Plán semestra

Minule: *tri algoritmy na zlepšenie pamäťovej/časovej zložitosti dyn.prog.*

Dnes: *približné výskyty vzorky, lokálne podobnosti, bioinformatika*

Zajtra: *zostavovanie DNA sekvencií, najkratšie spoločné nadslovo*

Streda 1.5.: sviatok

Štvrtok 2.5.: *viacnásobné zarovnanie, opakujúce sa sekvenčné motívy*

Streda 8.5.: sviatok

Štvrtok 9.5.: prezentácie

Streda 15.5.: prezentácie

Štvrtok 16.5.: prezentácie

Edit distance, Levenshtein distance (editačná vzdialenosť)

Edit operations: $(u, v \in \Sigma^*, a, b \in \Sigma)$

– insertion (inzercia) $uv \rightarrow uav$

– deletion (delécia) $uav \rightarrow uv$

– substitution (substitúcia) $uav \rightarrow ubv$

Edit distance $d_E(S, T) =$

shortest sequence of edit operations that changes S to T

Example:

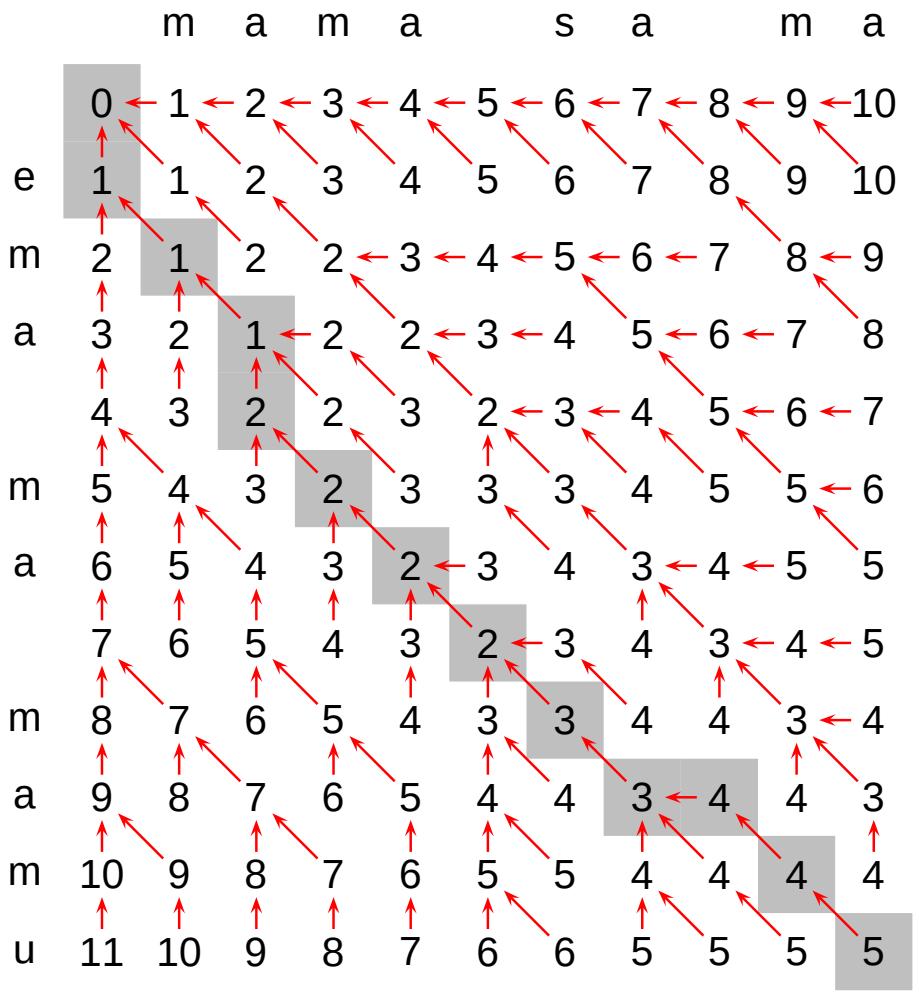
$S =$ema ma mamu, $T =$mama sa ma, $d_E(S, T) = 5$

ema␣ma␣mamu (delete e) ma␣ma␣mamu (delete space)

mama␣mamu (substitute m->s) mama␣samu (insert space)

mama␣sa␣mu (substitute u-a) mama␣sa␣ma

Dynamic programming for computing $d_E(S,T)$



Subproblem:

$$A[i,j] =$$

$$d_E(S[1..i], T[1..j])$$

Recurrence:

$$A[i,j] =$$

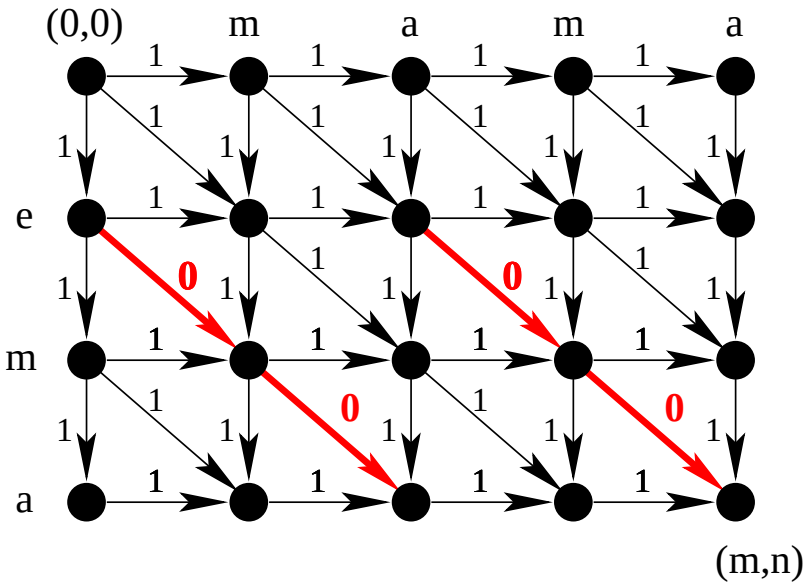
$$\min \begin{cases} A[i-1,j-1] + c(S[i], T[j]) \\ A[i-1,j] + 1 \\ A[i,j-1] + 1 \end{cases}$$

Alignment:

```
ema␣ma␣ma-mu
-ma-ma␣sa␣ma
```

Dynamic programming in graph representation

$$A[i,j] = \min \begin{cases} A[i-1,j-1] + c(S[i], T[j]) \\ A[i-1,j] + 1 \\ A[i,j-1] + 1 \end{cases}$$



Each alignment – path from $(0,0)$ to (m,n)

Alignment with lowest cost – path with lowest cost (shortest path)

Approximate string matching (približné výskyty vzorky)

Find substrings of T s.t. $d(T[i..j], P) \leq k$

For Hamming distance $O(nk)$ (suffix trees and LCA)

Edit distance:

Today: Simple dynamic programming $O(nm)$

– How to create a graph in which short paths correspond to occurrences?

Today: two simple algorithms that work well on average

Not covered: dynamic programming + suffix trees + LCA $O(nk)$

Approximate string matching (približné výskyty vzorky)

Simple dynamic programming $O(mn)$

$$A[0,j] = 0$$

$$A[i,0] = i$$

$$A[i,j] = \min \begin{cases} A[i-1,j-1] + c(S[i],T[j]) \\ A[i-1,j] + 1 \\ A[i,j-1] + 1 \end{cases}$$

Print all j s.t. $A[m,j] \leq k$ (ends of occurrences)

We need only 2 columns from A

What if we want the best start (with smallest d_E) for each end?

Approximate string matching (približné výskyty vzorky)

Simple improvement, good on average:

Compute correct values only for active cells

We call $A[i, j]$ active if $A[i, j] \leq k$

Let $L[j]$ be largest i s.t. $A[i, j]$ is active

Lemma: $L[j] \leq L[j-1] + 1$

Recall: adjacent cells in a row/column differ by at most 1

Approximate string matching (približné výskyty vzorky)

```
1   Compute A[*,0];
2   L = k;
3   for(j=1; j<=n; j++) {
4     L2 = 0;
5     for(i=0; i<=L+1; i++) {
6       compute A[i,j];   // if unknown A[i,j-1], assume k+1
7       if(A[i,j]<=k) L2 = i;
8     }
9     L = L2;
10  }
```

Average-case running time $O(kn)$, worst-case $O(mn)$

Baeza-Yates, Perleberg 1992

Divide P to $k+1$ substrings of size $r = \lfloor \frac{m}{k+1} \rfloor$ (last one possibly longer). Denote the set of these substrings $\mathcal{P}$.

Lemma: Let T' be a substring of T. If $d_E(T', P) \leq k$, at least one of the strings in $\mathcal{P}$ is a substring of T'.

Let $\mathcal{I}$ be the set of occurrences of strings in $\mathcal{P}$ in T.

Baeza-Yates, Perleberg 1992

Average-case running time $O(n + n(k+1)\sigma^{-r}m^2)$

Claim: If m is a multiple of $k+1$ and $k \leq \frac{m}{3\log_\sigma m} - 1$, then average-case running time is $O(n)$.

Proof:

$$r = \left\lfloor \frac{m}{k+1} \right\rfloor = \frac{m}{k+1} \geq \frac{m}{m/(3\log_\sigma m)} = 3\log_\sigma m$$

$$\sigma^{-r} \leq \sigma^{-3\log_\sigma m} = m^{-3}$$

$$n(k+1)\sigma^{-r}m^2 \leq n$$

For example:

$m = 10, \sigma = 4$ needs $k \leq 1$

$m = 100, \sigma = 4$ needs $k \leq 9$

$m = 1000, \sigma = 4$ needs $k \leq 61$ (considering also $\lfloor\rfloor$)

Variants of edit distance

Weighted edit distance:

different cost of different operations depending on symbols

$w(a, b)$: cost of aligning a to b ($a, b \in \Sigma \cup \{-\}$)

Sequence similarity:

Use positive $w(a, b)$ for equal characters

negative for substitutions and gaps

sim(S, T) = highest score of an alignment of S and T

$$A[i, j] = \max \begin{cases} A[i-1, j-1] + w(S[i], T[j]) \\ A[i-1, j] + w(S[i], -) \\ A[i, j-1] + w(-, T[j]) \end{cases}$$

Local alignment

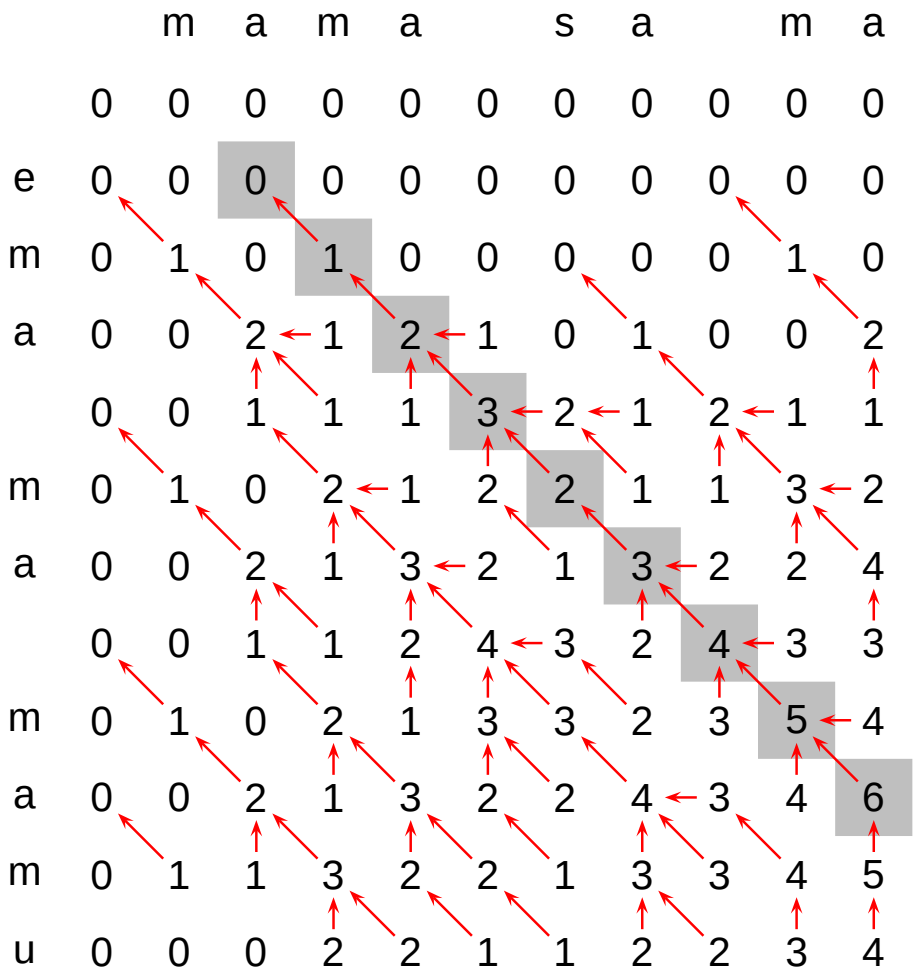
Find substrings of S and T with highest similarity

$\max_{i,j,k,l} \text{sim}(S[i..j], T[k..l])$

Useful if only parts of strings similar (biology, plagiarism)

How to modify graph to compute this?

Local alignment



$$w(a,b) = \begin{cases} 1 & \text{if } a = b,\ a \in \Sigma \\ -1 & \text{otherwise} \end{cases}$$

`ma␣ma␣ma`

`ma␣sa␣ma`

Introduction to bioinformatics

Deoxyribonucleic acid (DNA)

Contains genetic information passed to next generations.

Long string of nucleotides over $\{A, C, G, T\}$

(adenine, cytosine, guanine, thymine).

Information stored in symbolic, digital form.

Ribonucleic acid (RNA)

Similar to DNA, thymine T replaced with uracil U

Proteins (proteíny, bielkoviny)

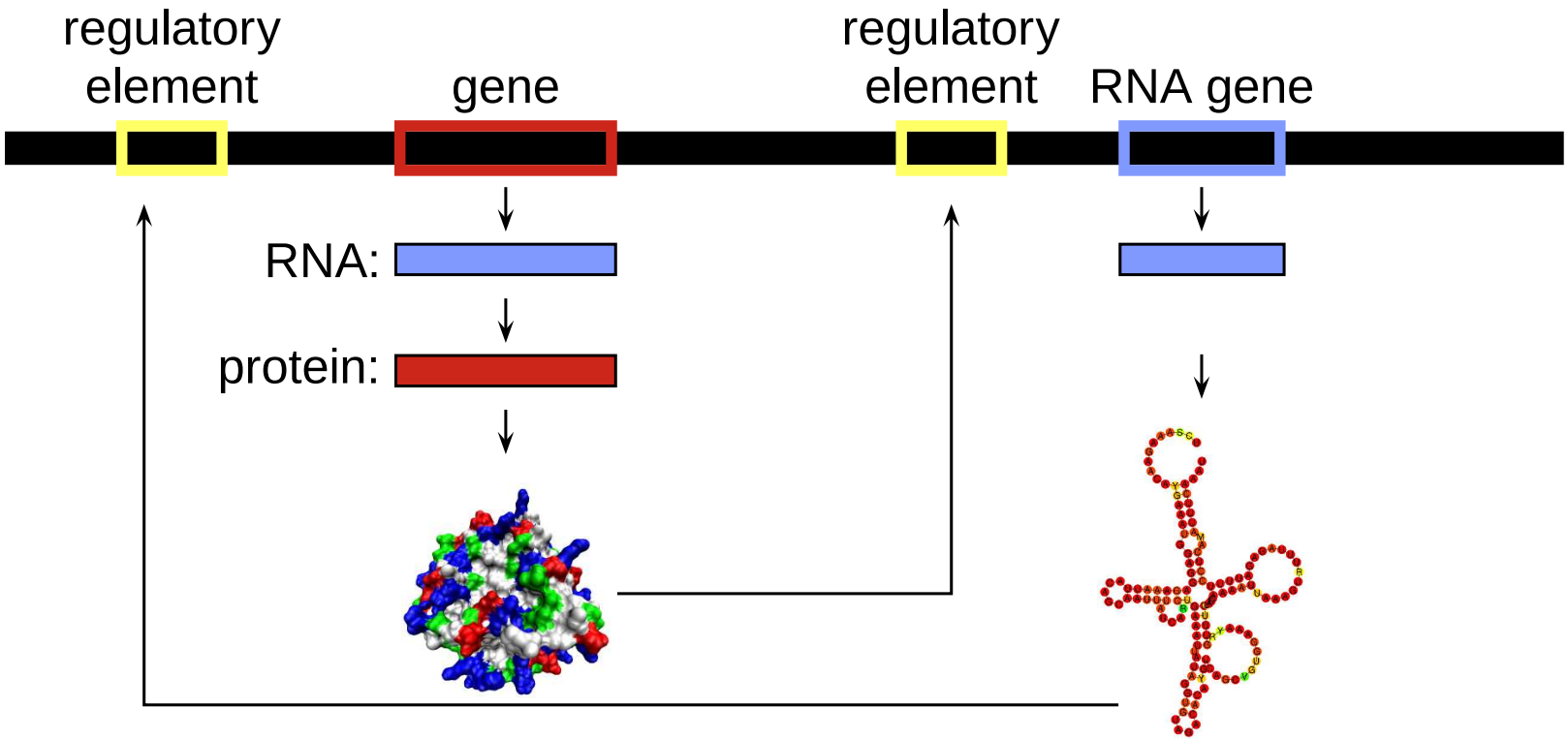
Catalyze biochemical reaction in a cell (enzymes)

transfer signals within or between cells,

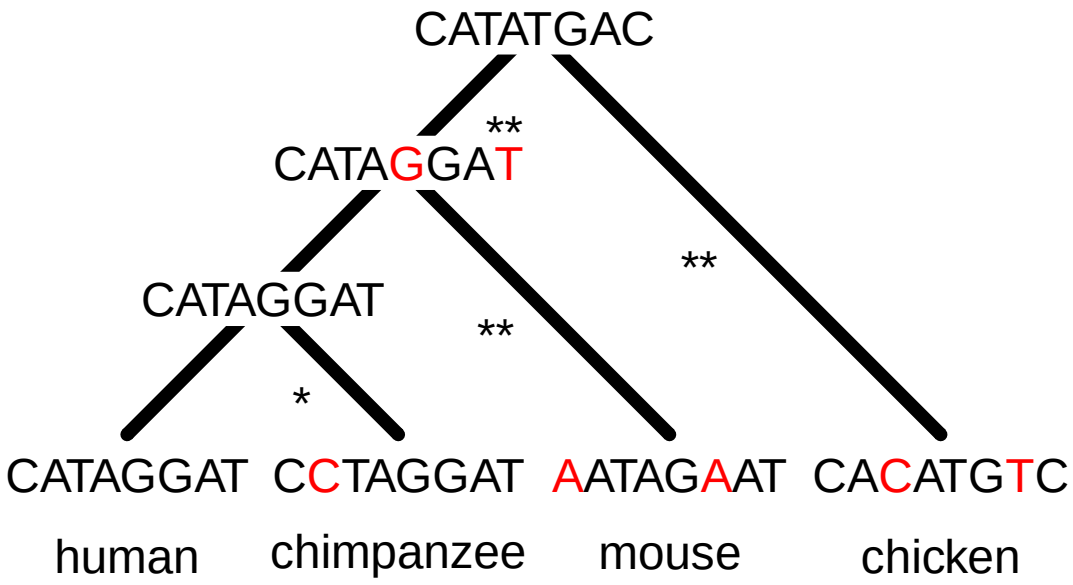
important for cell structure and movement.

String of aminoacids (alphabet of size 20).

What is encoded in DNA?



Evolution



Mutations: substitutions, deletions, insertions, rearrangements

Local alignment: one of basic tools in molecular biology

What is the function of a newly sequenced DNA/protein?

Search in a database if any known sequences similar.

Similar sequences often share similar function.

Very popular program: BLAST (Basic local alignment search tool)

Altschul, Gish, Miller, Myers, Lipman 1990

Runs online vs. Genbank database

http://blast.ncbi.nlm.nih.gov/Blast.cgi

Local alignment: bioinformatics tricks

- Full dynamic programming too slow, use **heuristic filtering**:
  – find exact matches of length w
  – extend them along diagonal
  – connect nearby diagonals, improve by dyn.prog.
- **Scoring matrices** based on probabilistic models of evolution
- **Affine gap cost:** insertion/deletion of length k costs $a + bk$
- **Statistical significance** of found alignments
  How many matches with equal or higher score are expected by chance?

Other examples of bioinformatics problems

- Phylogenetic tree reconstruction (NP hard problems)
- RNA structure prediction (dynamic programming)
- Gene finding (probabilistic models, dynamic programming)
- Protein structure prediction (very hard problems, how to score?)

Useful techniques: probability and statistics, dynamic programming, machine learning, methods for solving hard problems

More in a special course:

Metódy v bioinformatike 2-AIN-501 (ZS, 4 hodiny, 6 kreditov)