Izračunljivost in računska zahtevnost

1. **Turingov stroj**
   - **teza o izračunljivosti**
   - **Turingov stroj**
     - o Turingovem stroju
     - trenutni opis
   - **uporaba Turingovega stroja**
     - računanje vrednosti funkcij (k-mestna lastna funkcija)
     - razpoznavanje množic (sprejeta beseda, sprejet jezik, množica S)
     - generiranje množice (generator, jezik generiran z TM, izračunljivo preštevni jeziki)
   - **razširitve Turingovih strojev**
   - **univerzalni Turingov stroj**
     - kodiranje TM
     - oštevilčenje TM

2. **Neodločljivost**
   - **računski problemi (odločitveni, problemi iskanja, problemi štetja, problemi generiranja)**
   - **odločitveni problemi**
     - jezik odločitvenega problema
     - kodirna funkcija
     - povezava/enakovrednost med odločitvenimi problemi in množicami/jeziki
     - razpoznavnost L(D) in rešljivost problema D
   - **problem ustavitve**
     - problem ustavitve
     - univerzalni jezik
     - diagonalni jezik
   - **osnovne vrste odločitvenih problemov**
     - razred vseh odločitvenih problemov
     - vrste odločitvenega problema D
   - **komplementarne množice in odločitveni problemi**
   - **izreki iz teorije izračunljivosti (odločljivost in polodločljivost množic/jezikov)**
   - **neizračunljiv problem - problem garača (busy beaver)**

3) **Teorija računske zahtevnosti**

- **deterministična časovna zahtevnost (razred zahtevnosti DTIME)**
  - TM M
  - jezik L in razred vseh takih jezikov
  - odločitveni problem in razred vseh takih odločitvenih problemov
- **nedeterministična časovna zahtevnost (razred zahtevnosti NTIME)**
  - TM N
  - jezik L in razred vseh takih jezikov
  - odločitveni problem in razred vseh takih odločitvenih problemov
- **deterministična prostorska zahtevnost (razred zahtevnosti DSPACE)**
  - TM M
  - jezik L in razred vseh takih jezikov
  - odločitveni problem in razred vseh takih odločitvenih problemov
- **nedeterministična prostorska zahtevnost (razred zahtevnosti NSPACE)**
  - TM N
  - jezik L in razred vseh takih jezikov
  - odločitveni problem in razred vseh takih odločitvenih problemov
- **stiskanje trakov, linearna pohitritev, zmanjševanje števila trakov**
- **relacije med DTIME, DSPACE, NTIME in NSPACE**
  - relacije med različnimi razredi zahtevnosti
  - lepe, pohlevne funkcije zahtevnosti
    - prostorsko predstavljiva funkcija S(n)
    - časovno predstavljiva funkcija T(n)
- **razredi zahtevnosti P, NP, PSPACE, NPSPACE**
- **osnovne relacije med P, NP, PSPACE, NPSPACE**
- **NP-poln in NP-težek problem**
  - polinomsko-časovna prevedba
  - NP-težek problem
  - NP-poln problem
  - dokazovanje NP-polnih problemov
  - $P \neq NP$

Izračunljivost in računska zahtevnost

1) Turingov stroj

- **teza o izračunljivosti (Church-Turingova teza)**

  Osnovni koncepti računanja so formalizirani:

  - algoritem je formaliziran z Turingovim programom
  - računanje je formalizirano z izvršitvijo Turingovega programa v Turingovem stroju
  - funkcija izračunljivosti je formalizirana z Turingovo izračunljivo funkcijo

  Vse, kar je možno izračunati v intuitivnem smislu, je mogoče izračunati z enim od standardnih univerzalnih modelov računanja in obratno.

- **Turingov stroj**
  - **o Turingovem stroju**

    Turingov stroj ima kontrolno enoto, ki vsebuje Turingov program; trak sestavljen iz celic in premično okno čez trak, ki je povezano z kontrolno enoto.

    Celica vsebuje tračni simbol, ki pripada tračni abecedi.
    Vhodni podatki so vsebovani v vhodni besedi.
    Kontrolna enota je vedno v nekem stanju iz končne množice stanj.
    Turingov program je funkcija prehodov.

    TM je sedmerica $\mathbf{T = (Q, \Sigma, \Gamma, \delta, q_1, B, F)}$.

    Q ... končna množica stanje

    Σ ... končna množica vhodnih simbolov

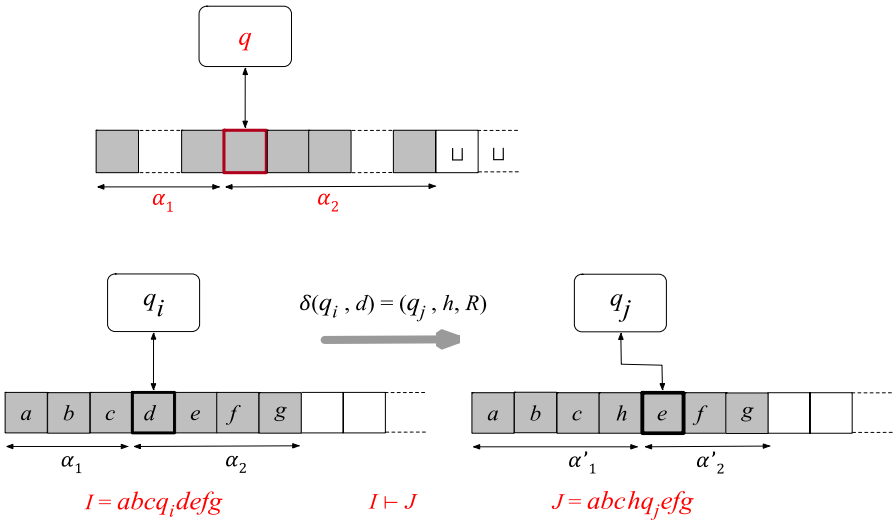
    Γ … množica tračnih simbolov

    δ … funkcija prehodov: $\mathbf{Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, S\}}$

    q_1 ... začetni simbol
    B … prazen simbol
    F … množica končnih stanj

- **trenutni opis TM**

  To je niz I = $\alpha_1 q \alpha_2$, če je trenutna konfiguracija TM takšna, da je okno nad prvim simbolom niza α_2 in da se α_2 konča na najbolj desnem nepraznem simbolu.

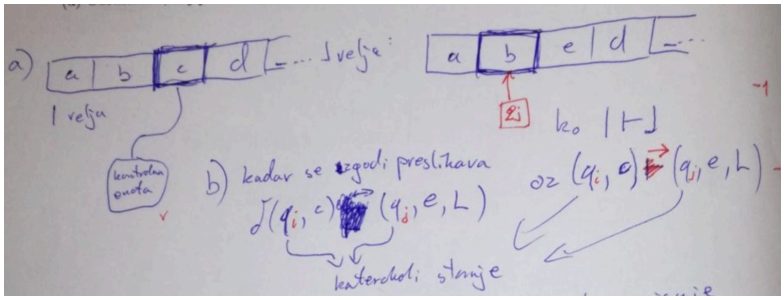


I = (abq$_i$cd) in J = (aq$_j$bed).

Nariši M, ko velja I in ko velja J.

Kdaj se I lahko neposredno spremeni v J.

ID I lahko neposredno preide v J .... I ⊢ J -- če obstaja navodilo v TM programu, čigar izvedba, I preide v J.



Ali je relacija ⊢ neposredne spremembe tranzitivna?

Ni tranzitivna, saj iz a ⊢ b in b ⊢ c ne sledi a ⊢ c, ker opisuje posamezno spremembo, ki definira začetno in končno stanje v vsakem koraku, torej iz a ⊢ b sledi natanko samo a ⊢ b.

Kaj pomeni Kleenejevo zaprtje ⊢*.

Vsi trenutni opisi TM, ki se jih da po preslikovalni funkciji po x >= 0 potezah iz nekega trenutnega stanja doseči.

- **uporaba Turingovega stroja**
  - **računanje vrednosti funkcij**

    (Dana je funkcija φ in argumenti $a_1 , \dots , a_k$, izračunaj $\varphi(a_1 , \dots , a_k)$. Dana je k-mestna lastna funkcija $\varphi : (\Sigma^*)^k \rightarrow \Sigma^*$, poišči TM T, ki lahko izračuna vrednosti φ. Dan je φ, kontruirati moramo TM T za katerega velja $\varphi_T = \varphi$.)

    K-mestna lastna funkcija *T - ja* je funkcija $\varphi_T : (\Sigma^*)^k \rightarrow \Sigma^*$

    Če je vhod T-ja sestavljen iz *k* besed $u_1, \dots , u_k \in \Sigma^*$, potem je vrednost φ_T pri $u_1, \dots , u_k$ *definirana*

    $\varphi_T (u_1,\dots, u_k) :=$

    - v , če se T ustavi in vrne na trak besedo $v \in \Sigma^*$
    - ↑, *če se T* ne ustavi *ali* trak nima besede v Σ^*

    Funkcija φ:

    - **izračunljiva**: če obstaja tak TM, ki **lahko izračuna** vrednosti φ **za vsak argument**
    - **delno izračunljiva:** če obstaja tak TM, ki **lahko izračuna** vrednosti φ **kadarkoli je** φ **definirana**
    - **neizračunljiva:** če **ne obstaja** noben tak TM, ki bi lahko izračunal vrednosti φ **kadarkoli je** φ **definirana**

- **razpoznavanje množic**

(Dana je množica S ⊆ Σ^*, poišči TM T, ki sprejme S. Dan je jezik (množica) S, konstruirati moramo TM T, za katerega velja L(T) = S.)

Beseda w je sprejeta z TM *T*, če $q_1 w \vdash^* \alpha_1 p \alpha_2$, za nek $p \in F$ in $\alpha_1\alpha_2 \in \Gamma^*$.

(Beseda povzroči, da T vstopi v končno stanje, $w \in \Sigma^*$ niz.)

Jezik L sprejet z TM *T* je množica ***L*(*T*)** = $\{w \mid w \in \Sigma^* \wedge w$ je sprejet z $T\}$.

(Jezik sprejet z T je sestavljen iz točno takih besed.)

Množica/jezik S ⊆ Σ^* (sposobnost TM-ja razpoznavati množice):

- **odločljiv**: če obstaja tak TM, **ki odgovori z DA/NE na "Je x ∈ S?" za vsak x ∈ Σ^***
- **polodločljiva**: če obstaja tak TM, **ki odgovori z DA na "Je x ∈ S?" kadarkoli x ∈ S**
- **neodločljiva**: če **ne obstaja** noben TM, **ki odgovori z DA/NE na "Je x ∈ S?" za vsak x ∈ Σ^***

- **generiranje množic**

(Dana je množica S, generiraj seznam $x_1, x_2, x_3, \ldots$, ki so točno elementi S. Dan je jezik (množica) *S*, konstruirati moramo TM *T* za katerega velja *G* (*T*) = S.)

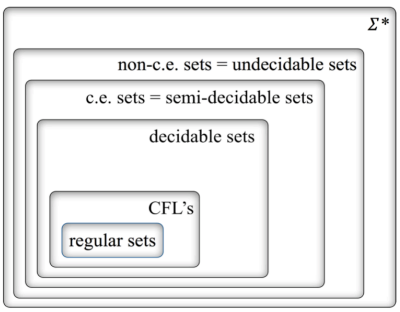
TM *T* je **generator**, če piše na trak besede iz Σ^*, v zaporedju in razmejevano (delimited) z #.

Jezik generiran z TM T je množica **G(T)** = {w | w ∈ Σ^*∧ T sčasoma zapiše w na trak}.

Izračunljivo preštevni jeziki (množice)

Množica *S* je **izračunljivo preštevna** (c.e.), če je ***S* = *G*(*T*)** za nek TM T (če je *S* lahko generirana z Turingovim strojem).

Množica S je **izračunljivo preštevna** iff **S je polodločljiva**.



- **razširitve Turingovih strojev**
  - **TM s končnim pomnilnikom**
  - **TM z večslednim trakom**
  - **TM z dvosmernim trakom**
  - **TM z večtračnim trakom**
  - **TM z večdimenzionalnim trakom**
  - **TM z nedeterminističnim programom**

    Turingov program δ dodeli vsakemu (q_i, z_r) končno množico alternativnih prehodov.

    Stroj nedeterministično izbere prehod iz množice in ga naredi.
    Z njim definiramo minimalno število korakov, ki jih potrebujemo, da izračunamo rešitev (če rešitev obstaja). To je pomembno, ko preiskujemo računsko zahtevnost problema.

- **univerzalni Turingov stroj (UTM)**

  Univerzalni TS izračuna vse kar je izračunljivo z kakšnim drugim Turingovim strojem.
  Kot vhod sprejme TM (njegovo kodo) in njegov vhod (besedi w), ter ta stroj na podanem vhodu simulira.
  - **kodiranje TM**

    TM predstavimo z besedo nad neko kodirno abecedo. Šifra < T > TM *T* je identificirana z < δ >.
  - **oštevilčenje TM**

    <T> lahko interpretiramo kot binarno kodo nekega naravnega števila. To število je indeks T - ja.
    Vsako naravno število je indeks točno enega Turingovega stroja.
    Oštevilčenje osnovnih Turingovih strojev (zaporedje TM) dobimo, če gre n čez 0, 1, 2 ... T_0, T_1, T_2, …

2) Neodločljivost

- **računski problemi**
  - **odločitveni problemi** (rešitev je odgovor DA ali NE)
  - **problemi iskanja** (rešitev je element množice)
  - **problemi štetje** (rešitev je naravno število)
  - **problemi generiranja** (rešitev je zaporedje elementov množice)

- **odločitveni problemi**
  - **jezik odločitvenega problema**

    Povezava med odločitvenimi problemi in množicami/jeziki, zato lahko skrčimo vprašanja o odločitvenih problemih na vprašanja o množicah.

    D je odločitveni problem in množica vseh možnih primerkov d D-ja. **Primerek d** je lahko pozitiven/negativen, če je odgovor za d DA/NE.

    Kodirna funkcija (code): $D \rightarrow \Sigma^*$, spremeni vsak primerek D-ja v besedo iz Σ^*.

    code (D) oz. **<d>** je množica kod vseh primerkov problema D.

    Jezik odločitvenega problema *D* je množica $L(D)$, ki je definirana z

    $L(D) = \{ \langle d \rangle \in \Sigma^* \mid d \text{ je pozitiven primerek problema D} \}$.

    Povezava/enakovrednost med odločitvenimi problemi in množicami/jeziki (*):

    ***d* ∈ *D* je pozitiven ⟺ <d> ∈ *L* (*D*)**

    Reševanje odločitvenega problema D je lahko zreducirano v razpoznavanje množice L(D) v Σ^*.

    Odgovor za primer d problema D lahko najdemo, če določimo, kako je <d> povezan z L(D).

- **razpoznavnost L(D) in rešljivost problema D**
  - L(D) je **odločljiv <--->** D je **odločljiv (izračunljiv)**
    Obstaja algoritem, ki za vsak d ∈ D, odgovori z DA ali NE.

    (Obstaja TM, ki se za vsak <d> ∈ Σ^*, odloči, ali je ali ni <d> ∈ L(D).)

  - L(D) je **polodločljiv <--->** D je **polodločljiv**
    Obstaja algoritem, ki
    - za vsako pozitivni primerek d ∈ D, odgovori z DA
    - za negativen primerek d ∈ D, v končnem času lahko odgovori z NE ali pa ne odgovori z NE

    (Obstaja TM, ki za vsak <d> ∈ L(D) sprejme <d>. Če <d> ∉ L(D), algoritem lahko zavrže ali ne zavrže <d> v končnem času.)

  - L(D) je **neodločljiv <--->** D je **neodločljiv (neizračunljiv)**
    Ne obstaja algoritem, ki bi za vsak d ∈ D odgovoril z DA ali NE.

    (Ne obstaja TM, ki je sposoben odločanja, za vsak <d> ∈ Σ^*, ali je ali ni <d> ∈ L(D).)

    *D je odločljiv, če je L(D) odločljiva množica.

- **problem ustavitve (neizračunljiv problem)**
  - **problem ustavitve**

    $\mathbf{D_{Halt}}$ = "Dan je TM T in w ∈ Σ^*, ali se T ustavi na w?"

    Ne obstaja noben algoritem, ki bi bil sposoben za poljuben T in w, odgovoriti na vprašanje "Ali se T ustavi na w?"

  - **univerzalni jezik** (jezik problema ustavitve)

    $\boldsymbol{L_u} = L(D_{Halt}) = \{ \langle T, w \rangle \mid T \text{ se ustavi na } w \}$

    Dan je poljuben vhod ⟨*T, w*⟩, stroj mora simulirati *T* na *w*, in *če* se simulacija *ustavi*, mora stroj vrniti DA in se ustaviti. Če tak stroj obstaja, bo odgovoril z DA *iff* $\langle T, w \rangle \in ? K_0$

    $D_{\overline{Halt}}$ = Dan je TM *T* in beseda *w*, ali se *T nikoli ne ustavi* na *w*?
  - **diagonalni jezik**, D_H

    $\mathbf{L_d} = \{ \langle T, T \rangle \mid T \text{ se ustavi na } \langle T \rangle \}$ // dobimo ga iz univerzalnega z uvedbo w = <T>

    $D_{\overline{H}}$ = Dan je TM T, ali se T nikoli ne ustavi na ⟨T ⟩?

    <u>Dokaz</u>

    Dokažemo, da je L_d neodločljiva množica, kar implicira na to, da je L_u neodločljiva, posledično je D_{Halt} neodločljiv problem.

- **osnovne vrste odločitvenih problemov**
  - **neodločljivi in nepolodločljivi**: komplement L_u (komplement D_{Halt}) in komplement L_d (kompelement D_H)
  - **nedoločljivi in polodločljivi**: L_u (D_{Halt}), L_d (D_H)

  Razred vseh odločitvenih problemov je sestavljen iz
  - razred **odločljivih** problemov
  - razred **neodločljivih** problemov

  Obstaja še treji razred, razred **polodločljivih** problemov (vsebuje vse odločljive in nekaj neodločljivih problemov).

  Odločitveni problem D je lahko
  - **odločljiv**

    Obstaja algoritem (**odločevalec** problema D), ki lahko reši poljubni primerek $d \in D$.
  - **polodločljivo neodločljiv**

    Ne obstaja algoritem (**prepoznalec** problema D), ki bi rešil poljuben primerek $d \in D$, ampak obstaja algoritem, ki lahko reši poljubnen pozitiven primerek $d \in D$.
  - **ni polodočljiv**

    Za poljuben algoritem obstaja pozitiven in negativen primerek $d \in D$, tak, da ju algoritem ne more rešiti.

- **komplementarne množice in odločitveni problemi**
  - S in komplement S sta **odločljiva**
  - S in komplement S sta **neodločljiva**; **en je pododločljiv** in drugi ni
  - S in komplement S sta **neodločljiva**; **noben** od njiju ni polodločljiv

- **izreki iz teorije izračunljivosti (odločljivost in polodločljivost množic/jezikov)**

  S *je odločljiv* $\implies S$ *je polodločljiv*

  S *je odločljiv* $\implies \overline{S}$ *je odločljiv*

  S *in* $\overline{S}$ *sta polodločljiva* $\implies S$ *je odločljiv*

  A *in* B *sta polodločljiva* $\implies A \cap B$ *in* $A \cup B$ *sta polodločljiva*

  A *in* B *sta odločljiva* $\implies A \cap B$ *in* $A \cup B$ *sta odločljiva*

  Komplement polodločljivega ampak neodločljivega jezika ne more biti polodločljiv.

- **neizračunljiv problem - problem garača (busy beaver)**

  $\mathcal{T}_n$ (za $n \geq 1$) je razred vseh TM, ki imajo:

  - neomejen trak v obe smeri
  - n ne-končnih stanj (vključno q_1) in 1 končno stanje q_{n+1}
  - $\Sigma = \{0,1\}$ in $\Gamma = \{0,1,\sqcup\}$
  - δ piše samo simbol 1 in premika okno v levo ali v desno

    TM $T \in \mathcal{T}_n$ je stroj, ki se ustavi, če se T ustavi pri praznem vhodu.

3) <u>Teorija računske zahtevnosti</u>

Koliko časa ali prostora potrebuje algoritem, da reši odločitveni problem?

Koliko korakov ali tračnih celic potrebuje TM, da prepozna jezik L(D) odločitvenega problema D?

DTIME(T (n)) = {odločitveni problemi deterministično rešljivi v času T(n)}

DSPACE(S (n)) = {odločitveni problemi deterministično rešljivi na prostoru S(n)}

NTIME(T (n)) = {odločitveni problemi nedeterministično rešljivi v času T(n)}

NSPACE(S (n)) = {odločitveni problemi nedeterministično rešljivi na prostoru S(n)}

- **deterministična časovna zahtevnost (razred zahtevnosti DTIME)**

  M = (Q, Σ, Γ, δ, q_1, ⊔, F) DTM z k ≥1 dvosmernimi neskončnimi trakovi.

  - **TM M**

    M je deterministične časovne zahtevnosti T(n), če za vsak vhod w ∈ Σ* dolžine n,

    M naredi ⩽ T(n) korakov pred ustavitvijo.

    (w ∈? L(M) v ⩽ T(|w|) korakih)

    Privzeto je, da M prebere vse w - je … T(|w|) ⩾ |w| + 1, T(n) ⩾ n + 1. T(n) je najmanj linearen.

  - **jezik L in razred vseh takih jezikov**

    Jezik L je deterministične časovne zahtevnosti T(n), če obstaja DTM M deterministične časovne zahtevnosti T(n), da velja L = L(M).

    DTIME(T(n)) = {L | L je jezik ∧ L je deterministične časovne zahtevnosti T (n)}

    (vsi L-ji za katere je problem w ∈? L lahko deterministično rešljiv v ⩽ T (|w|) času)

  - **odločitveni problem in razred vseh takih odločitvenih problemov**

    Odločitveni problem D je deterministične časovne zahtevnosti T(n), če je njegov jezik L(D) deterministične časovne zahtevnosti T(n).

    DTIME(T(n)) = {D | D je odločitveni problem ∧ D je deterministične časovne zahtevnosti T(n)}

    (vsi D - ji čigar primerki d so lahko deterministično rešljivi v ⩽ T (|⟨d ⟩|) času)

- **nedeterministična časovna zahtevnost (razred zahtevnosti NTIME)**

  N = (Q, Σ, Γ, δ, q1, ⊔, F) nedeterministični NTM.

  - **TM N**

    N je nedeterministične časovne zahtevnosti T(n), če za vsak vhod w ∈ Σ* dolžine n,

    N naredi ⩽ T(n) korakov pred ustavitvijo.

  - **jezik L in razred vseh takih jezikov**

    Jezik L je nedeterministične časovne zahtevnosti T(n), če obstaja NTM N nedeterministične časovne zahtevnosti T(n), da velja L = L(N).

    NTIME(T(n)) = {L | L je jezik ∧ L je nedeterministične časovne zahtevnosti T (n)}

  - **odločitveni problem in razred vseh takih odločitvenih problemov**

    Odločitveni problem D je nedeterministične časovne zahtevnosti T(n), če je njegov jezik L(D) nedeterministične časovne zahtevnosti T(n).

    NTIME(T(n)) = {D | D je odločitveni problem ∧ D je nedeterministične časovne zahtevnosti T (n)}

- **deterministična prostorska zahtevnost (razred zahtevnosti DSPACE)**

  M = (Q, Σ, Γ, δ, q1, ⊔, F) DTM z 1 vhodnim trakom in k ⩾1 delovnimi trakovi.

  - **TM M**

    M je deterministične prostorske zahtevnosti S(n), če, za vsak vhod w ∈ Σ* dolžine n,

    M pred zaustavitvijo uporabi ⩽ S(n) celic na vsakem delovnem traku.
    (S(n) se lahko odloči w ∈? L(M) na prostoru ⩽ S(n)).
    M uporabi najmanj celico pod začetnim položajem okna. S(n) je najmanj konstantna funkcija 1.

  - **jezik L in razred vseh takih jezikov**

    Jezik L je deterministične prostorske zahtevnosti S(n), če obstaja DTM M deterministične prostorske zahtevnosti S(n), da velja L = L(M).
    DSPACE(S(n)) = {L | L je jezik ∧ L je deterministične prostorske zahtevnosti S (n)}
    (vsi L-ji za katere je lahko problem w ∈? L deterministično rešljiv na ⩽ S(|w|) prostoru)

  - **odločitveni problem in razred vseh takih odločitvenih problemov**

    Odločitveni problem D je deterministične prostorske zahtevnosti S(n), če ima njegov jezik L(D) deterministično prostorsko zahtevnost S(n).
    DSPACE(S(n)) = {D | D je odločitveni problem ∧ D je deterministične prostorske zahtevnosti S(n)}
    (vsi D-ji čigar primerki d so lahko deterministično rešljivi na ⩽ S (|⟨d ⟩|) prostoru)

- **nedeterministična prostorska zahtevnost (razred zahtevnosti NSPACE)**

  N = (Q, Σ, Γ, δ, q1, ⊔, F) nedeterministični NTM z 1 vhodnim trakom in k ⩾1 delovnimi trakovi.

  - **TM N**

    N je nedeterministične prostorske zahtevnosti S(n), če za vsak vhod w ∈ Σ* dolžine n,

    N pred ustavitvijo uporabi ⩽ S(n) celic na vsakem delovnem traku.

  - **jezik L in razred vseh takih jezikov**

    Jezik L je nedeterministične prostorske zahtevnosti S(n), če obstaja NTM N nedeterministične prostorske zahtevnosti S(n), da velja L = L(N).
    NSPACE(S(n)) = {L | L je jezik ∧ L je nedeterministične prostorske zahtevnosti S (n)}

  - **odločitveni problem in razred vseh takih odločitvenih problemov**

    Odločitveni problem D je nedeterministične prostorske zahtevnosti S(n), če ima njegov jezik L(D) nedeterministično prostorsko zahtevnost S(n).
    NSPACE(S(n)) = {D | D je odločitveni problem ∧ D je nedeterministične prostorske zahtevnosti S (n)}

- **stiskanje trakov, linearna pohitritev, zmanjševanje števila trakov**

  Prostorsko/časovno zahtevnost je lahko vedno zmanjšana z konstantnim faktorjem (z zakodiranjem večih tračnih simbolov v enega/z združitvijo nekaj korakov v enega).

  - **stiskanje trakov** (prostor)

    Če ima L prostorsko zahtevnost **S(n)**, potem za vsak **c > 0**, ima L časovno zahtevnost **c S(n).**
    Za vsak c > 0 je **DSPACE(S(n)) = DSPACE(cS(n))** in **NSPACE(S(n)) = NSPACE(cS(n))**.

  - **linearna pohitritev** (čas)

    Izpolnjena morata biti **2 pogoja**: k > 1 in $\inf_{n\to\infty} T(n)/n = \infty$ (T(n) raste hitreje kot n).

    Če ima L časovno zahtevnosti **T(n)**, potem ima za vsak **c > 0**, L časovno zahtevnost **cT(n).**
    Če je inf T(n) /n = ∞, potem za vsak c > 0 velja:
    DTIME(T(n)) = DTIME(cT(n)) in **NTIME(T(n)) = NTIME(cT(n))**.

    DTIME(n^2) = DTIME($4n^2$) = ... D ima deterministično časovno zahtevnost reda O(n^2).

  - **zmanjševanje števila trakov**
    - **zmanjševanje števila k vpliva na časovno zahtevnost**

      Če je $L \in$ DTIME($T(n)$), potem je L sprejet v času **O($T^2(n)$)** z 1-tračnim TM.
      Če je $L \in$ NTIME($T(n)$), potem je L sprejet v času **O($T^2(n)$)** z 1-tračnim NTM.
      Če je $L \in$ DTIME($T(n)$), potem je L sprejet v času **O($T(n) \log T(n)$)** z 2-tračnima TM.
      Če je $L \in$ NTIME($T(n)$), potem je L sprejet v času **O($T(n) \log T(n)$)** z 2-tračnima NTM.

    - **zmanjševanje števila k ne vpliva na prostorsko zahtevost**

      Če je L sprejet z **k**-delovnimi traki TM prostorske zahtevnosti S(n), potem je L sprejet z **1**-delovnim trakom TM prostorske zahtevnosti S(n).

- **relacije med DTIME, DSPACE, NTIME in NSPACE**

  Zamenjava ned. algoritma z det. povzroči največ **eksponento** rast v časovni zahtevnosti in največ **kvadratno** rast v prostorski zahtevnosti.

  - **relacije med različnimi razredi zahtevnosti**

    DTIME(T(n)) ⊆ DSPACE(T(n))

    Kar je lahko rešljivo v *času* O($T(n)$), je lahko rešljivo tudi na *prostoru* O($T(n)$).

    $L \in DSPACE(S(n)) \wedge S(n) \geq \log_2 n \Rightarrow \exists c : L \in DTIME(c^{S(n)})$

    Kar je lahko rešljivo na *prostoru* O($S(n)$), je lahko tudi rešljivo *(največ) v času* O($c^{S(n)}$). (Tukaj je c odvisen od L.)

    $L \in NTIME(T(n)) \Rightarrow \exists c : L \in DTIME(c^{T(n)})$

    Kar je lahko rešljivo v *nedeterminističnem času* O($T(n)$), je lahko rešljivo v *največ determinističnem času* O($c^{T(n)}$). Zamenjava nedeterminističnega algoritma z determinističnim povzroči največ eksponentno rast potrebnega časa za rešitev problema.

    NSPACE($S(n)$) ⊆ DSPACE($S^2(n)$), *if* $S(n) \geq \log_2 n \wedge S(n)$ je lepa, pohlevna funkcija

    Kar je lahko rešljivo na *nedeterminističnem prostoru* O($S(n)$), je lahko rešljivo na *determinističnem prostoru* O($S^2(n)$). Posledično, zamenjava nedeterminističnega algoritma z determinističnim povzroči največ kvadratno rast na prostoru, ki je potreben za reševanje problema.

  - **lepe, pohlevne funkcije zahtevnosti**
    - **prostorsko predstavljiva funkcija S(n)**

      Funkcija S(n) je **prostorsko predstavljiva**, če obstaja TM M prostorske zahtevnosti S(n), da za vsak n, obstaja vhod dolžine n na katerem M uporabi točno S(n) tračnih celic.
      Funkcija S(n) je **popolnoma prostorsko predstavljiva**, če za vsak n, M uporabi točno S(n) celic za vsak vhod dolžine n, potem rečemo.

    - **časovno predstavljiva funkcija T(n)**

      Funkcija T(n) je **časovno predstavljiva**, če obstaja TM M časovne zahtevnosti T(n), da za vsak n, obstaja vhod dolžine n na katerem M uporabi točno T(n) prehodov.
      Funkcija S(n) je **popolnoma časovno predstavljiva**, če za vse n, M naredi točno T(n) prehodov za vsak vhod dolžine n.

- **razredi zahtevnosti P, NP, PSPACE, NPSPACE**

  Razredi zahtevnosti DTIME(T(n)), NTIME(T(n)), DSPACE(S(n)), NSPACE(S(n)), ki imajo funkcije zahtevnosti T(n) in S(n) so polinomi.

  P = $\cup_{i \geq 1}$**DTIME(**n^i**)**

  je razred vseh odločitvenih problemov rešljivih *v deterističnem polinomskem času*

  NP = $\cup_{i \geq 1}$**NTIME(**n^i**)**

  je razred vseh odločitvenih problemov rešljivih *nedeterministično v polinomskem času*

  PSPACE = $\cup_{i \geq 1}$**DSPACE(**n^i**)**

  je razred vseh odločitvenih problemov *determinističmo* rešljivih na *polinomskem prostoru*

  NPSPACE = $\cup_{i \geq 1}$**NSPACE (**n^i**)**

  je razred vseh odločitvenih problemov *nedeterministično* rešljivih na *polinomskem prostoru*

  i (1 do neskončnosti, velikost primerka, ki se rešuje; npr. n^2)

- **osnovne relacije med P, NP, PSPACE, NPSPACE**

  P $\subseteq$ **NP** $\subseteq$ **PSPACE = NPSPACE**

  (P $\subseteq$ **NP)**

  Vsak deterministični TM polinomske časovne zahtevnosti si lahko predstavljamo kot (trivialni) *ne*deterministični TM enake časovne zahtevnosti.

  (**NP** $\subseteq$ **PSPACE**)

  Če je $L \in$ NP, potem obstaja tak $\exists k$, *da je* $L \in$ NTIME(n^k).

  Zaradi izreka je $L \in$ NSPACE(n^k), in posledično (Savitch) je $L \in$ DSPACE(n^{2k}). Potem je $L \in$ PSPACE.

  (**PSPACE = NPSPACE**)

  PSPACE $\subseteq$ NPSPACE

  Druga smer: NSPACE = (def) = $\cup$ NSPACE(n^i) $\subseteq$ (Savitch) $\cup$ DSPACE(n^i) $\subseteq$ PSPACE.

  Če je prostorska zahtevnost polinomska, nedeterminizem ne doda nič k računski moči.

- **NP-poln in NP-težek problem**
  - **polinomsko-časovna prevedba**

    Problem $D \in$ NP je **polinomsko-časovno prevedljiv** na problem D', $\boldsymbol{D \leq_p D'}$, če obstaja deterministični TM *M* polinomske časovne zahtevnosti, ki za vsak $d \in D$, vrne $d' \in D'$, tako, da je, *d pozitiven* $\iff$ *d' je pozitiven*. Relacija $\leq_p$ je **polinomsko-časovna prevedba**.

  - **NP-težek problem**

    Problem D* je **NP-težek**, če D ≤p D*, za vsak D ∈ NP (D* je ali ni v NP).

  - **NP-poln problem**

    Problem D* je **NP -poln**:

    - $D^* \in$ NP
    - $D \leq_p D^*$, za vsak $D \in$ NP (največ tako težek kot D*)

    Primer: problem SAT (problem izpolnjivosti), particije

    D* je NP-poln, če je D* v NP in D* je NP-težek.

  - **dokazovanje NP-polnih problemov**

    Naj bo D ≤p D'. Potem:

    - D' ∈ P ⇒ D ∈ P
    - D' ∈ NP ⇒ D ∈ NP

    Vsak problem D, ki je lahko ≤**p-zmanjšan** na problem v P (ali v NP), je tudi v P (ali v NP).

    Relacija ≤**p** je tranzitivna: D ≤ p D' ∧ D' ≤ p D'' ⇒ D ≤ p D''.

    D* je NP-težek ∧ D* ≤p D☆ ⇒ D☆ je NP-težek

    D* je NP-poln ∧ D* ≤p D ☆ ∧ D☆∈ NP ⇒ D☆ je NP-poln

  - **P ≠ NP**

    NPC je razred vseh NP-polnih problemov.

    NPI je razred vseh NP-vmesnih problemov (problem v NP, ki ni niti v P niti v NPC).

    Noben problem v NPC ali NPI nima zahtevnosti polinomskega časa.

    Problemi v P so obvladljivi. Drugi računski problemi so neobvladljivi (za NPC in NPI ni jasno).