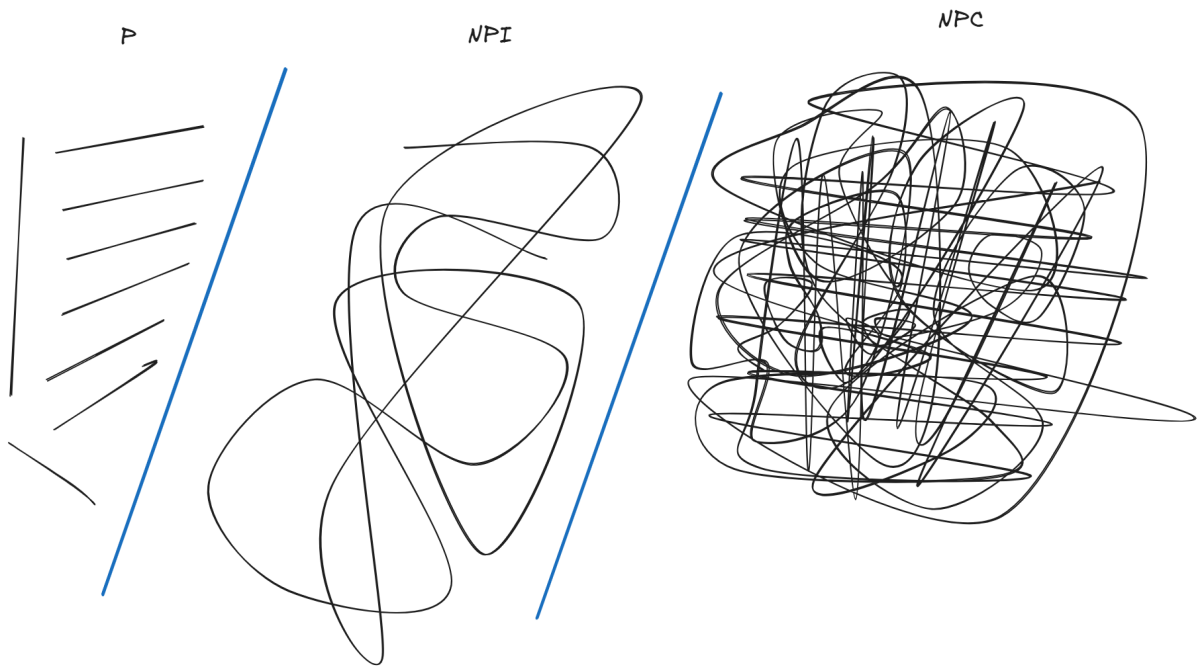
Vmesni problemi, diagonalizacija

Uroš Čibej



Pregled

1. Ladnerjev izrek
2. Izreki o časovnih hierarhijah
3. Meje diagonalizacije

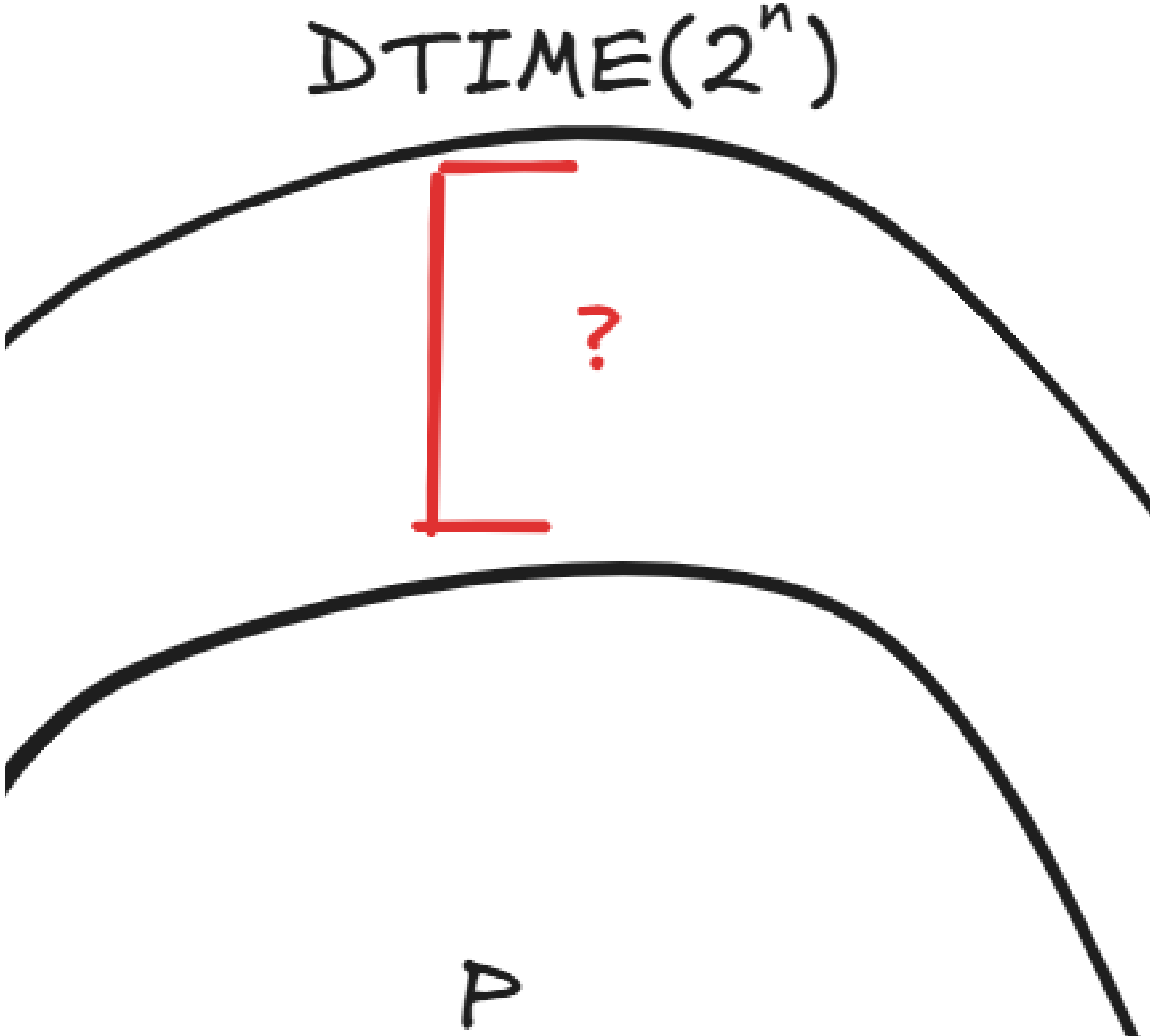
Nekaj o funkcijah

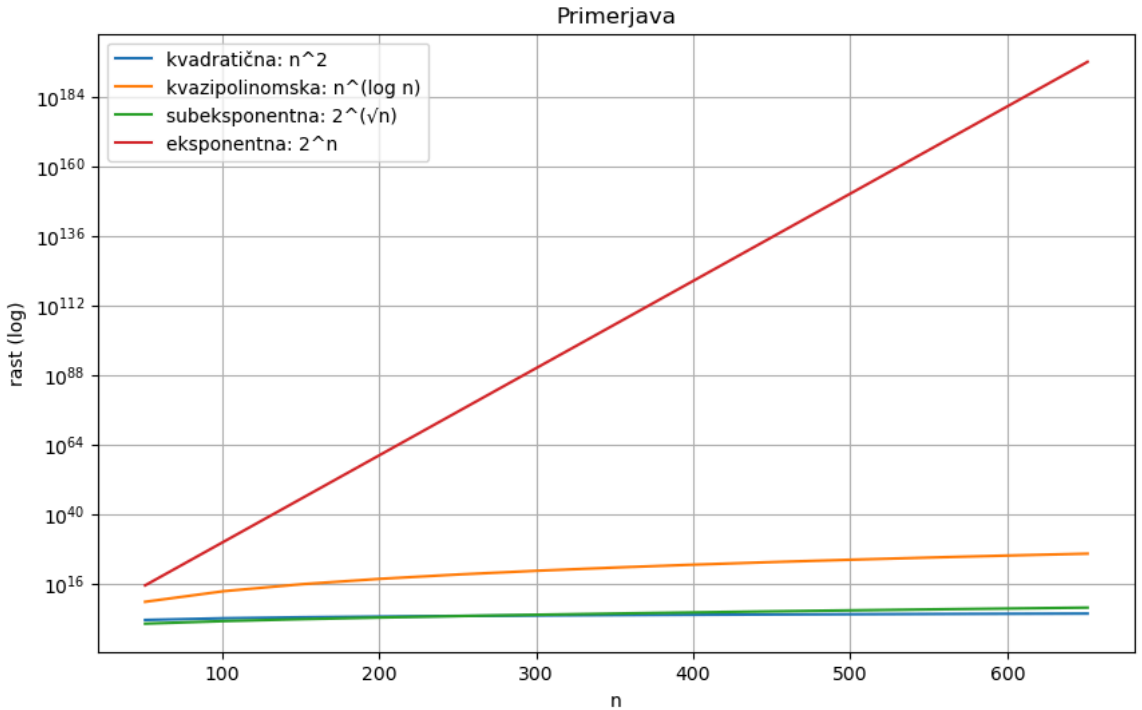
- subeksponentna časovna zahtevnost

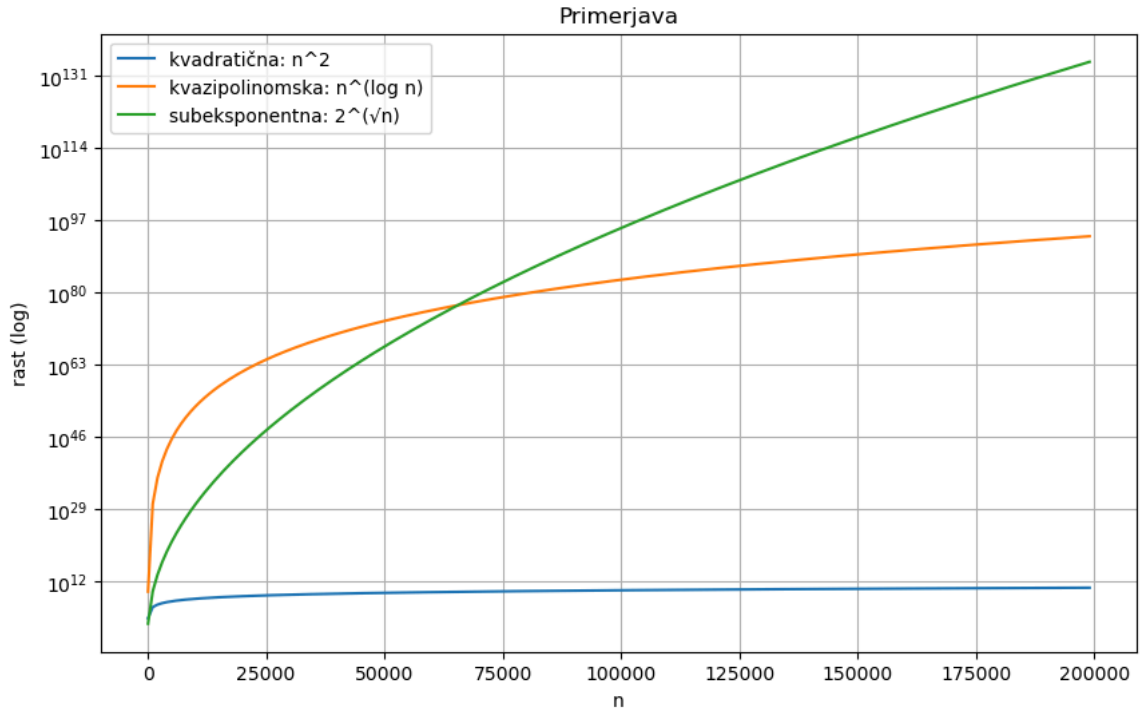
$$SUBEXP = DTIME(2^{o(1)})$$

- kvazipolinomska časovna zahtevnost

$$QP = \bigcup_c n^{\log n^c}$$




Primerjava
kvadratična: n^2
kvazipolinomska: n^(log n)
subeksponentna: 2^(√n)
eksponentna: 2^n
rast (log)
n
10^{184}
10^{160}
10^{136}
10^{112}
10^{88}
10^{64}
10^{40}
10^{16}
100
200
300
400
500
600


Primerjava
kvadratična: n^2
kvazipolinomska: n^(log n)
subeksponentna: 2^(√n)
rast (log)
n
10^{131}
10^{114}
10^{97}
10^{80}
10^{63}
10^{46}
10^{29}
10^{12}
0
25000
50000
75000
100000
125000
150000
175000
200000

Hipoteza o eksponentni časovni zahtevnosti (ETH)

3-SAT ni rešljiv v subeksponentnem času

Bolj formalno:

Obstaja konstanta $c < 1$ in $c > 0$, da problem 3-SAT z n spremenljivkami ni rešljiv v času 2^{cn}

Ladnerjev izrek

$$P \neq NP \implies \exists A \in NP, A \notin P, A \notin NPC$$

Ladnerjev izrek ob ETH

$$ETH \implies \exists A \in NP, A \notin P, A \notin NPC$$

Tehnika zapolnjevanja (padding)

Problem umetno poenostavimo (pohitrimo njegovo reševanje?), tako da mu umetno dodamo balast

Banalen primer

$$L_{banal} = \{\langle x, 1^* \rangle \mid |x| < 7, x \in 3SAT\}$$

Umetni problem

$$L = \{\langle x, 1^{2^{\sqrt{|x|}}} \rangle \mid x \in 3SAT\}$$

$L \in NP$

```
V(w, c):
    preveri format w, vrni FALSE, če v napačnem formatu
    x, ones = w
    return CheckSat(x, c)
```

$$L \notin P$$

Ne obstaja algoritem (ob ETH) za L, ki bi za velikost problema N porabil največ N^c časa.

Dokaz s protislovje z ETH

Predpostavimo, da obstaja polinomski algoritem A za L, konstruirajmo hiter algoritem A' za 3-SAT.

```
def A'(phi): # phi je dolžine n
    y = <phi, 1^2^{sqrt(len(x))}>
    return A(y)
```

Časovna zahtevnost algoritma A'

- konstrukcija niza y traja $O(2^{\sqrt{n}})$
- A teče N^c časa, velikost vhoda je $2^{\sqrt{n}}$, torej

$$(2^{\sqrt{n}})^c = 2^{O(\sqrt{(n)})}$$

- vse skupaj torej $2^{O(\sqrt{n})}$, kar je subeksponenten čas, v nasprotju z ETH.

$L \notin NPC$

Predpostavimo, da $L \in NPC$ (obstajajo torej prevedbe $X \leq_p L$, za $X \in NP$).

Kako hitro pa lahko rešimo L?

```
def solve_L(w):
    preveri format w, vrni FALSE, če v napačnem formatu
    x, ones = w
    return brute_force(x) # v času 2^n
```

Koliko časa traja ta algoritem v odvisnosti od $N = |w|$?

$$O(N^{\log N})$$

Algoritem za 3-SAT

naj bo f prevedba iz $3SAT$ na L (predpostavili smo obstoj)

```
def solve_3SAT(phi):
    y = f(phi)
    return solve_L(y)
```

1. dolžina $|y| = O(n^c)$
2. čas izvajanja `solve_L` je torej

$$(n^c)^{\log n^c} = n^{O(\log n)}$$

Zopet imamo subeksponentni algoritem za 3-SAT, protislovje z ETH

Dokaz brez ETH

- splošen dokaz (Ladnerjev izrek) uporablja diagonalizacijo
- Tehnika, ki smo jo uporabili zgoraj pa deluje za poljubno predpostavko oblike: Najhitrejši možen algoritem za 3-SAT je

$$f(n)$$

- Ne pokrije pa možnosti, da mogoče najhitrejši algoritem za 3-SAT ne obstaja

Naravni NPI problemi

- faktorizacija

$$FAKT = \{\langle n, a, b\rangle \mid m \text{ ima faktor na intervalu } [a, b]\}$$

- grafni izomorfizem

$$GI = \{\langle G, H\rangle \mid G \text{ je izomorfen } H\}$$

Kaj vemo (faktorizacija)

- $FAKT \in NP$
- $\overline{FAKT} \in NP$
- ne poznamo:
  - $FAKT \in P$
  - $FAKT \in NPC$
- RSA temelji na predpostavki "nerešljivosti" tega problema
- rešljiv v **polinomskem času na kvantnem računalniku**

Kaj vemo (grafni izomorfizem)

- $GI \in NP$
- ne poznamo
  - $GI \in P$
  - $GI \in NPC$
  - $\overline{GI} \in NP$
- imamo izjemno hitre praktične reševalnike (nauty)

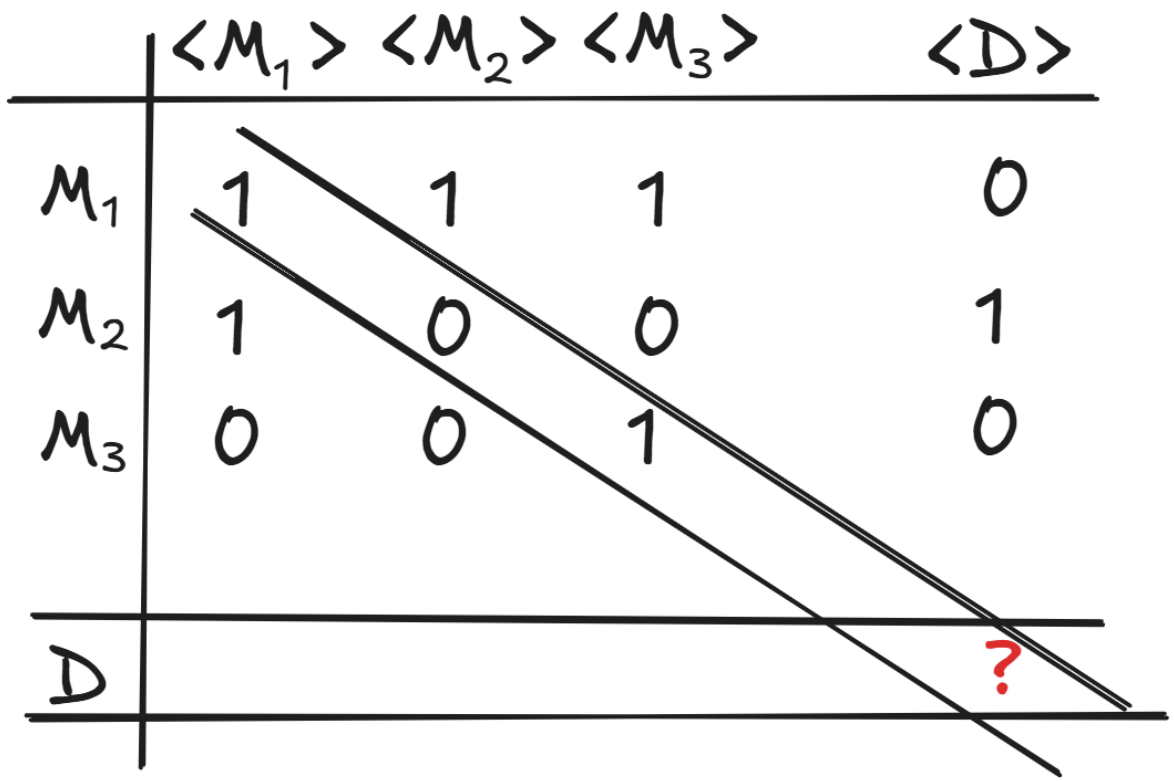
Velik preboj (Babai, 2015)

Babai je pokazal, da je GI rešljiv v

$$n^{\log n^c}$$

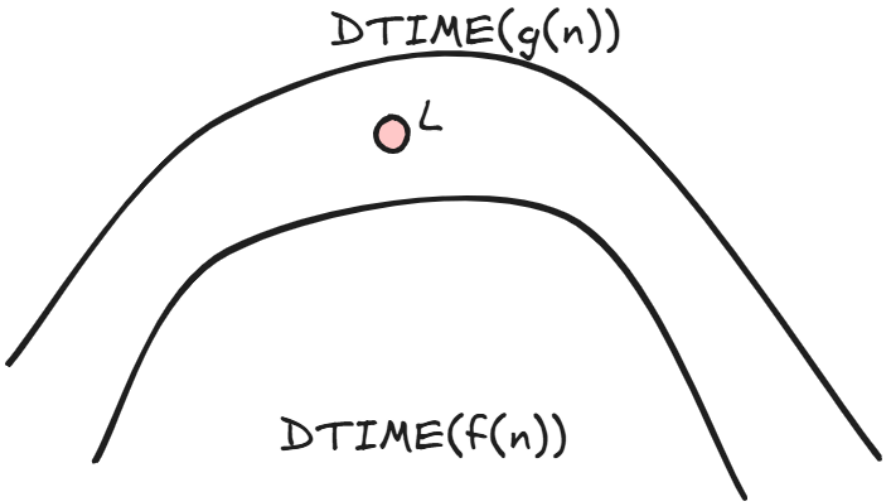
To je **kvazipolinomski čas**.

Diagonalizacija (kot jo že poznate)



Izrek o časovnih hierarhijah

Naj bosta funkciji f, g časovno predstavljivi in $f(n) \log f(n) = o(g(n))$, potem $DTIME(f(n)) \subsetneq DTIME(g(n))$



Dokaz

Dokažimo $DTIME(n) \subsetneq DTIME(n^{1.5})$

Naj bo $M_u(M, w, timeout)$ univerzalni Turingov stroj z omejitvijo korakov.

```
def D(x):
    answer = M_u(x, x, |x|^1.4)
    if answer = timeout:
        return 0
    else:
        return 1-answer
```

Vemo (tako je D definiran), da $L(D) \in DTIME(n^{1.5})$

Dokazati moramo, da $L(D) \notin DTIME(n)$

$L(D) \notin DTIME(n)$

Predpostavimo, da obstaja M, ki v $c|x|$ korakih vrne $D(x)$

Ta stroj naj ima kodo m, kakšen je odgovor $D(m)$?

```
def D(x):
    answer = M_u(x, x, |x|^1.4)
    if answer = timeout:
        return 0
    else:
        return 1-answer
```

Časovne hierarhije z nedeterminističnimi stroji

Zelo podoben izrek obstaja za nedeterministične hierarhije

Posledica

$$EXP = \bigcup_{k \geq 1} DTIME(2^k)$$

$$P \subsetneq EXP$$

Meje diagonalizacije

Izrek: **Baker-Gil-Solovay**

1. Diagonalizacija deluje za marsikatere izreke o hierarhijah (vemo recimo $P \neq EXP$)
2. Diagonalizacija relativizira (predpostavlja obstoj nekega algoritma)
3. Nobena tehnika, ki relativizira, ne more razrešiti vprašanja $P = NP, P \neq NP$