

Univerzita Karlova v Praze  
Matematicko-fyzikální fakulta

## BAKALÁŘSKÁ PRÁCE



Richard Eliáš

## Vizualizace sekundární struktury RNA s využitím existujících struktur

Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. David Hoksza, Ph.D.

Studijní program: Informatika

Studijní obor: Obecná informatika

Praha 2016

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne .....

Podpis autora

Název práce: Vizualizace sekundární struktury RNA s využitím existujících struktur

Autor: Richard Eliáš

Katedra: Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. David Hoksza, Ph.D., Katedra softwarového inženýrství

Abstrakt: Abstrakt .. TODO

Klíčová slova: TODO klíčová slova

Title: RNA secondary structure visualization using existing structures

Author: Richard Eliáš

Department: Department of Software Engineering

Supervisor: RNDr. David Hoksza, Ph.D., Department of Software Engineering

Abstract: RNA secondary structure data, both experimental and predicted, are becoming increasingly available which is reflected in the increased demand for tools enabling their analysis. The common first step in the analysis of RNA molecules is visual inspection of their secondary structure. In order to correctly lay out an RNA structure, the notion of optimal layout is required. However, optimal layout of RNA structure has never been formalized and is largely habitual. To tackle this problem we propose an algorithm capable of visualizing an RNA structure using a related structure with a well-defined layout. The algorithm first converts both structures into a tree representation and then uses tree-edit distance algorithm to find out the minimum number of tree edit operations to convert one structure into the other. We couple each tree edit operation with a layout modification operation which is then used to gradually transform the known layout into the target one. The optimality of tree edit distance algorithm causes that the common motives are retained and the regions which differ in both the structures are taken care of. Visual inspection and planarity evaluation reveals that the algorithm is able to give good layouts even for relatively distant structures while keeping the layout planar. The new method is well suited for situations when one needs to visualize a structure for which a homologous structure with a good visualization is already available. ii

Keywords: RNA secondary structure, visualization, homology

Poděkování.

# Obsah

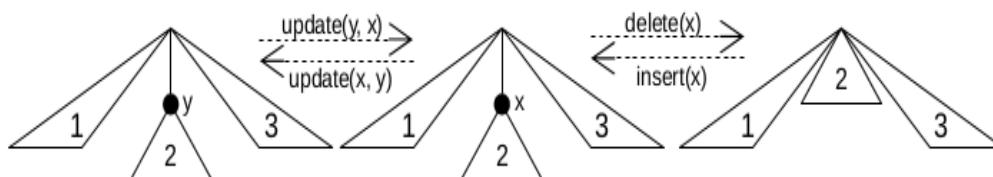
<b>1</b>	<b>Tree-edit-distance algoritmus</b>	<b>2</b>
1.1	Hlavná myšlienka TED-u . . . . .	2
1.2	Znacie . . . . .	2
1.3	Algoritmy dynamickeho programovania . . . . .	3
1.3.1	RTED . . . . .	3
1.4	Mapovanie medzi stromami . . . . .	8
<b>2</b>	<b>Kreslenie molekuly</b>	<b>11</b>
<b>3</b>	<b>Nápověda k sazbě</b>	<b>12</b>
3.1	Úprava práce . . . . .	12
3.2	Jednoduché příklady . . . . .	12
3.3	Matematické vzorce a výrazy . . . . .	13
3.4	Definice, věty, důkazy, ... . . . .	14
<b>4</b>	<b>Odkazy na literaturu</b>	<b>16</b>
4.1	Několik ukázek . . . . .	16
<b>5</b>	<b>Tabulky, obrázky, programy</b>	<b>17</b>
5.1	Tabulky . . . . .	17
5.2	Obrázky . . . . .	18
5.3	Programy . . . . .	18
	<b>Závěr</b>	<b>23</b>
	<b>Seznam použité literatury</b>	<b>24</b>
	<b>Zoznam obrázkov</b>	<b>26</b>
	<b>Zoznam tabuliek</b>	<b>27</b>
	<b>Seznam použitých zkratek</b>	<b>28</b>
	<b>Přílohy</b>	<b>29</b>

# 1. Tree-edit-distance algoritmus

Jadro aplikacie lezi v pouziti tree-edit-distance (TED) algoritmu, vďaka ktorému dostaneme mapovanie medzi 2 RNA stromami. Mapovanie nam ukáže spoločne časti oboch RNA stromov. TED algoritmus je obdoba Levensteinovoho string-edit-distance algoritmu. Problém u retazcov je špeciálnym prípadom TED-u, kedy stromy zdegenerovali na cesty (spojový zoznam).

## 1.1 Hlavná myšlienka TED-u

Základ TED algoritmu je v rekurzívnom vzorci ?? z Demaine a kol. (2009) a Pawlik a Augsten (2011). Vzdialenosť medzi lesmi  $F$  a  $G$ ,  $\delta(F, G)$  je definovaná ako minimálny počet editačných operácií, ktoré z  $F$  urobia  $G$ . Používame štandardné editačné operácie - delete, insert, update.



Obr. 1.1: Ukazky TED operácií

Delete, zmazanie vrcholu, znamená pripojiť k predkovi všetkých jeho potomkov so zachovaním poradia medzi nimi. Insert, vloženie vrcholu, je opačná operácia k delete, čo znamená, že vkladáme vrchol medzi rodiča nejakých jeho, po sebe nasledujúcich potomkov. Update iba zmení hodnotu vo vrchole stromu.

## 1.2 Znacenie

V tejto kapitole sa budeme riadiť značením Pawlik a Augsten (2011). Teda, používame definíciu stromu a lesa z ?? . Ak  $F$  je les (strom),  $N_F$  označuje množinu jeho vrcholov a  $E_F$  množinu jeho hran. Platí ďalej že  $E_F \subseteq N_F \times N_F$ .  $\emptyset$  označuje prázdny strom, resp. prázdny les. Podľa lesa  $F$  je graf  $\bar{F}$  s vrcholmi  $N_{\bar{F}} \subseteq N_F$  a hranami  $E_{\bar{F}} \subseteq E_F \cap N_{\bar{F}} \times N_{\bar{F}}$ . Obdobne to platí aj pre podstrom stromu  $T$ .  $F_v$  označuje podstrom  $F$  zakorenený vo  $v$ , t.j. v strome ostávajú iba potomkovia  $v$ .  $F - v$  budeme znčiť les, ktorý dostaneme zmazaním vrcholu  $v$  z  $F$ , spolu so všetkými hranami zasahujúcimi do  $v$ . Podobne  $F - F_v$  budeme znčiť les, ktorý dostaneme zmazaním podstromu  $F_v$  z  $F$ .

**Definícia 1** (Editačná vzdialenosť). *Nech  $F$  a  $G$  sú dva lesy. Editacia vzdialenosť, tree-edit-distance -  $\delta(F, G)$ , medzi  $F$  a  $G$  je rovná minimálnej cene, za ktorú les  $F$  transformujeme na  $G$ .*

Vo vzorci ?? počítame editačnú vzdialenosť  $\delta(F, G)$ ,  $c_{del}$ ,  $c_{ins}$  a  $c_{upd}$  sú ceny zmazania, vloženia a editácie vrcholu v strome a  $r_F$  a  $r_G$  sú korene, buď obidva

najpravejšie alebo najlavejšie (tzn. vyberieme najpravejši/najlavejši strom lesa a jeho koren).

$$\begin{aligned}\delta(\emptyset, \emptyset) &= 0 \\ \delta(F, \emptyset) &= \delta(F - r_F, \emptyset) + c_{del}(r_F) \\ \delta(\emptyset, G) &= \delta(\emptyset, G - r_G) + c_{ins}(r_G)\end{aligned}\tag{1.1a}$$

$$\delta(F, G) = \begin{cases} \delta(F - r_F, G) + c_{del}(r_F) \\ \delta(F, G - r_G) + c_{ins}(r_G) \\ \delta(F - F_{r_F}, G - G_{r_G}) + \\ \delta(F_{r_F} - r_F, G_{r_G} - r_G) + c_{upd}(r_F, r_G) \end{cases}\tag{1.1b}$$

Obr. 1.2: Rekurzivny vzorec pre vypocet tree-edit-distance

## 1.3 Algoritmy dynamickeho programovania

Tai (1979) predstavil algoritmus s priestorovou a casovou zlozitostou  $\mathcal{O}(m^3 \cdot n^3)$ , Zhang a Shasha (1989) algoritmus nasledne vylepsili pozorovanim toho, ze nepotrebuje vzdialenosti medzi vsetkymi parmi podlesov. Algoritmus mal casovu zlozitost  $\mathcal{O}(m^2 \cdot n^2)$  a priestorovu  $\mathcal{O}(m \cdot n)$ . Klein (1998) dosiahol casovu zlozitost  $\mathcal{O}(m^2 \cdot n \cdot \log n)$ , avsak jeho riesenie potrebovalo rovnako vela pamete. Dulucq a Touzet (2003) ukazali, ze minimalny cas na beh algoritmu je  $\mathcal{O}(m \cdot n \cdot \log m \cdot \log n)$ . Demaine a kol. (2009) predviedli worst-case optimalny algoritmus pre tree-edit-distance. Jeho casova a priestorova zlozitost je  $\mathcal{O}(m^2 \cdot n \cdot (1 + \log \frac{n}{m}))$  a  $\mathcal{O}(m \cdot n)$ . Pawlik a Augsten (2011) ukazali spojitosť medzi efektivnostou predchadzajucich algoritmov a tvarom stromov. Zovseobecniili predchadzajuce pristupy a vytvorili algoritmus beziaci vo worst-case case  $\mathcal{O}(m^3)$  a priestore  $\mathcal{O}(m \cdot n)$ . Ich algoritmus je teda efektivny pre vsetky tvary stromov a nikdy nespadne do worst-case, ak existuje lepsi smer vypoctu.

### 1.3.1 RTED

Dalej sa v nasej praci budeme venovat vyhradne algoritmu RTED od tvorcov Pawlik a Augsten (2011). Ich algoritmus rozdelime na 2 casti, rovnako pomenovany RTED a GTED.

RTED (Robust Tree Edit Distance) algoritmus bude pre nas algoritmus na vypocet optimalnej dekompozicnej strategie (viz definicia 2) a GTED (General Tree Edit Distance) algoritmus samotny vypocet rekuzie ?? s aplikovanim danej strategie.

**Definícia 2** (Dekompozicna strategia). *Nech  $F$  a  $G$  su lesy. Dekompozicna strategia v rekuzii ?? priradi kazdej dvojici podstromov  $F_v$  a  $G_w$  lesov  $F$  a  $G$  jednu cestu  $\gamma_T$  z korena do listu, kde  $T \in \{F, G\}$ . LRH dekompozicna strategia vybera vzdy najlavejši/najpravejši/najtazsi (left/right/heavy) vrchol na ceste z korena do listu. Najtazsi vrchol je taky v ktoreho podstrome je najviac vrcholov.*

## GTED: General Tree Edit Distance algoritmu

Zacneme principom fungovania GTED algoritmu. Detaily pre LRH strategie su v Zhang a Shasha (1989) pre left/right a v Demaine a kol. (2009) pre heavy strategiu.

---

### Algorithm 1 General Tree Edit Distance for LRH strategies

---

```

1: procedure GTED( $F, G, TreeDistance, S$ )
2:    $\sigma \leftarrow S[F, G]$ 
3:   if  $\sigma \in \sigma^*(F)$  then
4:     for all  $F' \in F - \sigma$  do
5:        $TreeDistance \leftarrow TreeDistance \cup \text{GTED}(F', G, TreeDistance, S)$ 
6:      $TreeDistance \leftarrow TreeDistance \cup$ 
7:        $\text{COMPUTE\_DISTANCE}(F, G, TreeDistance, \sigma)$ 
8:   else
9:      $TreeDistance \leftarrow TreeDistance \cup (\text{GTED}(G, F, TreeDistance^T, S^T))^T$ 
10:  return  $TreeDistance$ 

```

---

*Poznámka.* Funkcia *GetOrderedSubforests()* v algoritme 2 vracia lesy zoradene v opacnom poradí, ako ich pridavame v definícii 3.

Algoritmus 1 funguje v troch krokoch.

Najprv podľa stratégie dekomponuje jeden zo stromov podľa cesty  $\gamma$ , bez ujmy na obecnosti, nech je to  $F$  a rekurzívne spočíta editacnú vzdialenosť medzi všetkými podstromami ktoré susedia s dekompozícnou cestou a stromom  $G$ .

Následne pre všetky relevant-subtrees (viz definície 3) podstromy  $G'$  stromu  $G$  vyrata vzdialenosti medzi  $F_v$  a  $G'$  pomocou single-path funkcie. Tá dopocita vzdialenosti medzi vrcholmi  $v \in \gamma_F$  a stromami  $G'$ .

**Definícia 3.** *Relevant subtrees stromu  $F$  pre root-leaf cestu  $\gamma$  su definovane ako  $F - \gamma$ . Relevant subforests stromu  $F$  pre nejaku root-leaf cestu  $\gamma$  su definovane rekurzívne ako*

$$\begin{aligned}
\mathcal{F}(\emptyset, \gamma) &= \emptyset \\
\mathcal{F}(F, \gamma) &= \{F\} \cup \begin{cases} \mathcal{F}(F - r_R(F), \gamma), & \text{ak } r_L(F) \in \gamma \\ \mathcal{F}(F - r_L(F), \gamma), & \text{v ostatných prípadoch} \end{cases}
\end{aligned}$$

**Lemma 1.** *Ak compute-distance funkcia dopocita editacnú vzdialenosť medzi vrcholmi na ceste  $\gamma$  a všetkými podstromami druhého stromu, potom GTED vrati maticu vzdialenosti medzi všetkými dvojicami podstromov  $F_v$  a  $G_w$ , pre  $v \in F; w \in G$ .*

*Dôkaz.* Nech  $\gamma \in F$ . Po vyrataní editacnej vzdialenosti medzi stromami  $F - \gamma$  a  $G$  nám staci dopocitať už len vrcholy na ceste, teda vzdialenosti medzi stromami  $F_v$  a  $G$  pre  $v \in \gamma_F$ .

□

Vďaka doslednému usporiadaniu lesov si v každom kroku pripravíme potrebné data pre ďalší krok algoritmu 2.



---

**Algorithm 2** Single path function

---

```
1: procedure COMPUTE DISTANCE( $F, G, TreeDistance, \sigma$ )
2:   if  $\sigma \in \sigma^*(F)$  then
3:     for all  $G' \in \text{RELEVANT SUBTREES}(G)$  do
4:       SINGLE PATH( $F, G', TreeDistance, \sigma$ )
5:   else
6:     for all  $F' \in \text{RELEVANT SUBTREES}(F)$  do
7:       SINGLE PATH( $F, G, TreeDistance, \sigma$ )
8:
9: procedure SINGLE PATH( $F, G, TreeDistance, \sigma$ )
10:   $ForestDistance \leftarrow$  empty array  $|F| + 1 \times |G| + 1$ 
11:   $ForestDistance[\emptyset][\emptyset] := 0$ 
12:  for  $F'$  subforest in GET ORDERED SUBFORESTS( $F, \sigma$ ) do
13:     $Last_F \leftarrow$  last added node to  $F'$ 
14:     $ForestDistance[F'][\emptyset] := ForestDistance[F' - Last_F][\emptyset] +$ 
15:       $C_{del}(Last_F)$ 
16:  for  $G'$  subforest in GET ORDERED SUBFORESTS( $G, \sigma$ ) do
17:     $Last_G \leftarrow$  last added node to  $G'$ 
18:     $ForestDistance[\emptyset][G'] := ForestDistance[\emptyset][G' - Last_G] +$ 
19:       $C_{ins}(Last_G)$ 
20:  for  $F'$  subforest in GET ORDERED SUBFORESTS( $F, \sigma$ ) do
21:    for  $G'$  subforest in GET ORDERED SUBFORESTS( $G, \sigma$ ) do
22:       $Last_F \leftarrow$  last added node to  $F'$ 
23:       $Last_G \leftarrow$  last added node to  $G'$ 
24:      if both  $F'$  and  $G'$  are trees then
25:         $C_{min} := \min\{$ 
26:           $ForestDistance[F' - Last_F][G'] +$ 
27:           $C_{del}(Last_F),$ 
28:           $ForestDistance[F'][G' - Last_G] +$ 
29:           $C_{ins}(Last_G),$ 
30:           $ForestDistance[F' - Last_F][G' - Last_G] +$ 
31:           $C_{upd}(Last_F, Last_G)$ 
32:         $ForestDistance[F', G'] := C_{min}$ 
33:         $TreeDistance[Last_F][Last_G] := C_{min}$ 
34:      else
35:         $C_{min} := \min\{$ 
36:           $ForestDistance[F' - Last_F][G'] +$ 
37:           $C_{del}(Last_F),$ 
38:           $ForestDistance[F'][G' - Last_G] +$ 
39:           $C_{ins}(Last_G),$ 
40:           $ForestDistance[F' - F_{Last_F}][G' - G_{Last_G}] +$ 
41:           $TreeDistance[F_{Last_F}][G_{Last_G}]\}$ 
42:         $ForestDistance[F'][G'] := C_{min}$ 
```

---

Najprv si este ale vysvetlime hodnoty pouzivane v algoritme 2 v podmienkach na riadkoch 24 a 34. Prve dva su v oboch rovnake. Pocitame hodnotu zmazania vrcholu z  $F$ , resp. vlozenia vrcholu do  $F$ .

Tretia hodnota sa lisi podla toho, ci su lesy zaroven aj stromami. Ak su, tak na danom mieste je cena namapovania podstromov  $F_v - v$  na  $F_w - w$  a updatu vrcholu  $v$  na  $w$ . Inac, ked aspon jeden z lesov nieje stromom, tak cenu medzi  $F_{Last_F}$  a  $G_{Last_G}$  mame vyratanu z predchadzajucich krokoch, alebo z inej vetvy rekurzcie.

Potom nastavime hodnotu vzdialenosti medzi lesmi na minimum a v pripade ze su to obidva stromy, tak nastavime aj ich vzdialenost.

Najprv este ukazeme, ze SPF pouziva vzdy inicializovane hodnoty, a kazdu hodnotu nastavuje prave raz.

*Poznámka.* Nikdy nepouzivam 2x rovnaku cestu  $\gamma$  v strome. To vyplyva z toho, ze po dekompozicii stromu podla  $\gamma$ , cesta v ostatnych stromoch neexistuje.

*Poznámka.* Single-path funkcia kazdu hodnotu *ForestDistance*, rovnako ako *TreeDistance* nastavuje prave raz.

*Dôkaz.* Ziadnu cestu nepouzivam opakovane. Hodnotu v *TreeDistance* nastavujem iba v momente, ked su obidva lesy stromami (teda ich korene lezia na cestach  $\gamma_F$  a  $\gamma_G$ ) a to sa udeje prave raz. Lesy vzdy iba zvacsujem, takže nikdy sa nedostanem do mensieho aby som mohol mu znovu nastavit hodnotu. To iste plati aj pre *ForestDistance*.

□

**Lemma 2.** *Nikdy nepouzivame neinicializovane hodnoty *TreeDistance* a *ForestDistance*.* ■

*Dôkaz.* Hodnota *ForestDistance* pre pouzitie s prazdnym lesom je inicializovana, a pri kazdej iteracii algoritmu citam iba z hodnot z predchadzajucich iteracii, napr *ForestDistance* $[F - Last_F][G - Last_G]$ , alebo *ForestDistance* $[F - F_{Last_F}][G - G_{Last_G}]$ . V prvom pripade mazem iba jeden vrchol, v druhom cely jeho podstrom.

Hodnoty *TreeDistance* pouzivame iba v pripade, ze aspon jeden z lesov  $F'$  alebo  $G'$  nieje stromom. To znamena, ze ak posledne pridany vrchol  $Last_F$  je mimo cesty  $\gamma_F$ , tak sme vzdialenost od  $Last_G$  vyratali rekurzivne po dekompozicii  $F$  uz skor. Naopak ak  $Last_F$  lezi na ceste, potom  $Last_G$  je mimo cesty, a editacnu vzdialenost sme vyratali pri pocitani relevant-subtrees.

□

*Dôsledok.* Algoritmus funguje.

*Dôkaz.* V predchadzajucich castiach sme dokazali, ze v kazdom kroku pouzivame iba korektne hodnoty a vsetky casti algoritmu pocitaju spravne, takže algoritmus GTED je v poriadku.

□

## RTED: Robust Tree Edit Distance algoritmus

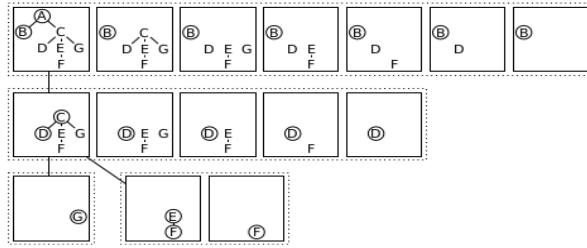
RTED budeme vnímať ako algoritmus na výpočítanie optimálnej stratégie - teda algoritmus, ktorý nám poradí ako najlepšie dekomponovať obidva stromy.

Funguje tak, že si predpocítajú koľko podproblémov budeme musieť vyriešiť, ak použijeme stratégiu *left*, *right*, alebo *heavy*.

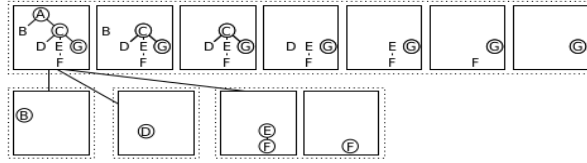
**Definícia 4.** Celková dekompozícia lesa (full decomposition)  $F$ ,  $\mathcal{A}(F)$  je množina všetkých podlesov  $F$ , ktoré dostaneme rekurzívnym odstránením najľavejšieho alebo najpravejšieho koreňového vrcholu -  $r_L(F)$  a  $r_R(F)$  - z  $F$  a následne aj všetkých jeho podlesov.

$$\mathcal{A}(\emptyset) = \emptyset$$

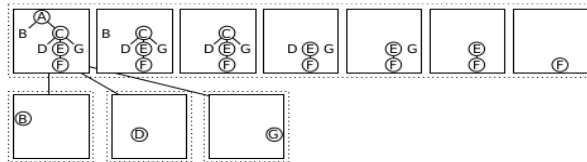
$$\mathcal{A}(F) = F \cup \mathcal{A}(F - r_L(F)) \cup \mathcal{A}(F - r_R(F))$$



(a) Left path decomposition (15 relevant subforests)



(b) Right path decomposition (11 relevant subforests)



(c) Heavy path decomposition (10 relevant subforests)

Obr. 1.3: Celková dekompozícia pomocou LRH stratégie

**Lemma 3.** Počet podproblémov (relevant-subproblems) počítaných single-path funkciou pre dvojicu stromov  $F$  a  $G$  je rovná

$$\# = \begin{cases} |F| \times |\mathcal{F}(G, \Gamma^L(G))| & \text{pre left-paths} \\ |F| \times |\mathcal{F}(G, \Gamma^R(G))| & \text{pre right-paths} \\ |F| \times |\mathcal{A}(G)| & \text{pre heavy-paths} \end{cases}$$

*Dôkaz.* Demaine a kol. (2009) dokázali, že vzorec pre tazku cesty je v poriadku. Rovnako tak, Zhang a Shasha (1989) to dokázali pre ľavé cesty. Jednoduchou úpravou vieme upraviť ich vzorec na použitie pravých ciest.

□

**Definícia 5.** *Minimalny počet podproblemov ktore potrebujeme vyratat pri pouziti GTEDu je*

$$cena(F, G) = \begin{cases} |F| \times |\mathcal{A}(G)| & + \sum_{F' \in F - \gamma^H(F)} cena(F', G) \\ |G| \times |\mathcal{A}(F)| & + \sum_{G' \in G - \gamma^H(G)} cena(G', F) \\ |F| \times |\mathcal{F}(G, \Gamma^L(G))| & + \sum_{F' \in F - \gamma^L(F)} cena(F', G) \\ |G| \times |\mathcal{F}(F, \Gamma^L(F))| & + \sum_{G' \in G - \gamma^L(G)} cena(G', F) \\ |F| \times |\mathcal{F}(G, \Gamma^R(G))| & + \sum_{F' \in F - \gamma^R(F)} cena(F', G) \\ |G| \times |\mathcal{F}(F, \Gamma^R(F))| & + \sum_{G' \in G - \gamma^R(G)} cena(G', F) \end{cases}$$

*Dôkaz.* je uvedený v Pawlik a Augsten (2011)

□

Namiesto  $\mathcal{O}(n^3)$  rekurzie potrebujeme algoritmus, ktorý optimalnú stratégiu vyrata s nízskimi časovými nárokmi ako potrebuje optimalný beh GTEDu.

Popiseme teda algoritmus 3 - RTED, od tvorcov Pawlik a Augsten (2011). Beziaci v case  $\mathcal{O}(n^2)$ .

Prechádza vrcholmi v postorder, aby sa znížila pamäťová náročnosť algoritmu a nemuseli ukladať hodnoty medzi dvojicami relevant-subforest. Namiesto toho inkrementujeme hodnotu v rodičovskom vrchole pri každej navsteve jeho potomka.

**Lemma 4.** *Algoritmus 3 vyrata optimalnú LRH stratégiu pre dvojicu podstromov  $F$  a  $G$  a časová náročnosť algoritmu je  $\mathcal{O}(n^2)$ .*

*Dôkaz.* Toto tvrdenie dokazali Pawlik a Augsten (2011).

□

## 1.4 Mapovanie medzi stromami

Tabuľka vzdialenosti z GTEDu medzi stromami  $F$  a  $G$  nám nebude stačiť. Potrebujeme vedieť ako strom  $F$  namapovať na  $G$ .

Princíp je v backtrackovaní matice *ForestDistance*, teda zisťujeme, akú operáciu sme v ktorom bode použili, podobne ako v zisťovaní operácií pri editačnej vzdialenosti retazcov. Musíme ale používať *ForestDistance* maticu, nie *TreeDistance*, keďže v nej sa odzrkadľuje detailnejšia štruktúra stromov. Maticu *TreeDistance* používame iba na počítanie single-path funkcie. Algoritmus mapovania 4

---

**Algorithm 3** Optimalna strategija
 

---

```

1: procedure RTED( $F, G$ )
2:    $L_v, R_v, H_v \leftarrow$  polia velkosti  $|F| \times |G|$ 
3:    $L_w, R_w, H_w \leftarrow$  polia velkosti  $|G|$ 
4:   for all  $v$  postorder v  $F$  do
5:     for all  $w$  postorder v  $G$  do
6:       if  $v$  je list then
7:          $L_v[v, w] \leftarrow R_v[v, w] \leftarrow H_v[v, w] \leftarrow 0$ 
8:       if  $w$  je list then
9:          $L_w[w] \leftarrow R_w[w] \leftarrow H_w[w] \leftarrow 0$ 
10:       $C := \{$ 
11:         $(|F_v| \times \mathcal{A}(G_w) + H_v[v, w], \gamma^H(F)),$ 
12:         $(|G_w| \times \mathcal{A}(F_v) + H_w[w], \gamma^H(G)),$ 
13:         $(|F_v| \times |\mathcal{F}(G_w, \Gamma^L(G))| + L_v[v, w], \gamma^L(F)),$ 
14:         $(|G_w| \times |\mathcal{F}(F_v, \Gamma^L(F))| + L_w[w], \gamma^L(G)),$ 
15:         $(|F_v| \times |\mathcal{F}(G_w, \Gamma^R(G))| + R_v[v, w], \gamma^R(F)),$ 
16:         $(|G_w| \times |\mathcal{F}(F_v, \Gamma^R(F))| + R_w[w], \gamma^R(G))$ 
17:       $\}$ 
18:       $(c_{min}, \gamma_{min}) \leftarrow (c, \gamma)$  take, ze  $(c, \gamma) \in C \wedge c = \min\{c' | (c', \gamma) \in C\}$ 
19:       $Strategies[v, w] := \gamma_{min}$ 
20:      if  $v$  nije koren then
21:        UPDATE( $L_v, v, w, c_{min}, \gamma^L(parent(v))$ )
22:        UPDATE( $R_v, v, w, c_{min}, \gamma^R(parent(v))$ )
23:        UPDATE( $H_v, v, w, c_{min}, \gamma^H(parent(v))$ )
24:      if  $w$  nije koren then
25:        UPDATE( $L_w, w, c_{min}, \gamma^L(parent(w))$ )
26:        UPDATE( $R_w, w, c_{min}, \gamma^R(parent(w))$ )
27:        UPDATE( $H_w, w, c_{min}, \gamma^H(parent(w))$ )
28:      return  $Strategies$ 

29: procedure UPDATE( $Table, v, w, c_{min}, \gamma$ )
30:    $Table[parent(v), w] \stackrel{\pm}{=} \begin{cases} Table[v, w] & \text{ak } v \in \gamma \\ c_{min} & \text{v opacnom pripade} \end{cases}$ 

31: procedure UPDATE( $Table, w, c_{min}, \gamma$ )
32:    $Table[parent(w)] \stackrel{\pm}{=} \begin{cases} Table[w] & \text{ak } v \in \gamma \\ c_{min} & \text{v opacnom pripade} \end{cases}$ 

```

---

---

**Algorithm 4** Pocitanie mapovania

---

```
1: procedure MAPPING( $F, G, TreeDistance$ )
2:    $\sigma \leftarrow$  lubovolna LRH strategija
3:    $ForestDistance \leftarrow$  SINGLE PATH( $F, G, TreeDistance, \sigma$ )
4:   while  $F \neq \emptyset \wedge G \neq \emptyset$  do
5:      $v \leftarrow$  UPDATE( $F, \sigma$ )
6:      $w \leftarrow$  UPDATE( $G, \sigma$ )
7:     if  $ForestDistance[F, G] = ForestDistance[F - v, G] + C_{del}$  then
8:        $Mapping \leftarrow Mapping \cup (v \rightarrow 0)$ 
9:        $F \leftarrow F - v$ 
10:    else if  $ForestDistance[F, G] = ForestDistance[F, G - w] + C_{ins}$  then
11:       $Mapping \leftarrow Mapping \cup (0 \rightarrow w)$ 
12:       $G \leftarrow G - w$ 
13:    else
14:      if  $F$  a  $G$  su stromy then
15:         $Mapping \leftarrow Mapping \cup (v \rightarrow w)$ 
16:         $F \leftarrow F - v$ 
17:         $G \leftarrow G - w$ 
18:      else
19:         $Mapping \leftarrow Mapping \cup$ 
20:          MAPPING( $F - F_v, G - G_w, TreeDistance$ )
21:         $F \leftarrow F - F_v$ 
22:         $G \leftarrow G - G_w$ 

23: procedure UPDATE( $Forest, \sigma$ )
24:    $\gamma \leftarrow$  cesta v lese  $Forest$  podla strategie  $\sigma$ 
25:   return vrchol  $r_L(Forest)$  alebo  $r_R(Forest)$  alebo  $\emptyset$  z  $Forest$ 
26:   rovnako ako v definicii 3
```

---

## 2. Kreslenie molekuly

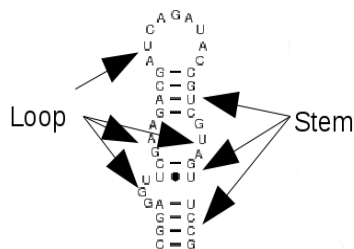
Po tom co ziskame a aplikujeme mapovanie medzi sablonovou a cielovou molekulou RNA, ziskame cielovu molekulu s ciastocnou vizualizaciou, ktorej zvysoz treba dopocitat.

Po operaciach delete ostavaju v molekule prazdne diery, naopak po insertoch potrebujeme vypocitat, kam umiestnime bazovy par, resp. samotnu bazu, pripadne este potrebujeme pre nu urobit miesto. Update vrcholu v strome nerobi ziadne strukturne zmeny, zmeni sa iba nazov bazy na danom mieste.

Sekundarna struktura RNA obsahuje mnozstvo motivov popisanych na obrazku ???. Vo vseobecnosti ale sa kazdy z tychto motivov sklada zo stemu a loopu.

Stemom budeme dalej nazývat cast RNA ktora zodpoveda vnutornemu vrcholu v strome. Loopom budeme označovat listy v RNA strome (lese), nezalezi ci je to bulge, interior loop, hairpin alebo multibranch loop, ako aj ukazuje obrazok 2.1.

Stem zacina vzdy v najvyssom vrchole stromu (v smere ku korenu), ktory je zaroven vnutornym vrcholom a nema ziadnych surodencov, ktory by boli rovnako vnutornymi vrcholmi. To znamena, ze do multibranch loop vchadza 1 stem (ten tu konci) a vychadza z nej niekoľko novych stemov. Naopak pre bulge a interior loopy jeden stem vchadza do struktury ale pokračuje dalej.



Obr. 2.1: Stem a loop v molekule

## 3. Nápořěda k sazřě

### 3.1 Űprava práce

Vlastní text bakalářské práce je uspořádaný hierarchicky do kapitol a podkapitol, každá kapitola začíná na nové straně. Text je zarovnán do bloku. Nový odstavec se obvykle odděluje malou vertikální mezerou a odsazením prvního řádku. Grafická úprava má být v celém textu jednotná.

Práce se tiskne na bílý papír formátu A4. Okraje musí ponechat dost místa na vazbu: doporučen je horní, dolní a pravý okraj 25 mm, levý okraj 40 mm. Číslojí se všechny strany kromě obálky a informačních stran na začátku práce; první číslovaná strana bývá obvykle ta s obsahem.

Písmo se doporučuje dvanáctibodové (12 pt) se standardní vzdáleností mezi řádky (pokud píšete ve Wordu nebo podobném programu, odpovídá tomu řádkování 1,5; v  $\text{\TeX}$ u není potřeba nic přepínat). Pro běžný text používejte vzpřímené patkové písmo. Text matematických vět se obvykle tiskne pro zdůraznění skloněným (slanted) písmem, není-li k dispozici, může být zastoupeno kurzívou.

Primárně je doporučován jednostranný tisk (příliš tenkou práci lze obtížně svázat). Delší práce je lepší tisknout oboustranně a přizpůsobit tomu velikosti okrajů: 40 mm má vždy *vnitřní* okraj. Rub titulního listu zůstává nepotištěný.

Zkratky použité v textu musí být vysvětleny vždy u prvního výskytu zkratky (v závorce nebo v poznámce pod čarou, jde-li o složitější vysvětlení pojmu či zkratky). Pokud je zkratek více, připojuje se seznam použitých zkratek, včetně jejich vysvětlení a/nebo odkazů na definici.

Delší převzatý text jiného autora je nutné vymežit uvozovkami nebo jinak vyznačit a řádně citovat.

### 3.2 Jednoduché příklady

Čísla v českém textu obvykle sázíme v matematickém režimu s desetinnou čárkou:  $\pi \doteq 3,141\,592\,653\,589$ . V matematických textech se považuje za přípustné používat desetinnou tečku (pro lepší odlišení od čárky v roli oddělovače). Numerické výsledky se uvádějí s přiměřeným počtem desetinných míst.

Mezi číslo a jednotku patří úzká mezera: šířka stránky A4 činí 210 mm, což si pamatuje pouze 5 % autorů. Pokud ale údaj slouží jako přívlastek, mezeru vynecháváme: 25mm okraj, 95% interval spolehlivosti.

Rozlišujeme různé druhy pomlček: červeno-černý (krátká pomlčka), strana 16–22 (střední), 45 – 44 (matematické minus), a toto je — jak se asi dalo čekat — vložená věta ohraňčená dlouhými pomlčkami.

V českém textu se používají „české“ uvozovky, nikoliv “anglické”.

Na některých místech je potřeba zabránit lámání řádku (v  $\text{\TeX}$ u značíme vlnovkou): u~předložek (neslabičných, nebo obecně jednopísmenných), vrchol~ $v$ , před  $k$ ~kroky, a~proto, . . . obecně kdekoliv, kde by při rozlomení čtenář „škobrt-nul“.



### 3.3 Matematické vzorce a výrazy

Proměnné sázíme kurzívou (to  $\text{\TeX}$  v matematickém módu dělá sám, ale nezapomínejte na to v okolním textu a také si matematický mód zapněte). Názvy funkcí sázíme vzpřímeně. Tedy například:  $\text{var}(X) = \text{E } X^2 - (\text{E } X)^2$ .

Zlomky uvnitř odstavce (třeba  $\frac{5}{7}$  nebo  $\frac{x+y}{2}$ ) mohou být příliš stísněné, takže je lepší sázet jednoduché zlomky s lomítkem:  $5/7$ ,  $(x+y)/2$ .

Nechť

$$\mathbb{X} = \begin{pmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{pmatrix}.$$

Povšimněme si tečky za maticí. Byť je matematický text vysázen ve specifickém prostředí, stále je gramaticky součástí věty a tudíž je zapotřebí neopomenout patřičná interpunkční znaménka. Výrazy, na které chceme později odkazovat, je vhodné očíslovat:

$$\mathbb{X} = \begin{pmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{pmatrix}. \quad (3.1)$$

Výraz (3.1) definuje matici  $\mathbb{X}$ . Pro lepší čitelnost a přehlednost textu je vhodné číslovat pouze ty výrazy, na které se autor někde v další části textu odkazuje. To jest, nečísľujte automaticky všechny výrazy vysázené některým z matematických prostředí.

Zarovnání vzorců do několika sloupečků:

$$\begin{aligned} S(t) &= \text{P}(T > t), & t > 0 & \quad (\text{zprava spojitá}), \\ F(t) &= \text{P}(T \leq t), & t > 0 & \quad (\text{zprava spojitá}). \end{aligned}$$

Dva vzorce se spojovníkem:

$$\left. \begin{aligned} S(t) &= \text{P}(T > t) \\ F(t) &= \text{P}(T \leq t) \end{aligned} \right\} \quad t > 0 \quad (\text{zprava spojité}). \quad (3.2)$$

Dva centrované nečíslované vzorce:

$$\begin{aligned} \mathbf{Y} &= \mathbb{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \\ \mathbb{X} &= \begin{pmatrix} 1 & \mathbf{x}_1^\top \\ \vdots & \vdots \\ 1 & \mathbf{x}_n^\top \end{pmatrix}. \end{aligned}$$

Dva centrované číslované vzorce:

$$\mathbf{Y} = \mathbb{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \quad (3.3)$$

$$\mathbb{X} = \begin{pmatrix} 1 & \mathbf{x}_1^\top \\ \vdots & \vdots \\ 1 & \mathbf{x}_n^\top \end{pmatrix}. \quad (3.4)$$

Definice rozdělená na dva případy:

$$P_{r-j} = \begin{cases} 0, & \text{je-li } r-j \text{ liché,} \\ r! (-1)^{(r-j)/2}, & \text{je-li } r-j \text{ sudé.} \end{cases}$$

Všimněte si použití interpunkce v této konstrukci. Čárky a tečky se dávají na místa, kam podle jazykových pravidel patří.

$$\begin{aligned} x &= y_1 - y_2 + y_3 - y_5 + y_8 - \cdots = & \text{z (3.3)} \\ &= y' \circ y^* = & \text{podle (3.4)} \\ &= y(0)y' & \text{z Axiomu 1.} \end{aligned} \quad (3.5)$$

Dva zarovnané vzorce nečíslované:

$$\begin{aligned} L(\boldsymbol{\theta}) &= \prod_{i=1}^n f_i(y_i; \boldsymbol{\theta}), \\ \ell(\boldsymbol{\theta}) &= \log\{L(\boldsymbol{\theta})\} = \sum_{i=1}^n \log\{f_i(y_i; \boldsymbol{\theta})\}. \end{aligned}$$

Dva zarovnané vzorce, první číslovaný:

$$\begin{aligned} L(\boldsymbol{\theta}) &= \prod_{i=1}^n f_i(y_i; \boldsymbol{\theta}), & (3.6) \\ \ell(\boldsymbol{\theta}) &= \log\{L(\boldsymbol{\theta})\} = \sum_{i=1}^n \log\{f_i(y_i; \boldsymbol{\theta})\}. \end{aligned}$$

Vzorec na dva řádky, první řádek zarovnaný vlevo, druhý vpravo, nečíslovaný:

$$\begin{aligned} \ell(\mu, \sigma^2) &= \log\{L(\mu, \sigma^2)\} = \sum_{i=1}^n \log\{f_i(y_i; \mu, \sigma^2)\} = \\ &= -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - \mu)^2. \end{aligned}$$

Vzorec na dva řádky, zarovnaný na =, číslovaný uprostřed:

$$\begin{aligned} \ell(\mu, \sigma^2) &= \log\{L(\mu, \sigma^2)\} = \sum_{i=1}^n \log\{f(y_i; \mu, \sigma^2)\} = \\ &= -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - \mu)^2. \end{aligned} \quad (3.7)$$

### 3.4 Definice, věty, důkazy, . . .

Konstrukce typu definice, věta, důkaz, příklad, . . . je vhodné odlišit od okolního textu a případně též číslovat s možností použití křížových odkazů. Pro každý typ těchto konstrukcí je vhodné mít v souboru s makry (`makra.tex`) nadefinované jedno prostředí, které zajistí jak vizuální odlišení od okolního textu, tak automatické číslování s možností křížově odkazovat.

**Definícia 6.** *Nechť náhodné veličiny  $X_1, \dots, X_n$  jsou definovány na témž pravděpodobnostním prostoru  $(\Omega, \mathcal{A}, P)$ . Pak vektor  $\mathbf{X} = (X_1, \dots, X_n)^\top$  nazveme náhodným vektorem.*

**Definícia 7** (náhodný vektor). *Nechť náhodné veličiny  $X_1, \dots, X_n$  jsou definovány na témž pravděpodobnostním prostoru  $(\Omega, \mathcal{A}, P)$ . Pak vektor  $\mathbf{X} = (X_1, \dots, X_n)^\top$  nazveme náhodným vektorem.* ■

Definice 6 ukazuje použití prostředí pro sazbu definice bez titulku, definice 7 ukazuje použití prostředí pro sazbu definice s titulkem.

**Věta 5.** *Náhodný vektor  $\mathbf{X}$  je měřitelné zobrazení prostoru  $(\Omega, \mathcal{A}, P)$  do  $(\mathbb{R}_n, \mathcal{B}_n)$ .*

**Lemma 6** (Anděl, 2007, str. 29). *Náhodný vektor  $\mathbf{X}$  je měřitelné zobrazení prostoru  $(\Omega, \mathcal{A}, P)$  do  $(\mathbb{R}_n, \mathcal{B}_n)$ .*

*Dôkaz.* Jednotlivé kroky důkazu jsou podrobně popsány v práci Anděl (2007, str. 29). □

Věta 5 ukazuje použití prostředí pro sazbu matematické věty bez titulku, lemma 6 ukazuje použití prostředí pro sazbu matematické věty s titulkem. Lemmata byla zavedena v hlavním souboru tak, že sdílejí číslování s větami.

## 4. Odkazy na literaturu

Odkazy na literaturu vytváříme nejlépe pomocí příkazů `\citet`, `\citep` atp. (viz L<sup>A</sup>T<sub>E</sub>Xový balíček `natbib`) a následného použití BibT<sub>E</sub>Xu. V matematickém textu obvykle odkazujeme stylem „Jméno autora/autorů (rok vydání)“, resp. „Jméno autora/autorů [číslo odkazu]“. V českém/slovenském textu je potřeba se navíc vypořádat s nutností skloňovat jméno autora, respektive přechylovat jméno autorky. Je potřeba mít na paměti, že standardní příkazy `\citet`, `\citep` produkují referenci se jménem autora/autorů v prvním pádě a jména autorek jsou nepřechýlena.

Pokud nepoužíváme bibT<sub>E</sub>X, řídíme se normou ISO 690 a zvyklostmi oboru. Jména časopisů lze uvádět zkráceně, ale pouze v kodifikované podobě.

### 4.1 Několik ukázek

Mezi nejvíce citované statistické články patří práce Kaplana a Meiera a Coxe (Kaplan a Meier, 1958; Cox, 1972). Student (1908) napsal článek o t-testu.

Prof. Anděl je autorem učebnice matematické statistiky (viz Anděl, 1998). Teorii odhadu se věnuje práce Lehmann a Casella (1998). V případě odkazů na specifickou informaci (definice, důkaz, ...) uvedenou v knize bývá užitečné uvést specificky číslo kapitoly, číslo věty atp. obsahující požadovanou informaci, např. viz Anděl (2007, Věta 4.22) nebo (viz Anděl, 2007, Věta 4.22).

Mnoho článků je výsledkem spolupráce celé řady osob. Při odkazování v textu na článek se třemi autory obvykle při prvním výskytu uvedeme plný seznam: Dempster, Laird a Rubin (1977) představili koncept EM algoritmu. Respektive: Koncept EM algoritmu byl představen v práci Dempstera, Lairdové a Rubina (Dempster, Laird a Rubin, 1977). Při každém dalším výskytu již používáme zkrácenou verzi: Dempster a kol. (1977) nabízejí též několik příkladů použití EM algoritmu. Respektive: Několik příkladů použití EM algoritmu lze nalézt též v práci Dempstera a kol. (Dempster a kol., 1977).

U článku s více než třemi autory odkazujeme vždy zkrácenou formou: První výsledky projektu ACCEPT jsou uvedeny v práci Genbergové a kol. (Genberg a kol., 2008). V textu *nenapíšeme*: První výsledky projektu ACCEPT jsou uvedeny v práci Genberg, Kulich, Kawichai, Modiba, Chingono, Kilonzo, Richter, Pettifor, Sweat a Celentano (2008).

## 5. Tabulky, obrázky, programy

Používání tabulek a grafů v odborném textu má některá společná pravidla a některá specifická. Tabulky a grafy neuvádíme přímo do textu, ale umístíme je buď na samostatné stránky nebo na vyhrazené místo v horní nebo dolní části běžných stránek. L<sup>A</sup>T<sub>E</sub>X se o umístění plovoucích grafů a tabulek postará automaticky.

Každý graf a tabulku očíslovujeme a umístíme pod ně legendu. Legenda má popisovat obsah grafu či tabulky tak podrobně, aby jim čtenář rozuměl bez důkladného studování textu práce.

Na každou tabulku a graf musí být v textu odkaz pomocí jejich čísla. Na příslušném místě textu pak shrneme ty nejdůležitější závěry, které lze z tabulky či grafu učinit. Text by měl být čitelný a srozumitelný i bez prohlížení tabulek a grafů a tabulky a grafy by měly být srozumitelné i bez podrobné četby textu.

Na tabulky a grafy odkazujeme pokud možno nepřímo v průběhu běžného toku textu; místo „*Tabulka 5.1 ukazuje, že muži jsou v průměru o 9,9 kg těžší než ženy*“ raději napíšeme „*Muži jsou o 9,9 kg těžší než ženy (viz Tabulka 5.1)*“.

### 5.1 Tabulky

U **tabulek** se doporučuje dodržovat následující pravidla:

- Vyhýbat se svislým linkám. Silnějšími vodorovnými linkami oddělit tabulku od okolního textu včetně legendy, slabšími vodorovnými linkami oddělovat záhlaví sloupců od těla tabulky a jednotlivé části tabulky mezi sebou. V L<sup>A</sup>T<sub>E</sub>Xu tuto podobu tabulek implementuje balík `booktabs`. Chceme-li výrazněji oddělit některé sloupce od jiných, vložíme mezi ně větší mezeru.
- Neměnit typ, formát a význam obsahu políček v tomtéž sloupci (není dobré do téhož sloupce zapisovat tu průměr, onde procenta).
- Neopakovat tentýž obsah políček mnohokrát za sebou. Máme-li sloupec *Rozptyl*, který v prvních deseti řádcích obsahuje hodnotu 0,5 a v druhých deseti řádcích hodnotu 1,5, pak tento sloupec raději zrušíme a vyřešíme to jinak. Například můžeme tabulku rozdělit na dvě nebo do ní vložit popisné řádky, které informují o nějaké proměnné hodnotě opakující se v následujícím oddíle tabulky (např. „*Rozptyl = 0,5*“ a níže „*Rozptyl = 1,5*“).

Efekt	Odhad	Směrod. chyba <sup>a</sup>	P-hodnota
Abs. člen	−10,01	1,01	—
Pohlaví (muž)	9,89	5,98	0,098
Výška (cm)	0,78	0,12	< 0,001

Pozn: <sup>a</sup> Směrodatná chyba odhadu metodou Monte Carlo.

Tabulka 5.1: Maximálně věrohodné odhady v modelu M.

- Čísla v tabulce zarovnávat na desetinnou čárku.
- V tabulce je někdy potřebné používat zkratky, které se jinde nevyskytují. Tyto zkratky můžeme vysvětlit v legendě nebo v poznámkách pod tabulkou. Poznámky pod tabulkou můžeme využít i k podrobnějšímu vysvětlení významu některých sloupců nebo hodnot.

## 5.2 Obrázky

Několik rad týkajících se obrázků a grafů.

- Graf by měl být vytvořen ve velikosti, v níž bude použit v práci. Zmenšení příliš velkého grafu vede ke špatné čitelnosti popisků.
- Osy grafu musí být řádně popsány ve stejném jazyce, v jakém je psána práce (absenci diakritiky lze tolerovat). Kreslíme-li graf hmotnosti proti výšce, nenecháme na nich popisky **ht** a **wt**, ale osy popíšeme *Výška [cm]* a *Hmotnost [kg]*. Kreslíme-li graf funkce  $h(x)$ , popíšeme osy  $x$  a  $h(x)$ . Každá osa musí mít jasně určenou škálu.
- Chceme-li na dvourozměrném grafu vyznačit velké množství bodů, dáme pozor, aby se neslily do jednolitě černé tmy. Je-li bodů mnoho, zmenšíme velikost symbolu, kterým je vykresluje, anebo vybereme jen malou část bodů, kterou do grafu zaneseme. Grafy, které obsahují tisíce bodů, dělají problémy hlavně v elektronických dokumentech, protože výrazně zvětšují velikost souborů.
- Budeme-li práci tisknout černobíle, vyhneme se používání barev. Čáry rozlišujeme typem (plná, tečkovaná, čerchovaná, ...), plochy dostatečně rozdílnými intenzitami šedé nebo šrafováním. Význam jednotlivých typů čar a ploch vysvětlíme buď v textové legendě ke grafu anebo v grafické legendě, která je přímo součástí obrázku.
- Vyhýbejte se bitmapovým obrázkům o nízkém rozlišení a zejména JPEGům (zuby a kompresní artefakty nevypadají na papíře pěkně). Lepší je vytvářet obrázky vektorově a vložit do textu jako PDF.

## 5.3 Programy

Algoritmy, výpisy programů a popis interakce s programy je vhodné odlišit od ostatního textu. Jednou z možností je použití L<sup>A</sup>T<sub>E</sub>Xového balíčku **fancyvrb** (fancy verbatim), pomocí něhož je v souboru **makra.tex** nadefinováno prostředí **code**. Pomocí něho lze vytvořit např. následující ukázky.

```
> mean(x)
[1] 158.90
> objekt$prumer
[1] 158.90
```

Menší písmo:

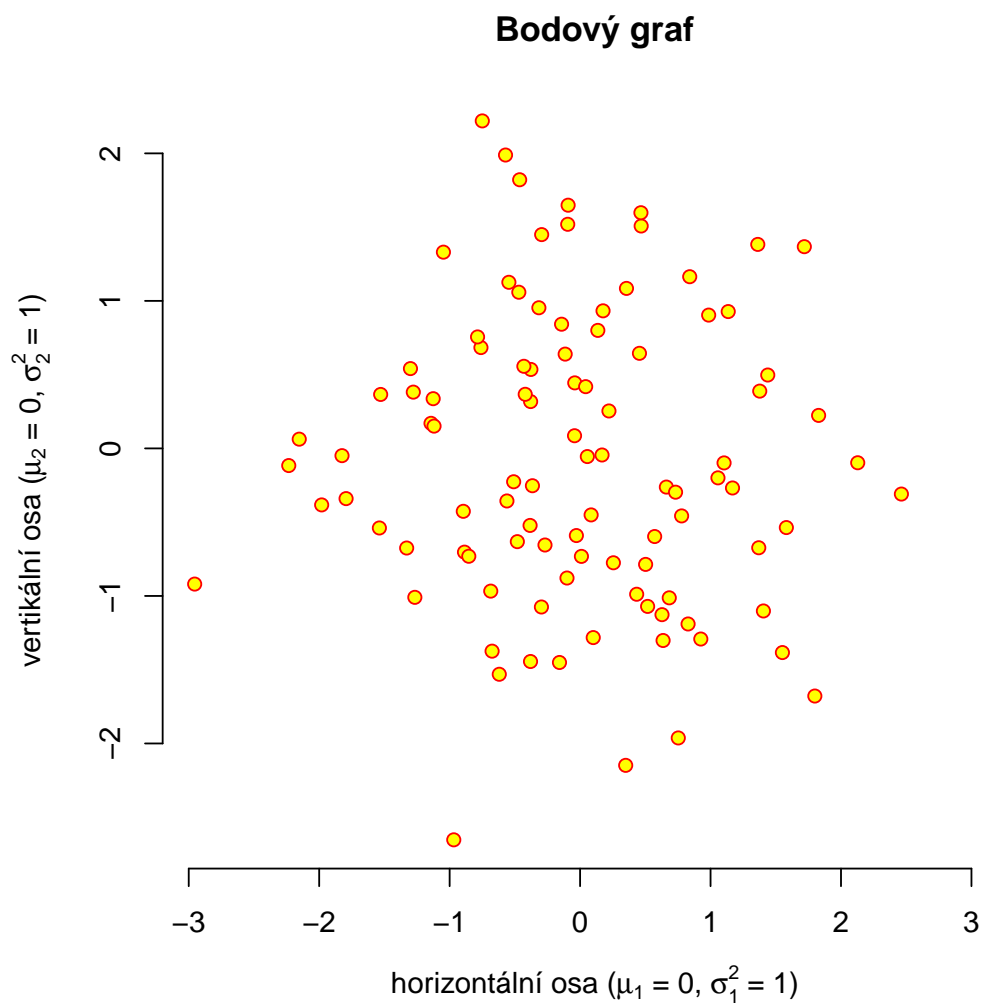
```
> mean(x)
[1] 158.90
> objekt$prumer
[1] 158.90
```

Bez rámečku:

```
> mean(x)
[1] 158.90
> objekt$prumer
[1] 158.90
```

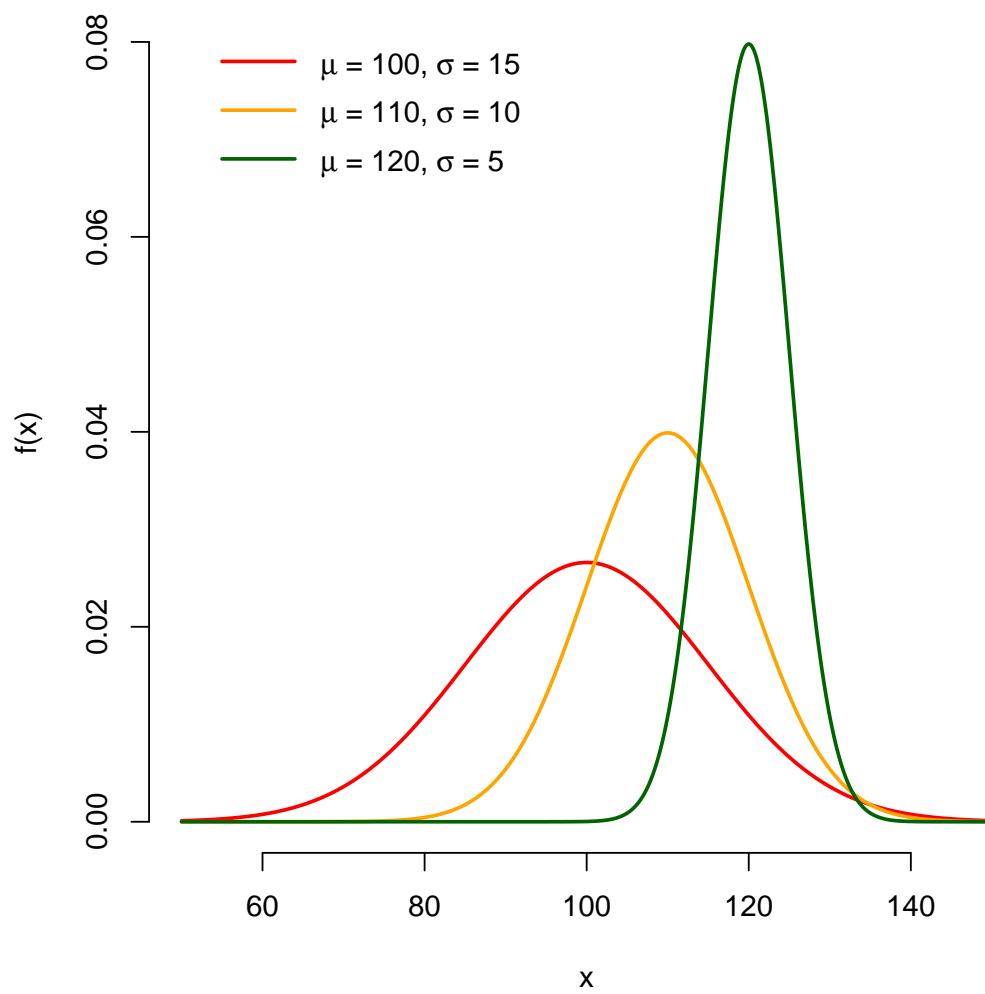
Užší rámeček:

```
> mean(x)
[1] 158.90
> objekt$prumer
[1] 158.90
```

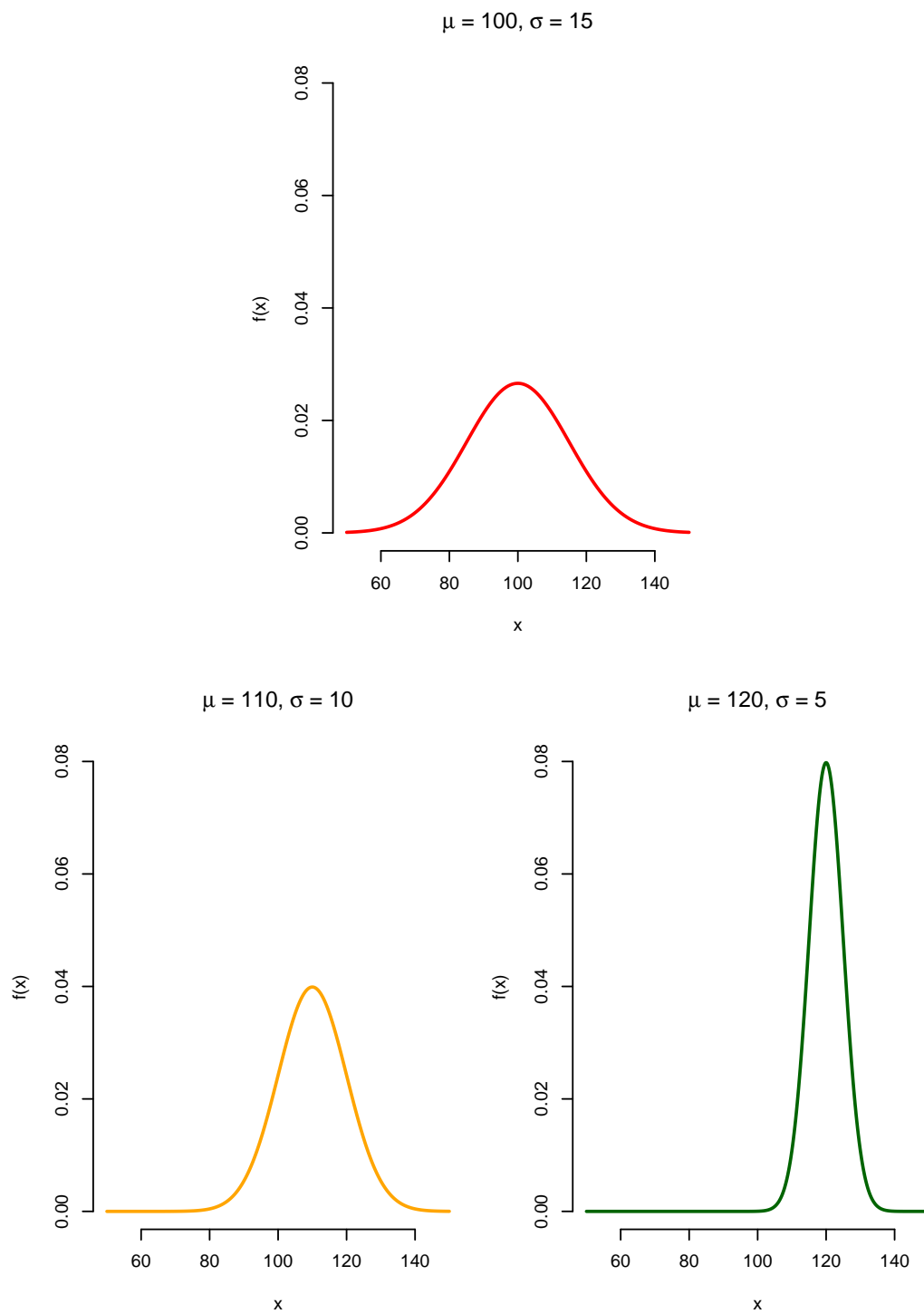


Obr. 5.1: Náhodný výběr z rozdělení  $\mathcal{N}_2(\mathbf{0}, I)$ .





Obr. 5.2: Hustoty několika normálních rozdělení.



Obr. 5.3: Hustoty několika normálních rozdělení.

# Závěr

# Seznam použité literatury

- ANDĚL, J. (1998). *Statistické metody*. Druhé přepracované vydání. Matfyzpress, Praha. ISBN 80-85863-27-8.
- ANDĚL, J. (2007). *Základy matematické statistiky*. Druhé opravené vydání. Matfyzpress, Praha. ISBN 80-7378-001-1.
- COX, D. R. (1972). Regression models and life-tables (with Discussion). *Journal of the Royal Statistical Society, Series B*, **34**(2), 187–220.
- DEMAINE, E. D., MOZES, S., ROSSMAN, B. a WEIMANN, O. (2009). An optimal decomposition algorithm for tree edit distance. *ACM Trans. Algorithms*, **6**(1), 2:1–2:19. ISSN 1549-6325. doi: 10.1145/1644015.1644017. URL <http://doi.acm.org/10.1145/1644015.1644017>.
- DEMPSTER, A. P., LAIRD, N. M. a RUBIN, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, **39**(1), 1–38.
- DULUCQ, S. a TOUZET, H. (2003). *Combinatorial Pattern Matching: 14th Annual Symposium, CPM 2003 Morelia, Michoacán, Mexico, June 25–27, 2003 Proceedings*, chapter Analysis of Tree Edit Distance Algorithms, pages 83–95. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-44888-4. doi: 10.1007/3-540-44888-8\_7. URL [http://dx.doi.org/10.1007/3-540-44888-8\\_7](http://dx.doi.org/10.1007/3-540-44888-8_7).
- GENBERG, B. L., KULICH, M., KAWICHAJ, S., MODIBA, P., CHINGONO, A., KILONZO, G. P., RICHTER, L., PETTIFOR, A., SWEAT, M. a CELENTANO, D. D. (2008). HIV risk behaviors in sub-Saharan Africa and Northern Thailand: Baseline behavioral data from project Accept. *Journal of Acquired Immune Deficiency Syndrome*, **49**, 309–319.
- KAPLAN, E. L. a MEIER, P. (1958). Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, **53**(282), 457–481.
- KLEIN, P. N. (1998). Computing the edit-distance between unrooted ordered trees. In *Proceedings of the 6th Annual European Symposium on Algorithms, ESA '98*, pages 91–102, London, UK, UK, 1998. Springer-Verlag. ISBN 3-540-64848-8. URL <http://dl.acm.org/citation.cfm?id=647908.740125>.
- LEHMANN, E. L. a CASELLA, G. (1998). *Theory of Point Estimation*. Second Edition. Springer-Verlag, New York. ISBN 0-387-98502-6.
- PAWLIK, M. a AUGSTEN, N. (2011). Rted: A robust algorithm for the tree edit distance. *Proc. VLDB Endow.*, **5**(4), 334–345. ISSN 2150-8097. doi: 10.14778/2095686.2095692. URL <http://dx.doi.org/10.14778/2095686.2095692>.
- STUDENT (1908). On the probable error of the mean. *Biometrika*, **6**, 1–25.

- TAI, K.-C. (1979). The tree-to-tree correction problem. *J. ACM*, **26**(3), 422–433. ISSN 0004-5411. doi: 10.1145/322139.322143. URL <http://doi.acm.org/10.1145/322139.322143>.
- ZHANG, K. a SHASHA, D. (1989). Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, **18**(6), 1245 – 1262.

# Zoznam obrázkov

1.1	Ukazky TED operaci . . . . .	2
1.2	Rekurzivny vzorec pre vypocet tree-edit-distance . . . . .	3
1.3	Celkova dekompozicia pomocou LRH strategii . . . . .	7
2.1	Stem a loop v molekule . . . . .	11
5.1	Náhodný výběr z rozdělení $\mathcal{N}_2(\mathbf{0}, I)$ . . . . .	20
5.2	Hustoty několika normálních rozdělení. . . . .	21
5.3	Hustoty několika normálních rozdělení. . . . .	22

# Zoznam tabuliek

5.1	Maximálne věrohodné odhady v modelu M. . . . .	17
-----	--	----

# Seznam použitých zkratek



# Přílohy