

Nedeterministický Turingův stroj (NTM) je sekvence $(Q, \Sigma, \Gamma, S, q_0, B, F)$
 - od deterministického TM se liší pětihodovou fází:
 $S(Q \times F) \times \Gamma \rightarrow P_f(Q \times \Gamma \times \{L, R\})$, kde P_f je množina kropicí
 - NTM udělá krok, když $(q, a, L) \in \delta(p, b)$

Přijímatelný slov: $w \in \Sigma^*$ je přijímatelné, jestliže $q_0 w \xrightarrow{*} \alpha f \beta, \alpha \in F$

Jestliže slovo je přijímatelné, pakud existuje posloupnost kroků (větva výpočtu), která skončí v F .
 $L(M)$ je množina slov přijímatelných M

Rozchadování slov: M rozchaduje jazyk $L(M)$, jestliže ho přijímá a jestliže se všechny některé výpočty
 pro všechna vstupní slova sestavená po konečně mnoha krocích.

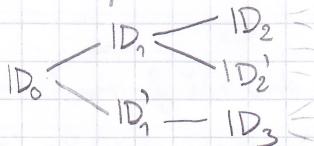
Časová složitost NTM: Maximální počet kroků, než se NTM sestaví. A je jedno, ve kterém větvě
 výpočtu, a jestli je to ta, ve které je správné řešení - konečný stav

Paměťová složitost NTM: Pocitelné zobrazení $S(n)$: největší rozdíl id použitých polí pásky,
 přes všechny délky vstupních slov $\leq n$ a všechny větve výpočtu. Pokud zábera nějaké rekoncovací
 polí, menej $S(n)$ definovatelné.

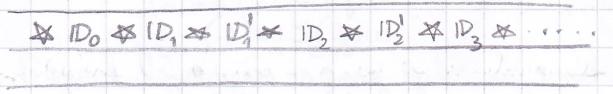
Veta: Pro každý NTM M existuje TM M_1 , takový, že $L(M) = L(M_1)$.

Idea důkazu: Vytvoříme TM se dvěma páskami, první páška je navíc dvoustopá. Na první
 stopě první pásky budou mimo konfigurace NTM oddílené hvězdičkami, na závratku tam
 bude jen jedna počáteční konfigurace. Na druhé stopě si jen snadne, kde srovná jeme. Druhá páška
 je pracovní, na ni všdy překopírujeme jednu konfiguraci z horní pásky ke spracování.

Větvení práce NTM:



První páška TM:



Simulace kroků NTM: Po každý krok přesunu na spodní pásku všechny varianty S. Pak budu
 po jedné tyhle varianty přesouvat na horní pásku (na její konec).

Pokud má NTM nekonečnou větev, bude tuto větev do nekonečna spracovávat i TM.

Návěstnost práce TM je exponenciální.

RAM (Random Access Machine) je model počítací. Skládá se z:

- 1) programové jednotky. Jednotka má programový registr, který ukazuje na instrukci, která se má vykonat
- 2) aritmetické jednotky. Jednotka provádí sčítání, odčítání, množení a celočíselné dělení
- 3) paměti. Paměť má buňky, které se adresují pořadovým číslem, obsahují nekonečné množství císel. Buňka je rekoncovatelná mnoho, tím se to liší od skutečných pc.

RAM má příkazy k přesunu paměti, aritmetickým operacím, skoku až

Časová složitost RAM: Program pracuje v čase $O(f(n))$, pokud pro každý vstup
 délky n je počet kroků počítací T(n) ve třídě $O(f(n))$.

Paměťová složitost RAM: Program pracuje s pamětí m, pokud se použije pole paměti
 s adresou m, ale nepoužije pole s adresou větší než m. Program má konstantní
 složitost $O(g(n))$, jestliže pro každý vstup délky n pracuje s pamětí $O(g(n))$.

Věta 1: ke každému TM M existuje program P pro RAM, a toto májí stejný chování. Navíc, když M potřeboval n kroků, bude P pracovat $\approx O(n^2)$.

Dk: přeskocíme, protože je snadné si představit, jak by se TM v PC implementoval.

Věta 2: ke každému programu P existuje TM M, a toto májí stejný chování

Dk: sestavíme 5-ti páskou TM, na kterém to bude dobré vidět

1. páška: simuluje paměť RAM # adresa 0 * obsah 0 # adresa 1 * ...
buněky jsou od sebe oddělené #, adresa od obsahu je oddělena *. Adresa je psána binárně.
2. páška: čítací instrukce, obsahuje pořadová čísla instrukcí, které se mají vykonat
3. páška: pamětová adresa
4. páška: vstup
5. páška: pomocná

Jak bude vypadat provedení jedné instrukce?

- 1) na druhé pášce najdu číslo instrukce, instrukci najdu podle adresy na první pášce
- 2) pokud instrukce pracuje s nějakou adresou, kopíruji ji na 3. pášku
- 3) provedu instrukci a výsledek kopíruji na správné místo
- 4) pokud se data nevejdou na místo do paměti, přesunem celý konec paměti do 5. pásky, zapsím ho do paměti a vrátím data z pásky 5 na konec pásky 1
- 5) při čtu 1 k čítací instrukci

Instrukce COPY

- 1) najdu adresu a kopíruji na 3. pásku
- 2) najdu cíl a přepíšu ho hodnotou 3. pásky (může mít všechny místa na která má vstupit někdo na pásku 5).

Instrukce ADD

- 1) napíšu si číslo k sčítání a zapíšu na třetí pásku
- 2) na první pášce najdu druhý sčítanec a sčtu ho o třetí pášku

Instrukce JUMP

Najdu adresu, kam chci skočit, a přepíšu podle toho čítací instrukci. Pokud jde o JZERO, navíc předčím skokobojin, jestliže hodnota opravdu nula, jinak abydlo skočím.

Instrukce ACCEPT: TM se přesune do koncového stavu

Instrukce REJECT: —————||———— nekoncového stavu

Věta: Ještě program P pro RAM:

- obsahuje pouze instrukce, které mají délku binárně zapsaných čísla nejvýš o 1:
- obsahuje pouze instrukce, které TM s více písmeni provede na slovech délky k $\approx O(k^2)$
pak TM s předchozí věty simuluje n kroků programu P $\approx O(n^2)$ kroků.

překladač

Dk: Přmožeme $c = \lfloor w \rfloor$, kde w je nejdélší slovo na vstupu, a d je počet slov v programu.

Po provedení n instrukcí bude mít každá binární slovo délku nejméně $c+n$. Navíc každá instrukce mohla použít adresu, kde máme nejméně $d+n$ adres.

Jedna pamětová bunka zabere $2(c+n) + 2$ buněk, protože obsahuje adresu a data délky $c+n$, a navíc smazky-oddělovače * a #.

Celkově zabere paměť délku $(2(c+n)+2) \cdot (n+d)$, což je $O(n^2)$, protože c a d jsou konstanty.

Práce se skládá z n kroků, které souborem nejméně n^2 , celkem je tedy potřeba $O(n^3)$ kroků.

Takže budeme mít TM a algoritmy volně překládat. Když algoritmus běží v polynomickém čase, potéž i v polynomickém čase i na TM.

Rozhodovací úloha: Turingův stroj vzdále skončí a vrátí jmenou odpověď ANO nebo NE.

- je π graf Hamiltonovská kružnice?
- je množina obecnit graf 3-mi barvami?
- existuje kosa délky nejméně k ?
- existuje kosa obeháního cestujícího s cenou nejméně k ? = TSP
- je formula v konjunktivním normálním tvare splnitelná? = SAT

φ je konjunkce $C_1 \wedge C_2 \wedge \dots \wedge C_n$, kde C_i je disjunkce literálů, a literál je logická proměnná nebo její negace. Když byla formula v DNF, je problém řešitelný v polynomálním čase, jenž převod CNF \rightarrow DNF je exponenciální.
Můžeme říci, že v grafu existuje cesta \Rightarrow TSP s cenou K .

Věta: Když existuje algoritmus A, který rozhoduje TSP v polynomálním čase, pak existuje polynomální algoritmus pro optimalizaci verze TSP.

Dk: Předpokládejme, že \exists polynomální alg. A pro rozhodování verze TSP. Máme n měst, $d(i,j)$ je cena cesty mezi městy i a j. Označme cenu $K = n \cdot d$, kde d je nejvyšší možné $d(i,j)$. Přitom musí platit, že d je polynomální vůči n: $d \in \Theta(p(n))$. Pro takto zadání TSP řešení zjednoduší, algoritmus vrátí ANO.

Označme $K_1 = \lfloor \frac{K}{2} \rfloor$. Pak jsou dvě možnosti:

- algoritmus vrátí ANO, v tom případě budu pokračovat dál v dělení
- algoritmus vrátí NE, pak nutně optimální řešení leží v $\langle \frac{K}{2} + 1, K \rangle$

Označme K_2 jako $K_2 = \left\lfloor \frac{K_1 + K}{2} \right\rfloor$. Protože k je vždy celé číslo, po konečném počtu opakování dostaneme kopt, když délka optimální trasy. Přitom jsem volal algoritmus A celkem $\lg(n \cdot d)$ krát, tj. polynomálně krát (protože d je taky polynomální).

Jednám o optimální cenu, a bude postupně clábat pyc hran, aby nastalo jen řešení, když tak, že budu svýsovat ceny hran na nejménou mezi (alg A potéže uplyne graf).

Uspořádáme hranu e_1, \dots, e_m , budeme je po jedné brát a nastavovat jejich cennu na $d(e_j) = \text{kopt} + 1$. Po každé změně spustím znova alg A, a jsou dílčí varianty:

- A vrátí ANO, v tom případě hrana e_i není na cestě
- A vrátí NE, pak cesta vedla přes e_i , cenu e_i mělím na původní hodnotu.

Pak mám graf, ve kterém má n hran, cena menší než $\text{kopt} + 1$, a ostatní hranu mají stejnou nejvyšší hodnotu. Hranu s maximální hodnotou leží na cestě, když jsem nastavil pomocí volání rozhodovacího algoritmu.

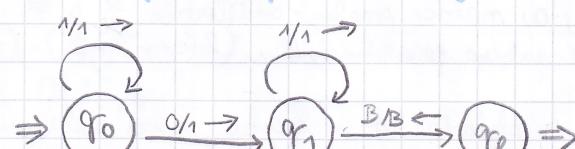
Počet volání A v druhé části byl nejméně $m = |E|$, a pro úplné grahy platí $m = \frac{n \cdot (n-1)}{2}$, což je case polynomální. Pokud polynomálně - kopt zavolejme polynomální algoritmu, výsledek bude opět polynomální, takže jsem nastavil polynomální alg. na hledání TSP.

Cílko Turingův stroj je sedmice $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$, $\delta(Q \times \Gamma^* \times \{L, R\}) \rightarrow \dots$
27.3.2019 $\rightarrow (Q \times \Gamma^* \times \{L, R\})$ stav \uparrow koncové stav $\subseteq Q$
volitelní symbol \uparrow přechodová funkce \uparrow blank $\in \Gamma \cup \Sigma$
přechodové symboly \uparrow počáteční stav $\in Q$

Př: Máme TM M_1 :

$M_1 = (\{q_0, q_1, q_F\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, B, \{q_F\})$, a přechodovou fci zademan v tabulce

δ	0	1	B
q_0	$(q_1, 1, R)$	$(q_0, 1, R)$	
q_1		$(q_1, 1, R)$	(q_F, B, L)



Ukážte práci M_1 na sloví $w = 1010$

$q_0 1010 \xrightarrow{\quad} q_0 010 \xrightarrow{\quad} q_1 10 \xrightarrow{\quad} q_1 11 \xrightarrow{\quad} q_F 0$ neúspěšně zastaven, δ nemí definovalo

Ukážte práci M_1 na slově $w = 1101$

$q_01101 + 1q_0101 + 11q_001 + 111q_11 + 1111q_1B + 111q_{ff}1$ rispečte sastavu!

Jaký jazyk M₁ přijímá?

Přijmá slova, která mají právě jednu nulu: $L(M_1) = \{w \mid |w|_0 = 1\}$

- když mělo slovo same jednichy, tak se M porád boří v Q, pak nachází a neúspěšné castan'
 - když mělo slovo něč něč, tak po načtem druhé ruly se slavou Q, neúspěšné castan'

Rzechodzie M, kento jaryk?

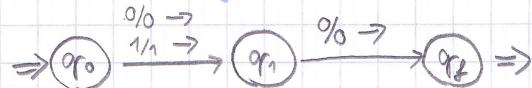
Mo, nechci jé slovo, pro které by se TM někde cyklil, hlavu se mdy posouá doprava ke konci.

Jaký jazyk je názvem M_1 ?

$L(M_1) = \{1^k \mid |w| = k\}$ se nazývá řetězec jež slovo stejně délky jako vstup, ale může mít rozdíl s ním.

Pi: Maine TM M₂, když se od M₁ liší přechodovou fází:

S	O	I	B
q_3	$(q_1, 0, R)$	$(q_1, 1, R)$	—
q_1	$(q_2, 0, R)$	—	—



Ukážte praci M_2 na slouš $\omega = 10\text{10}$

$g_0 \cdot 10 \cdot 10 + 10 \cdot g_1 \cdot 10 + 10 \cdot g_f \cdot 10$ úspěšně se sestaví

Ukážte práci H_2 na slož $w = 0$

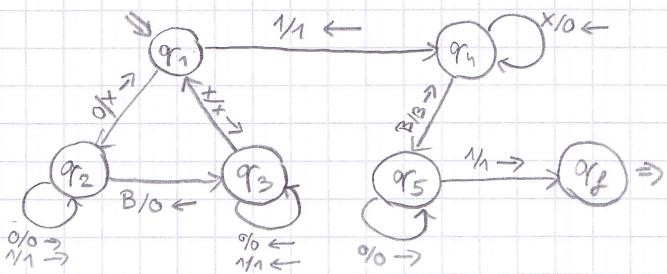
$q_0 O \vdash O q_1 B$ neispěšně se rozlavi

Jaký jazyk M_2 přijímá? Roshoduje ho?

M_2 füijíma slova, která mají jako druhý symbol nulu: $L(M_2) = \{w \mid \text{druhým znakem w je nulla}\}$.
 Tento jazyk je rozdorečen, protože opět v M_2 nejsou cykly, kde by se M_2 mohl strádat.

Ex: Minim TH M₃, körüljáró Q = {q₁, q₂, q₃, q₄, q₅, q_F}, Γ = {0, 1, X, B} a tablau S:

S	O	A	B	X
q_1	(q_{p_2}, X, R)	$(q_{p_1}, 1, L)$	—	—
q_2	$(q_{p_2}, 0, R)$	$(q_{p_2}, 1, R)$	$(q_{p_3}, 0, L)$	—
q_3	$(q_{p_3}, 0, L)$	$(q_{p_3}, 1, L)$	—	(q_{n_1}, X, R)
q_4	—	—	(q_{p_1}, B, R)	$(q_{n_1}, 0, L)$
q_5	$(q_{p_5}, 0, R)$	$(q_{p_1}, 1, R)$	—	—



Ukážka frekvencie Hz má súčinu $\omega = 0,101$

$q_1 0101 \vdash X q_2 101 \vdash^3 X 101 q_2 B \vdash X 10 q_3 10 \vdash^3 q_3 X 1010 \vdash X q_1 1010 \vdash$
 $\vdash q_4 X 1010 \vdash q_4 B 01010 \vdash q_5 01010 \vdash 0 q_5 1010 \vdash 01 q_6 010 \text{ následně nastaví'}$

Ukážte práci H_3 na $w = 0$

$q_1 O \vdash X q_2 B \vdash q_3 X O \vdash X q_1 O \vdash XX q_2 B \vdash X q_3 X O \vdash XX q_1 O \dots$
 Nezastaví se nikdy, lze se počítat cyklist v q_1, q_2, q_3 , doleva přidáváš další X k němu

Jaký jazyk M3 přijímá? Rovnodiely kdo? Co je nejdůležitější?

M_3 prvníma slova, která mají aspoň jednu jehněčku. Pokud jednáme množinu, lze na se cyklisty q_1, q_2, q_3 . Zde je se 1 následek, že projde se do slavného q_1 , kde v se jen směrem $X \rightarrow 0$ a slovo je vymýšleno. $L = \{w_1 \mid |w_1| \geq 1\}$.

Jazyk nemá rozhodování, protože některá slova se vyhýbají (např. $w=0$).

Výsledek je jazyk $L'(M_3) = \{0^k u 0^k \mid 0^k u \in L(M_3)\}$, tj. všechny počáteční řetězce připomíne na konci slova smysl - to je proto, že po každém přepsaném $B \rightarrow X$ ve stavu q_1 se provede jednorázovou přepsaním $B \rightarrow O$ ve stavu q_2 (na konci slova).