \text{ ki-szomszédai}\}$, $n_0 = 1$ illetve $n_i \leq \left|V(\vec{G})\right|$.

A bizonyítás során az lesz a célunk, hogy az $n_0, n_1, n_2, \ldots$ értékeket kiszámoljuk. Ugyanis az ELFOGAD állapot ekvivalens azzal, hogy valamely $i \leq |V(G)|$ esetén

n_i megegyezik n_{i+1}-gyel és $z \notin N_i$. Ez azzal indokolható, hogy ha valamely i-re $n_i = n_{i+1}$, akkor az egyenlőség az $(i+1)$-nél nagyobb indexekre is fennáll, így az N_i az összes a-ból elérhető csúcsot felsorolja, és csak azt kell ellenőrizni, hogy z benne van-e N_i-ben. A bizonyítás során egy nemdeterminisztikus rekurzív algoritmust fogunk felírni. A rekurzió feltétele, hogy ismerjük n_i-t. Az algoritmus segítségével n_i-ből meghatározzuk n_{i+1}-et, hogy a fenti célunkat elérjük.

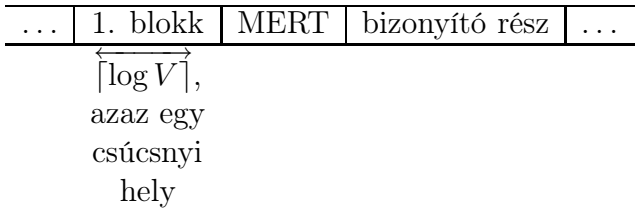
A bizonyítás során szükségünk lesz négy darab munkaszalagtöredékre. Gondolhatunk rájuk úgy is, mint ugyanannak a munkaszalagnak a különböző, de jól meghatározott része, vagy úgy is, mint külön munkaszalagok (ekkor többszalagos Turing-gép modellel dolgozunk).

(1)



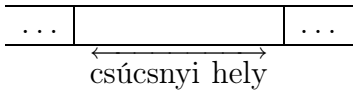
Az (1) munkaszalagtöredéken tároljuk a megfelelő indexekhez tartozó n_i-ket, és biztosnak kell lennünk abban, hogy $n_i = |N_i|$. A következő (2)-es munkaszalag(töredék) az N_i-ket sorolja föl.

(2)

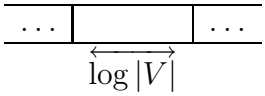


A (3) munkaszalag(töredék) a V tesztelésére szolgál, egy V-beli csúcs N_{i+1}-hez tartozását vizsgálja, a (4) pedig az n_{i+1}-et számolja, a (3) tartalma szerint.

(3)



(4)



A (1)-es szalagon kezdetben 0 : 1 van, a többi szalag pedig üres. A bizonyítást adó pontos algoritmus a következő:

1. Az algoritmusnak ez a része egy nemdeterminisztikus fázis, amire később úgy is lehet majd gondolni, mint egy szubrutin. Felsoroljuk az N_i elemeit, azaz a (2) szalagon a $v_1, v_2, \ldots, v_{n_i}$ csúcsok sorolódnak fel az 1. blokkban. Természetesen oly módon, hogy v_1 leíródik, azt v_2 felülírja, és ez így megy tovább, míg végül v_{n_i} lesz ráírva.

   - Azért, hogy elkerüljük, hogy egy csúcsot többször is felsoroljunk, feltesszük, hogy a csúcsok kódjai az indexekkel növekednek, azaz $\ulcorner v_1 \urcorner < \ulcorner v_2 \urcorner < \ldots$
   - Mielőtt az 1. blokkban felsorolnánk a következő csúcsot, a (2)-es bizonyító részében minden v_j-re legyártunk egy bizonyítást arra, hogy $v_j \in N_i$. Mivel nemdeterminisztikus az algoritmus, ezért ha $v_j \in N_i$, akkor van olyan tanúszalag-tartalom, ami ezt bizonyítja, és ezt logtárral le tudjuk ellenőrizni a (2)-es szalag bizonyító részében, mivel a probléma $\mathcal{NL}$-be tartozását tudjuk. A bizonyító rész tartalma a következőképpen alakul:

	a	a^+	számláló++	$\rightarrow$ 1. lépés
	a^+	a^{++}	számláló++	$\rightarrow$ 2. lépés
	$\vdots$			
átlagos szalagrész-tartalom:	x	y	számláló++	$\rightarrow k$. lépés
	$\vdots$			
		v_j	számláló	$\overset{?}{\leq} i$

Kezdetben a-ból átlépünk egy szomszédos a^+ csúcsba, majd növeljük a számlálót. Ezután az a felülíródik a szomszédjával (a^+-szal) és most annak egy szomszédja lesz a második csúcs (a^{++}), ami pedig az a^+ csúcsot írja felül. Ez a művelet addig ismétlődik, míg a második csúcs v_j nem lesz vagy a számláló túlcsordul (nagyobb lesz, mint i). Két állapot léphet fel: STIMMEL, illetve NEM-STIMMEL. A STIMMEL állapotban elérjük v_j-t a számláló túlcsordulása nélkül. A komplementer NEM-STIMMEL állapot ekvivalens azzal, hogy elvetjük az egész futást.

Ha egy jogkövető futásról van szó (speciálisan ha ismerjük n_i-t), akkor az N_i elemeit tényleg fel tudjuk sorolni. Ha végigfutunk a NEM-STIMMEL állapot elkerülésével, akkor biztosak lehetünk abban, hogy N_i elemeit soroltuk fel. (Pontosabban fogalmazva csak n_i darab különböző elemet soroltunk fel N_i-ből, de mivel n_i-ben megbízunk, így N_i összes elemét felsoroltuk.)

2. Az algoritmus következő fázisa a V tesztelése, azaz a (3)-as szalagon felsoroljuk V összes elemét, és ellenőrizzük, hogy benne vannak-e N_{i+1}-ben. Legyenek az $u_1, \ldots, u_n$ csúcsok a V elemei. Kezdetben a (4) szalag tartalma 0.

   - Minden u_j csúcs esetében meghívjuk a fenti szubrutint, azaz nemdeterminisztikusan felsoroljuk az N_i elemeit.
   - Azt teszteljük, hogy az aktuális u_j az fel van-e sorolva N_i-ben, vagy kiszomszédja egy olyan csúcsnak, ami fel van sorolva. Ha az N_i felsorolása NEM-STIMMEL állapot nélkül végigfut, akkor pontosan N_{i+1} elemei mennek át a teszten, és ezt mondja ki a következő lemma.

     2. Lemma. *$x \in N_{i+1} \iff x \in N_i$ vagy alkalmas $y \in N_i$ esetén: $\vec{yx} \in E$.*

     Az előző lemma tulajdonképpen egy trivialitás, a bizonyításához elég egyszerűen meggondolni azt, hogy mit jelent.

     Ha V egy eleme átmegy az előző teszten, akkor a (4)-es szalag(töredéken) növeljük eggyel a számlálót.

     3. Lemma. *Ha V összes elemét leteszteltük úgy, hogy a NEM-STIMMEL állapotot elkerültük, akkor a* (4)*-es szalag tartalma biztosan n_{i+1}.*

     Az előző lemma biztosítja a rekurziós lépést, és így a rekurzív lépés leírásának vége van.

3. Már csak annak az ellenőrzése van hátra, hogy n_i megegyezik-e n_{i+1}-gyel.

- Ha nem, akkor az (1)-es szalagon i-t kicseréljük $(i+1)$-re és n_i-t kicseréljük a (4)-es szalagon lévő számra, ami az előző lemma szerint pontosan az n_{i+1}, és visszatérünk az algoritmus 1. részéhez.
- Ha igen, akkor újból felsoroljuk N_i elemeit, és csak azt vizsgáljuk, hogy z előjön-e közöttük. Ha nincs NEM-STIMMEL állapot a felsorolás során és a z sem jön elő, akkor tudjuk, hogy nem létezik $\vec{az}$ út, így ELFOGAD állapottal leállunk.

Az algoritmus polinomtárat használ fel, a helyes működése a fentiekben bizonyítva lett, ezért az Immerman-Szelepcsényi tétel bizonyítása kész. ■

4. Következmény. $\mathcal{NL} = co\,\mathcal{NL}$

Bizonyítás. Ha L egy $\mathcal{NL}$-beli nyelv, akkor létezik egy $L \prec_{\mathcal{L}} \overrightarrow{\text{ELÉRHETŐSÉG}}$ visszavezetés (mivel az $\overrightarrow{\text{ELÉRHETŐSÉG}}$ $\mathcal{NL}$-teljes), így tudjuk, hogy az L nyelv $co\,\mathcal{NL}$-ben is benne van. Az $L \in co\,\mathcal{NL}$ pontosan azt jelenti, hogy L komplementere benne van $\mathcal{NL}$-ben, azaz $\overline{L} \in \mathcal{NL}$. Ezt a gondolatmenetet megismételve $\overline{L}$-re, azt kapjuk, hogy az $\overline{L} \in \mathcal{NL}$-ből következik, hogy $L \in \mathcal{NL}$. Ebből a két megállapításból pedig azt kapjuk, hogy L akkor és csak akkor van $\mathcal{NL}$-ben, ha a komplementere is ott van, és ez azt jelenti, hogy $\mathcal{NL} = co\,\mathcal{NL}$. ■

Ahogy a bizonyít'as is mutatta a tétel lényege, hogy $\mathcal{NL}$ zárt a komplementálásra nézve.

5. Következmény. *Ha $s(n)$ szép tárfüggvény, akkor*

$$\mathcal{NSPACE}(\mathcal{O}(s(n))) = co\,\mathcal{NSPACE}(\mathcal{O}(s(n))).$$

Bizonyítás. A bizonyítás hasonlóan történhet, mint az Immerman-Szelepcsényi tétel bizonyítása, csak itt a blokkok nagyobbak. ■

Megjegyzés. Az eredeti tétel, illetve a két következmény közül mindegyiket szokták Immerman-Szelepcsényi-tételnek nevezni.

6. Tétel (Omer Reingold, 2004). *ELÉRHETŐSÉG* $\in \mathcal{L}$.

Attól függetlenül, hogy klasszikus szélességi és mélységi keresések a problémát $\mathcal{P}$-be rakják, ez egy nagyon nehéz tétel. Omer Reingold bizonyította 2004-ben. Maga az algoritmus nem túl használható, de bonyolultságelméleti szempontból nagy jelentőséggel bír. Viszont az $\overrightarrow{\text{ELÉRHETŐSÉG}}$ és az $\mathcal{L}$ osztály viszonyáról még semmit sem tudunk. Az $\overrightarrow{\text{ELÉRHETŐSÉG}} \in \mathcal{L}$ bizonyítása azt jelentené, hogy $\mathcal{L} = \mathcal{NL}$.

2. $\mathcal{P}$ és $\mathcal{PSPACE}$ közötti problémák

Röviden tekintsük át, hogy jelenleg mit tudunk mondani a különböző bonyolultsági osztályok egymás közötti viszonyairól.

$$\mathcal{L} \subseteq \mathcal{NL} = co\,\mathcal{NL} \subseteq \begin{array}{|ccc|} \hline & \mathcal{NP} & \mathcal{NPSPACE} \\ \subsetneq & \subsetneq & \| \\ \mathcal{P} & & \mathcal{PSPACE} \\ \subsetneq & \subsetneq & \| \\ & co\,\mathcal{NP} & co\,\mathcal{NPSPACE} \\ \hline \end{array} \subseteq \mathcal{EXP}$$

Az ábrának nyilván a bekeretezett része a legérdekesebb és ennél többet a megjelölt osztályokról nem tudunk. Még azt sem tekinthetjük kizártnak, hogy $\mathcal{P} = \mathcal{PSPACE}$.

Emlékeztető. Akkor mondjuk, hogy $L \in \mathcal{NP}$, ha létezik olyan tanúszalagos (nemdeterminisztikus) T Turing-gép, ami úgy működik, hogy pontosan az L-beli ω szavakhoz létezik τ tanúszalag-tartalom úgy, hogy $T(\omega, \tau)$ ELFOGAD állapottal és ω hosszában polinomiális időben leáll. Formálisan:

$$\begin{array}{rcl} L \in \mathcal{NP} & \iff & \text{van olyan } T \text{ nem-determinisztikus Turing-gép, hogy} \\ & & T \text{ polinomiális } |\omega|\text{-ban és} \\ & & \omega \in L \quad \Leftrightarrow \quad \exists \tau : \ \ T(\omega, \tau) \text{ elfogadó} \end{array}$$

Definíció. Akkor mondjuk, hogy $L \in co\,\mathcal{NP}$, azaz $\overline{L} \in \mathcal{NP}$, ha létezik olyan tanúszalagos (nemdeterminisztikus) T Turing-gép, ami úgy működik, hogy pontosan az L-beli ω szavakhoz nem létezik τ tanúszalag-tartalom úgy, hogy $T(\omega, \tau)$ ELFOGAD állapottal áll le ω hosszában polinomiális időben. Formálisan:

$$\begin{array}{rcl} L \in co\,\mathcal{NP} & \iff & \text{van olyan } T \text{ nem-determinisztikus Turing-gép, hogy} \\ & & T \text{ polinomiális } |\omega|\text{-ban és} \\ & & \omega \in L \quad \Leftrightarrow \quad \forall \tau : \ \ T(\omega, \tau) \text{ elvető} \end{array}$$

2.1. Példák

A következőkben példákat mutatunk a fenti ábra bekeretezett részéből. Az első példához szükségünk lesz a konjuktív normálformákról tanultakra, ezért elevenítsük fel ezeket.

Definíció. A φ formula konjuktív normálforma (CNF), ha előáll

$$\varphi = \bigwedge_{i=1}^{l} C_i$$

alakban, ahol minden C_i klóz literálok diszjunkciója, azaz

$$C_i = \bigvee_{j=1}^{l_i} l_j^{(i)}.$$

A φ CNF felfogható úgy is, mint egy $\{C_i\}$ klózhalmaz, és minden klóz értelmezhető úgy, mint egy $\{l_i\}$ literálhalmaz. Ez utóbbi meggondolás alapján a φ CNF hosszán a $|\varphi| = \Sigma_i\,|C_i|$ számot értjük. Például, ha $\varphi = (x \vee y \vee \neg t) \wedge (\neg x \vee z) \wedge (z \vee \neg w \vee \neg u) \wedge u$, akkor $|\varphi| = 3 + 2 + 3 + 1 = 9$.

Példa. Legyen OPT-CNF az a probléma, hogy egy φ CNF-fel kapcsolatban azt akarjuk igazolni, hogy a φ által leírt Boole-függvényt nem tudjuk $|\varphi|$-nál kisebb méretű CNF-fel megfogalmazni, azaz a φ a lehető legrövidebben írja le a Boole-függvényt.

- OPT-CNF $\in \mathcal{PSPACE}$

  Ennek a bizonyítására adható egy olyan naív algoritmus, mellyel megvizsgálunk minden szóba jöhető lehetőséget. Így tulajdonképpen exponenciálisan

sok lehetőséget írunk a munkaszalagra, viszont ezek felülírhatók, ezért elég a polinomtár. Csak azokból a változókból építkezünk, amik φ-ben is vannak, kiválasztunk bizonyos kisebb méretű literál-részhalmazokat, és az összes értékadásra teszteljük, és azt kell tapasztalnunk, hogy minden $|\varphi|$-nél kisebb méretű ψ CNF esetén van olyan kiértékelés, amire ψ és φ értéke különbözik.

- A probléma formalizált változata a következő:

$$\ulcorner\varphi\urcorner \in \text{OPT-CNF} \iff \forall \ulcorner\psi\urcorner \text{ CNF-hez } \exists \ulcorner\upsilon\urcorner \text{ kiértékelés úgy, hogy } (|\psi| < |\varphi| \Rightarrow \psi(\upsilon) \neq \varphi(\upsilon)) .$$

  Emlékezzünk vissza az fejezet elején felelevenített definíciókra. Ott $\exists\tau$ és $\forall\tau$ szerepel. Ez is hasonló probléma, csak itt két kvantorok szerepel/alternál.

- A probléma „valahol $\mathcal{P}$ és $\mathcal{PSPACE}$ között" található.

Példa. Legyen PONTOS_FGTLEN_CSÚCSOK az a probléma, hogy egy adott G gráfról és egy adott t számról azt akarjuk igazolni, hogy a G gráfban lévő független csúcsok maximális száma pontosan t, azaz

$$\text{PFC} := \text{PONTOS\_FGTLEN\_CSÚCSOK} = \{\ulcorner G, t\urcorner : \ \alpha(G) = t\} .$$

- A probléma formalizált változata a következő:

$$\ulcorner G, t\urcorner \in \text{PFC} \iff (\exists I \subseteq V(G) : \ I \text{ független és } t = |I|) \bigwedge (\forall I \subseteq V(G) : \ I \text{ független} \to |I| \leq t) .$$

  Azt vehetjük észre, hogy a konjukció bal oldalán álló probléma $\mathcal{NP}$-beli, a jobb oldalán álló pedig *co* $\mathcal{NP}$-beli. Mindkét kvantorra szükségünk van, továbbá

$$\text{PONTOS\_FGTLEN\_CSÚCSOK} \in \mathcal{PSPACE},$$

  viszont a PFC $\in \mathcal{NP}$ és PFC $\in$ *co* $\mathcal{NP}$ megállapítások közül egyiket sem mondhatjuk biztosnak. (A probléma ugyan $\mathcal{PSPACE}$-en belül van, viszont úgy érezzük, hogy $\mathcal{NP}$-n és *co* $\mathcal{NP}$-n kívül.)

Definíció. Legyen $\mathcal{H}$ egy halmazrendszer V felett, azaz $\mathcal{H} \subseteq \mathcal{P}(V)$. Bevezetjük a $\mathcal{H}$-nak az A-beli nyomát:

$$\text{Trace}_A\mathcal{H} = \{A \cap E : \ E \in \mathcal{H}\} , \text{ ahol } A \subseteq V.$$

Nyilván $\text{Trace}_A\mathcal{H} \subseteq \mathcal{P}(A)$. Az A ponthalmazt telítettnek nevezzük, ha $\text{Trace}_A\mathcal{H} = \mathcal{P}(A)$. Most már definiálhatjuk a Vapnyik—Cservonyenszki-dimenziót:

$$\text{VCs-dim}\mathcal{H} = \max\{|A| : \ A \text{ telített}\} .$$

Észrevétel. A Vapnyik—Cservonyenszki-dimenzió a matematika sok területén előfordul, például a geometriában, kombinatorikában, mesterséges intelligenciában, illetve a statisztikában is. A dimenzió névadói is statisztikusok voltak.

Példa. Legyen VCS-DIM az a nyelv, hogy egy adott $\mathcal{H}$ halmazrendszerről és egy adott k számról azt akarjuk igazolni, hogy $\mathcal{H}$ VCs-dimenziója legalább k, azaz

$$\text{VCS-DIM} = \{\ulcorner V, \mathcal{H}, k \urcorner : \text{ VCs-dim}\mathcal{H} \geq k\}\,.$$

- A halmazrendszereket tömör kódolással kódoljuk.
- A $\ulcorner V, \mathcal{H}, k \urcorner$ hármasban a $(V, \mathcal{H})$ pár tulajdonképpen felfogható egy $C_{\mathcal{H}}$ hálózatnak, ahol a $v \in V$ csúcsok $\log|V|$ biten kódolhatók, az $E \in \mathcal{H}$ élek pedig $\log|\mathcal{H}|$ biten, és $C_{\mathcal{H}}$ kiszámolja, hogy v eleme-e E-nek. Ilyen kódolásnál a véletlen séta generálása rendkívül gyorsan megy, még egy milliószor milliós szomszédségi mátrix-szal megadható gráf esetén is.
- A probléma formalizált változata a következő:

$$\ulcorner V, \mathcal{H}, k \urcorner \in \text{VCS-DIM} \Longleftrightarrow \exists A\, \forall R\, \exists E\, \left(|A| = k \bigwedge (R \subseteq A \Rightarrow R = A \cap E)\right).$$

- VCS-DIM $\in \mathcal{PSPACE}$.

3. További osztályok, polinomiális hierarchia

Definíció ($\Sigma_i\mathcal{P}$ nyelvosztály).

$$\begin{aligned} L \in \Sigma_i\mathcal{P} \iff\ & \exists T \text{ polinomiális Turing-gép, melyre } \exists\tau_1, \forall\mu_2, \exists\tau_3, \ldots, QX_i \text{ úgy,} \\ & \text{hogy } \omega \in L \Leftrightarrow T(\omega, \tau_1, \mu_2, \tau_3, \ldots, X_i) \text{ ELFOGADÓ,} \\ & \text{ahol } Q = \begin{cases} \exists, & \text{ha } i \text{ páratlan,} \\ \forall, & \text{ha } i \text{ páros.} \end{cases} \text{ és } X_i = \begin{cases} \mu_i, & \text{ha } i \text{ páros,} \\ \tau_i, & \text{ha } i \text{ páratlan,} \end{cases} \end{aligned}$$

Definíció $\Pi_i\mathcal{P}$ nyelvosztály.

$$\begin{aligned} L \in \Pi_i\mathcal{P} \iff\ & \exists T \text{ polinomiális Turing-gép, melyre } \forall\mu_1, \exists\tau_2, \forall\mu_3, \ldots, QY_i \text{ úgy,} \\ & \text{hogy } \omega \in L \Leftrightarrow T(\omega, \mu_1, \tau_2, \mu_3, \ldots, Y_i) \text{ ELFOGADÓ,} \\ & \text{ahol } Q = \begin{cases} \exists, & \text{ha } i \text{ páros,} \\ \forall, & \text{ha } i \text{ páratlan.} \end{cases} \text{ és } Y_i = \begin{cases} \mu_i, & \text{ha } i \text{ páratlan,} \\ \tau_i, & \text{ha } i \text{ páros,} \end{cases} \end{aligned}$$

Észrevétel.

- $\Pi_0\mathcal{P} = \Sigma_0\mathcal{P} = \mathcal{P}$.
- $\Sigma_1\mathcal{P} = \mathcal{NP}$.
- $\Pi_1\mathcal{P} = co\,\mathcal{NP}$.
- $\mathcal{P} = \Sigma_0\mathcal{P} = \Pi_0\mathcal{P} \subseteq \Sigma_1\mathcal{P},\ \Pi_1\mathcal{P} \subseteq \Sigma_2\mathcal{P} \cap \Pi_2\mathcal{P} \subseteq \Sigma_2\mathcal{P},\ \Pi_2\mathcal{P} \subseteq \Sigma_3\mathcal{P} \cap \Pi_3\mathcal{P} \subseteq \Sigma_3\mathcal{P},\ \Pi_3\mathcal{P} \subseteq \ldots \subseteq \Sigma_n\mathcal{P} \cap \Pi_n\mathcal{P} \subseteq \Sigma_n\mathcal{P},\ \Pi_n\mathcal{P} \subseteq \Sigma_{n+1}\mathcal{P} \cap \Pi_{n+1}\mathcal{P} \subseteq \ldots \subseteq \mathcal{PSPACE}$.

Definíció (Polinomiális hierarchia, $\mathcal{PH}$). $\mathcal{PH} = \bigcup_{i\in\mathbb{N}} \Pi_i\mathcal{P} = \bigcup_{i\in\mathbb{N}} \Sigma_i\mathcal{P}$.

Észrevétel. A definíció második egyenlősége tulajdonképpen egy állítás, aminek a bizonyítása egy egyszerű meggondolás, ahol azt kell belátni, hogy a jobb oldali felülről becsüli a bal oldalit, és fordítva.

Definíció (Alternáló polinomiális idő, $\mathcal{AP}$).

$$L \in \mathcal{AP} \iff \exists T \text{ Turing-gép, melyre } \exists\tau_1, \forall\mu_1, \exists\tau_2, \forall\mu_2, \ldots, \exists\tau_N, \forall\mu_N \text{ úgy,}$$
$$\text{hogy } \omega \in L \Leftrightarrow T(\omega, \tau_1, \mu_1, \ldots, \tau_N, \mu_N) \text{ ELFOGADÓ,}$$
$$T \text{ polinomiális } |\omega| \text{-ban.}$$

Megjegyzés. A gép polinomialitásából következik, hogy N is legfeljebb polinomiális. Lehetséges, hogy N függ $|\omega|$-tól. Ez egy többlet a polinomiális hierarchiához képest.

7. Tétel.

1. $\mathcal{PH} \subseteq \mathcal{AP}$.
2. $\mathcal{AP} = \mathcal{PSPACE}$.

Bizonyítás. (Vázlat)

A tétel első része triviális.

A második rész bizonyításához azt kell kihasználnunk, hogy QBF $\in \mathcal{PSPACE}$ és QBF $\in \mathcal{AP}$. Tudjuk, hogy a QBF az $PSPACE$-teljes és könnyen belátható, hogy a QBF nyelv $\mathcal{AP}$-teljes is. Korábban tanult dolgokat kell alkalmazni, például egy Turing gépet átírni Boole-formulává polinomidőben, megfelelő hálózatot gyártani, ahogy ezeket már korábban megcsináltuk. Továbbá, visszavezetések alkalmazásával kijön, hogy $\mathcal{AP} \subseteq \mathcal{PSPACE}$ és $\mathcal{AP} \supseteq \mathcal{PSPACE}$, és ebből már következik a köztük lévő egyenlőség. ■