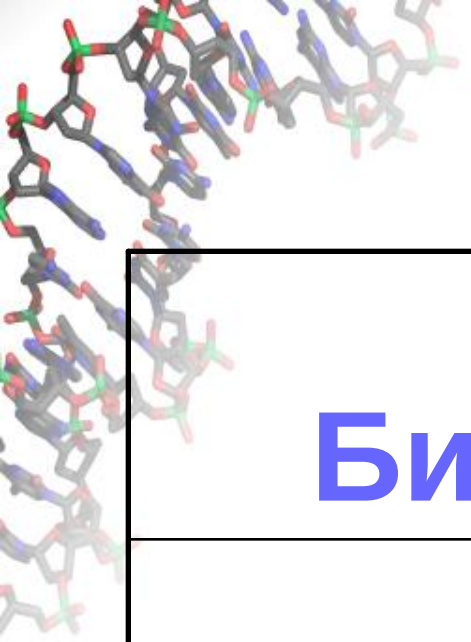
# Лингвистика

| | |
|---|---|
| Био | Физика |
| Эконо | Информатика |
| Социо | Лингвистика |

**Байкал**

***23 августа 2011***

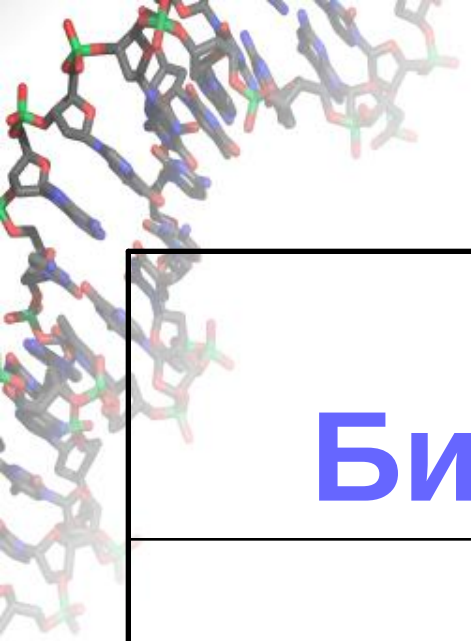


# Лингвистика

| | |
|---|---|
| **Био** | Физика |
| Эконо | **Информатика** |
| Социо | Лингвистика |

**Байкал**

***23 августа 2011***

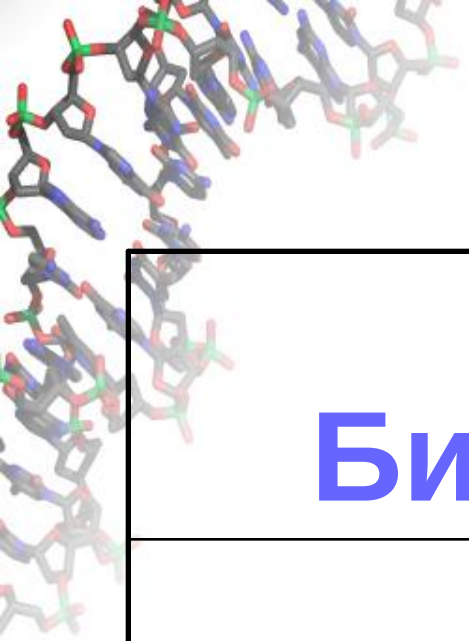


# Лингвистика

| | |
|---|---|
| **Био** | ***Физика*** |
| Эконо | **Информатика** |
| Социо | Лингвистика |

**Байкал**

***23 августа 2011***

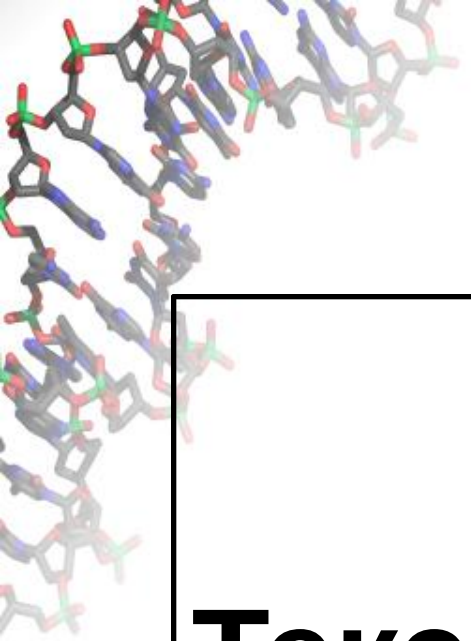


# *Лингвистика*

| | |
|---|---|
| **Био** | ***Физика*** |
| Эконо | **Информатика** |
| Социо | ***Лингвистика*** |

**Байкал**

***23 августа 2011***

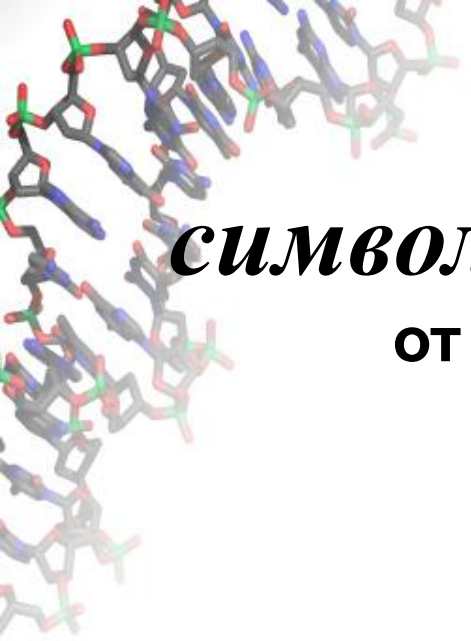


# *Анализ данных*

| | |
|---|---|
| **Тексты, графы** | Биология |
| | Информатика |
| | Лингвистика |

**Байкал**

***23 августа 2011***



# *Анализ символьных последовательностей*

**от биоинформатики до лингвистики**

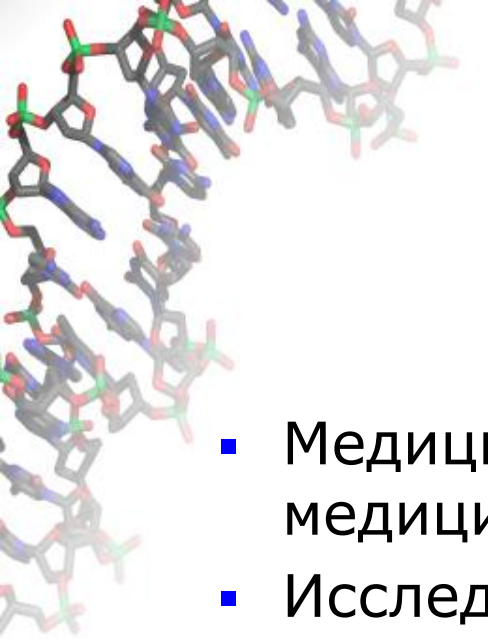
**М.А. Ройтберг**

**ЦЕЛИ**

- Знакомство с биоинформатикой (анализ данных в биоинформатике)
- Математические этюды

**Байкал**

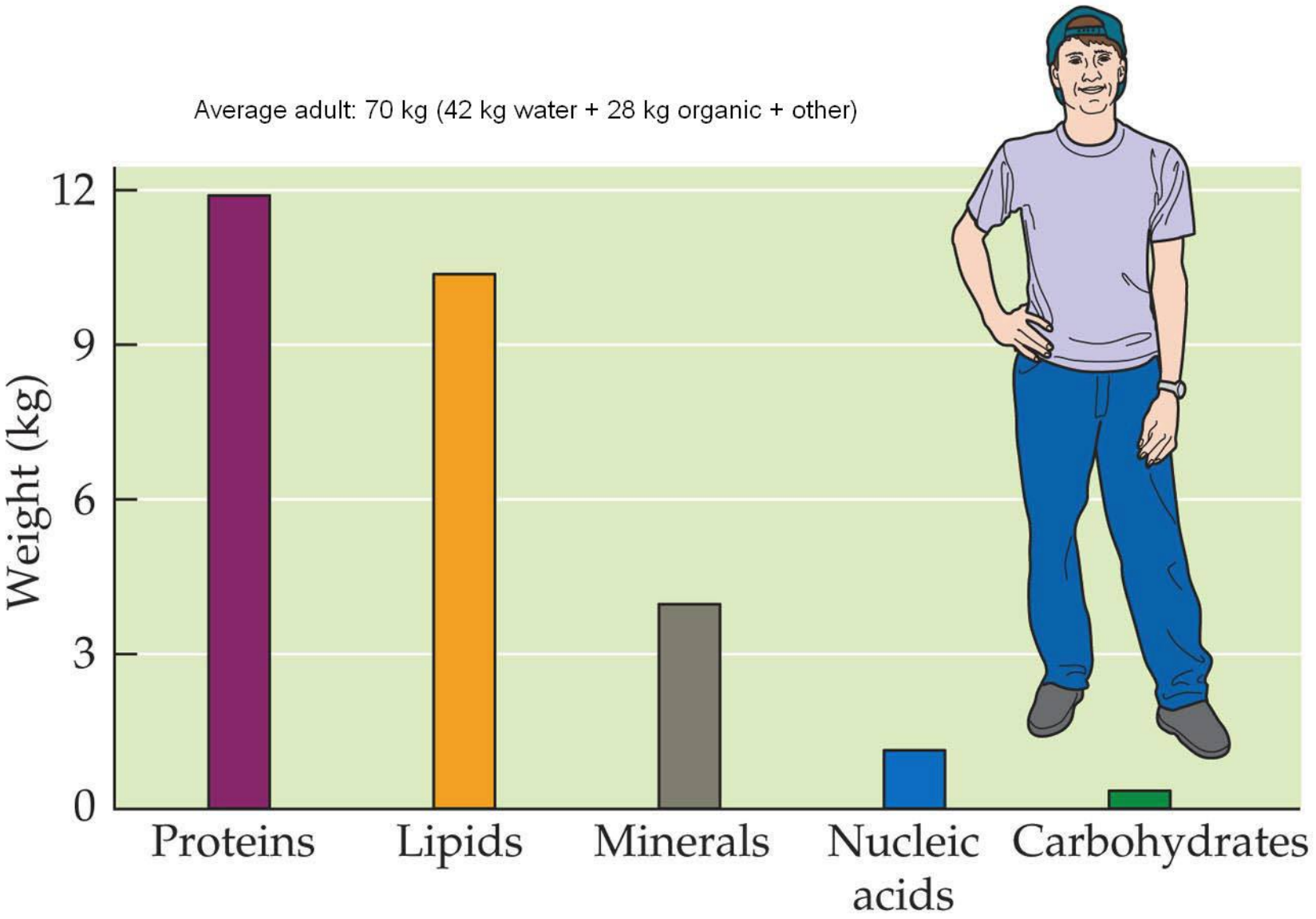
***23 августа 2011***



# Проблематика (молекулярная биология)

- Медицинские приложения (разработка лекарств, медицинская генетика, персональная медицина)
- Исследования механизмов функционирования клетки (и надклеточных структур): молекулярная биология, биофизики, биохимия…
- Теория эволюции, систематика, филогения

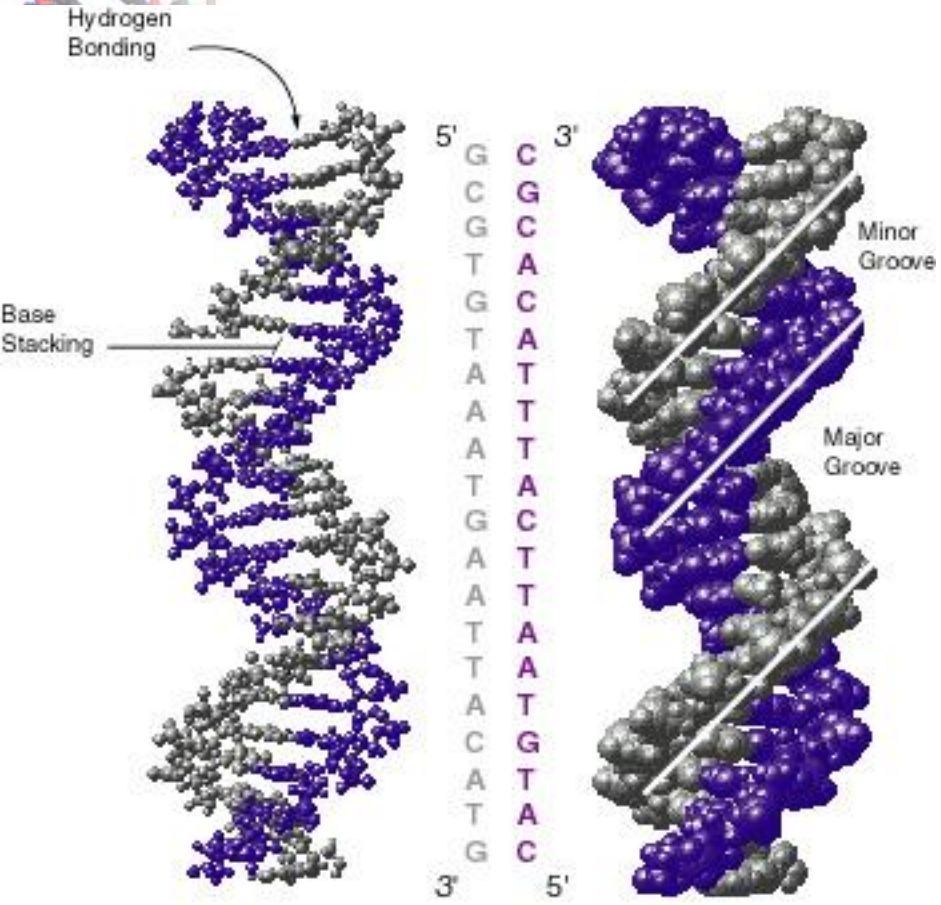
Average adult: 70 kg (42 kg water + 28 kg organic + other)

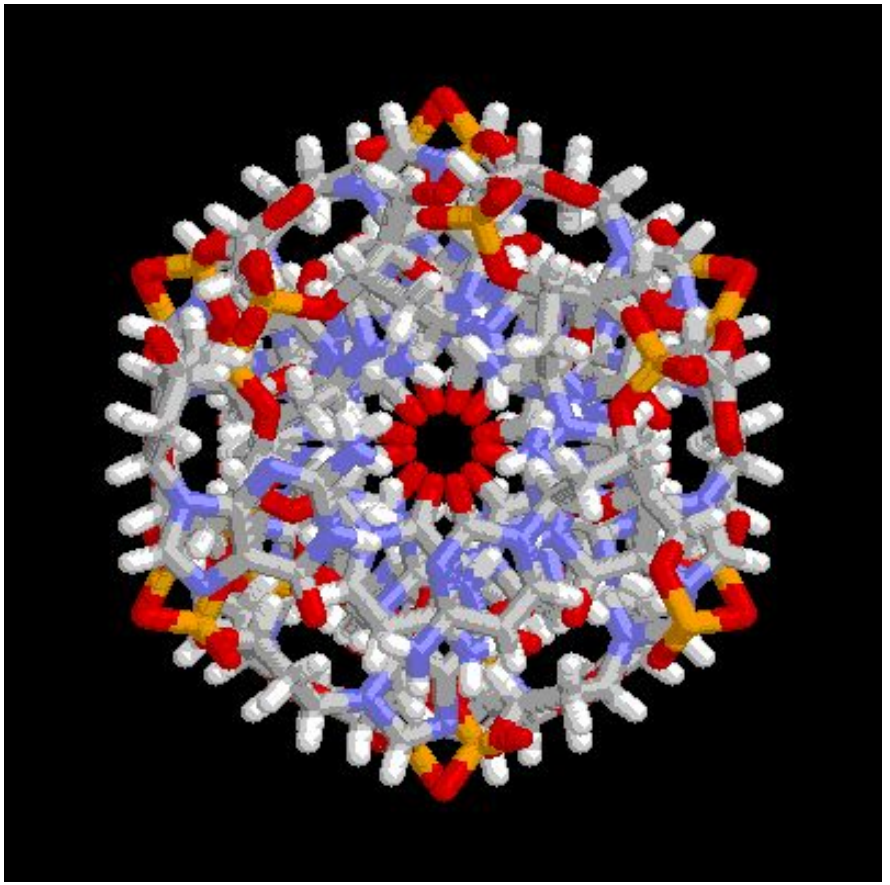



*ANIMAL PHYSIOLOGY*, **Figure 4.1** 

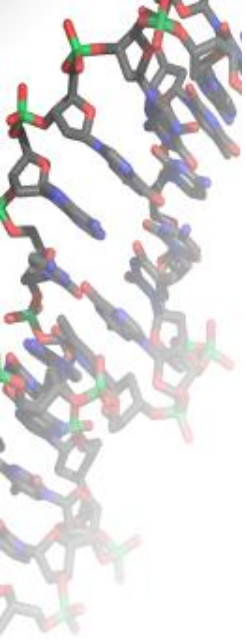
# ДНК: 2 нити; L ~ $10^5$ – $10^9$

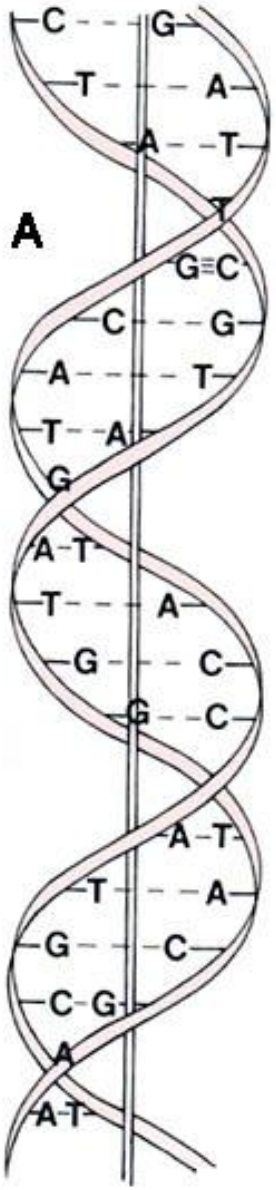
***нуклеотиды (4)***



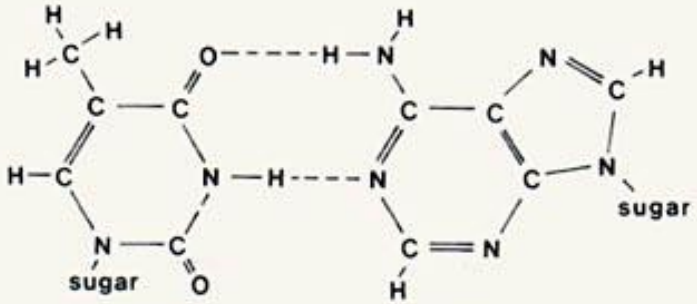



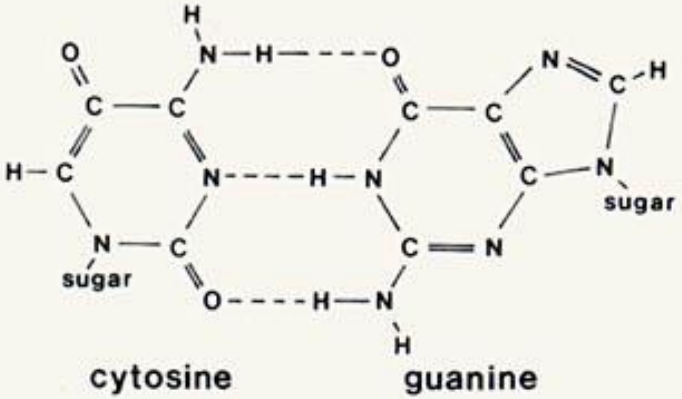
Miesfeld / Applied Molecular Genetics
Fig. 01.03. DNA Structure.






B



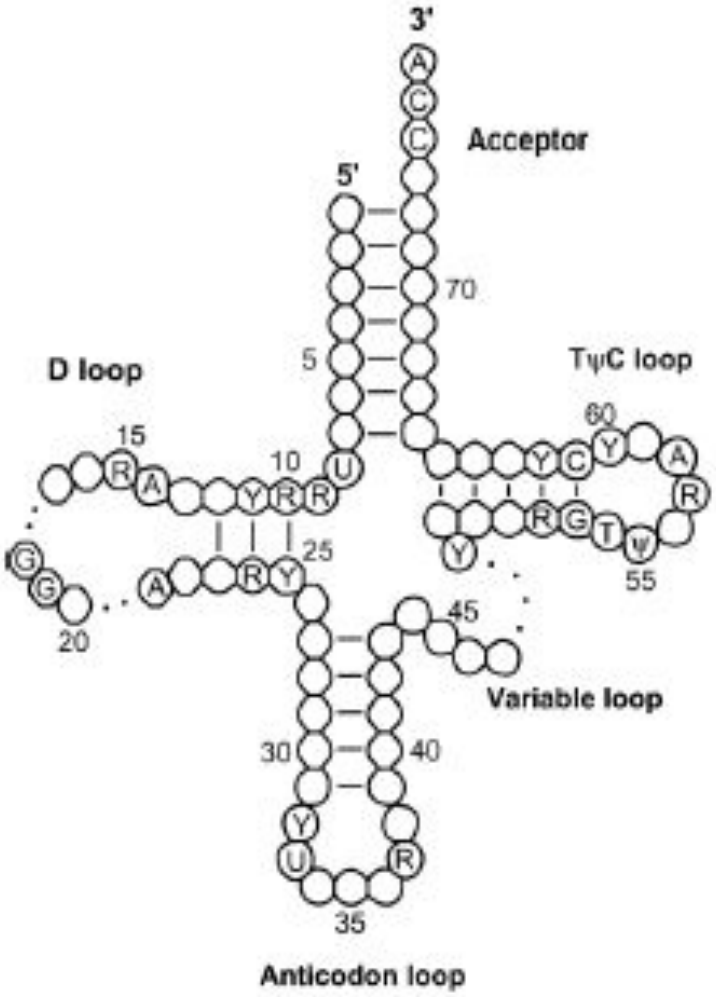



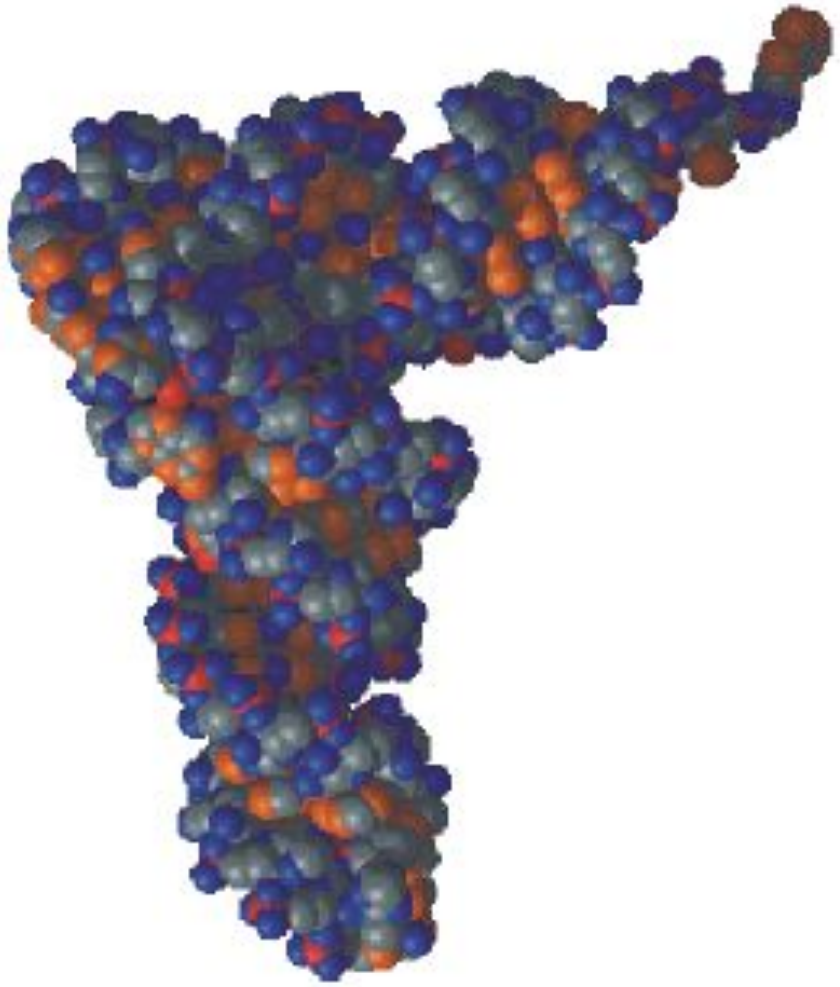

C

T---A
C---G
G---C
A---T
A---T
T---A
C---G

# РНК: 1 нить; L ~ $10^2$ – $10^3$

***нуклеотиды (4)***

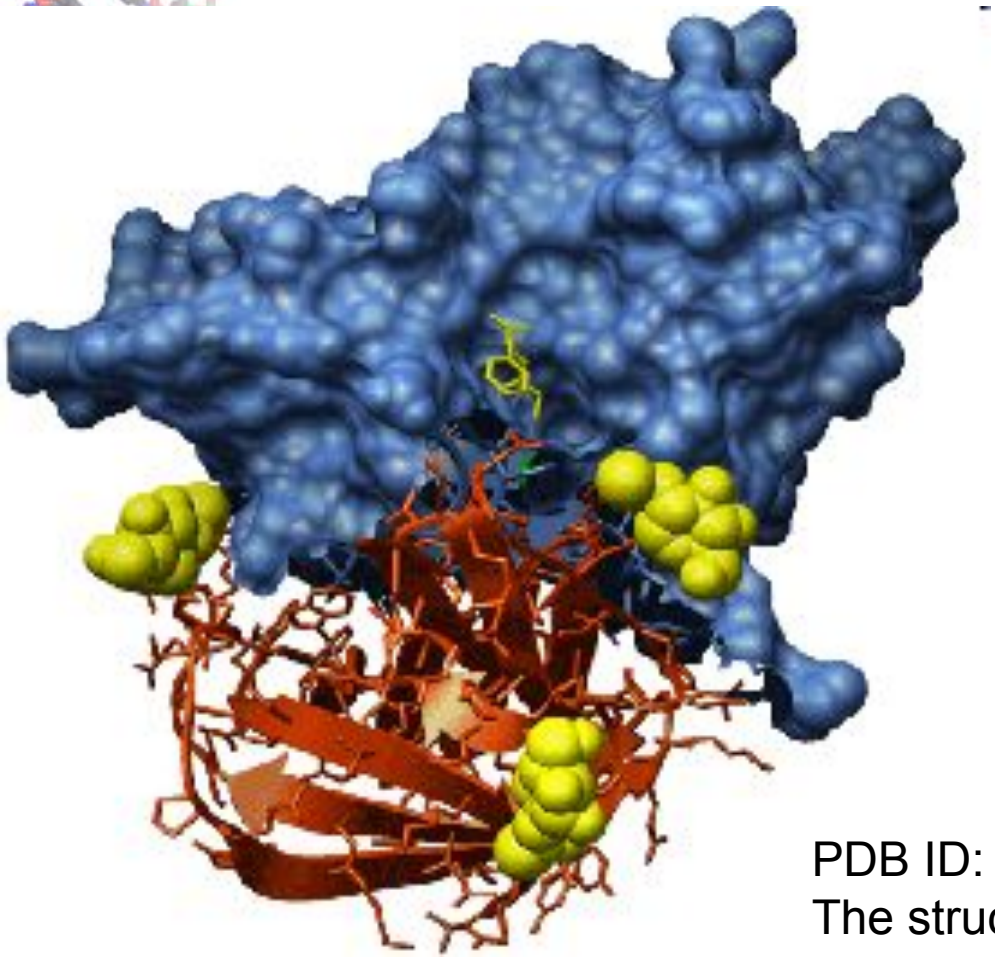


From Paul Higgs

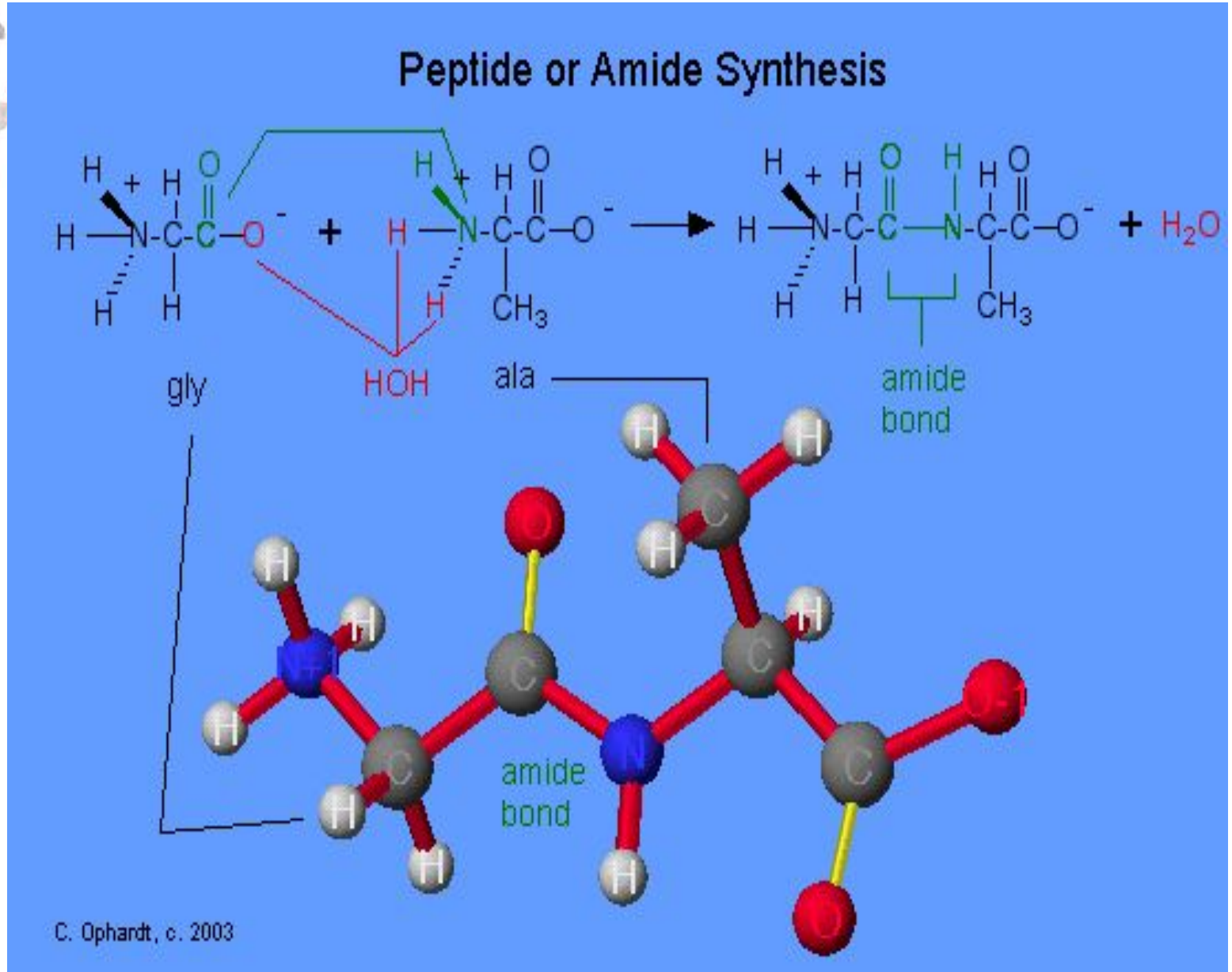
# Белки: 1 нить; L ~ $10^2$ – $10^3$

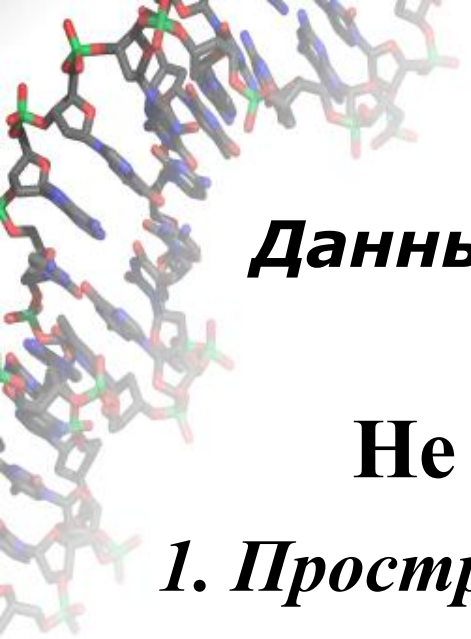
***аминокислоты (20)***



PDB ID: **2act** E.N. Baker, E.J. Dodson (1980): The structure of actinidin at 1.7 Ångstroms

# …Gly + Ala…  = …GA…


Peptide or Amide Synthesis
gly
HOH
ala
amide bond
amide bond
C. Ophardt, c. 2003



## *Данные:* последовательности

### Не только последовательности

***1. Пространственные структуры***

- сравнение, анализ (пример: «докинг»)

***2. Генные сети***

***3. «Секвенирование»***

***4. «Экспрессия генов»***

## Основные задачи анализа последовательностей

### *1. Сравнение*

- сопоставление в целом (в т.ч. - множественное); определение количественной меры сходства последовательностей в целом;

-поиск общих мотивов; поиск в базах данных;

### *2. Аннотация (описание)*

поиск и выделение функционально значимых участков (заданных «паттернов»);

- разбиение последовательности на «однородные» участки;

- определение статистической значимости результатов сравнения и поиска.

### *3. Структуры*

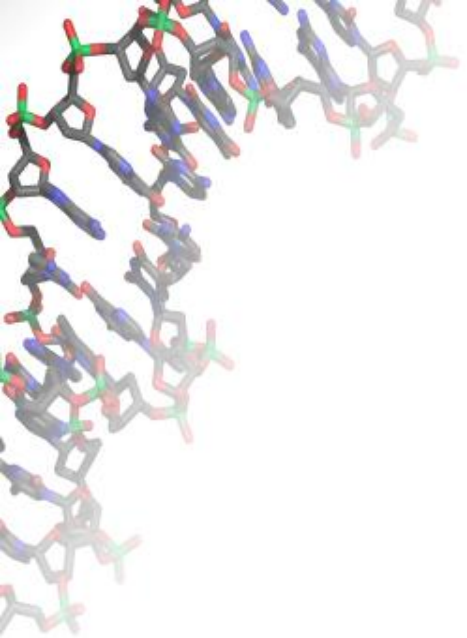
- предсказание; сравнение (обогащенные последовательности)

# *ИСТОРИЯ и ДЛИНЫ*

- tRNA - (**1964**) - **75** bases (old, slow, complicated method)
- First complete DNA genome: X174 DNA (**1977**) - **5386** bases
- human mitochondrial DNA (**1981**) - **16,569** bases
- tobacco chloroplast DNA (**1986**) - **155,844** bases
- First complete bacterial genome (*H. Influenzae*)(**1995**) - **1.9 x 10^6** bases
- Yeast genome (eukaryote at ~ **1.5 x 10^7**) completed in **1996**
- Several archaebacteria
- *E. coli* -- **4 x 10^6** bases [**1998**]
- Several pathogenic bacterial genomes sequenced
  - Helicobacter pyloris, Treponema pallidium, Borrelia burgdorferi, Chlamydia trachomatis, Rickettsia prowazekii, Mycobacterium tuberculosis
- Nematode C. elegans ( ~ **4 x 10^8**) - December **1998**
- Human genome (rough draft completed **2000**) - **3 x 10^9** base
- **2010** – rat, mouse, pig, fugu, etc, full genomes **50 x 10^9**
- ***~2015*** – individual human genomes ("$1000 per genome")

# План доклада

- Выравнивания.
- Динамическое программирование, графы и алгебра
- Поиск локальных сходств, затравки
- Структуры РНК
- Гиперграфы и контекстно-свободные грамматики
- Конечные автоматы и вероятности
- Разные примеры



# Тема 1. Выравнивание

# Варианты выравниваний

***Выровнять*** две символьные последовательности – удалить из них несколько фрагментов так, чтобы оставшиеся последовательности имели одинаковую длину.

--ПОДБЕРЕЗОВИК
ПРЕДОСИНОВИЧКИ

ПОДБЕРЕЗОВИК--
ПРЕДОСИНОВИЧКИ

ПО-ДБЕРЕЗОВИК--
ПРЕДОСИН-ОВИЧКИ

П-ОДБЕРЕЗОВИК--
ПРЕД-ОСИНОВИЧКИ

ПО-ДБЕРЕЗОВИ-К-
ПРЕД-ОСИНОВИЧКИ

# Какой вариант выбрать?

А) Б)

--ПОДБЕРЕЗОВИК ПОДБЕРЕЗОВИК--
ПРЕДОСИНОВИЧКИ ПРЕДОСИНОВИЧКИ

В) Г) Д)

ПО-ДБЕРЕЗОВИК-- П-ОДБЕРЕЗОВИК-- ПО-ДБЕРЕЗОВИ-К-
ПРЕДОСИН-ОВИЧКИ ПРЕД-ОСИНОВИЧКИ ПРЕД-ОСИНОВИЧКИ

***Предполагается: последовательности были получены редактированием» («эволюцией») из общего предка.***

***Требуется: установить соответствующие друг другу участки***

## Какой вариант выбрать?

# Нужно «знать» что-нибудь про эволюцию

А) --ПОДБЕРЕЗОВИК ПРЕДОСИНОВИЧКИ

Б) ПОДБЕРЕЗОВИК-- ПРЕДОСИНОВИЧКИ

В) ПО-ДБЕРЕЗОВИК-- ПРЕДОСИН-ОВИЧКИ

Г) П-ОДБЕРЕЗОВИК-- ПРЕД-ОСИНОВИЧКИ

Д) ПО-ДБЕРЕЗОВИ-К- ПРЕД-ОСИНОВИЧКИ

***Предположим:***

***Две одинаковые буквы скорее имеют общего предка, чем две разные буквы***

***Две буквы «одинаковой гласности» скорее имеют общего предка, чем две буквы «разные гласности»***

***Две одинаковые буквы скорее имеют общего предка, чем две разные буквы***

***Две буквы «одинаковой гласности» скорее имеют общего предка, чем две буквы «разные гласности»***

А) --ПОДБЕРЕЗОВИК ПРЕДОСИНОВИЧКИ

Б) ПОДБЕРЕЗОВИК-- ПРЕДОСИНОВИЧКИ

В) ПО-ДБЕРЕЗОВИК-- ПРЕДОСИН-ОВИЧКИ

Г) П-ОДБЕРЕЗОВИК-- ПРЕД-ОСИНОВИЧКИ

Д) ПО-ДБЕРЕЗОВИ-К- ПРЕД-ОСИНОВИЧКИ

**Г) лучше, чем В); Б) [немного] лучше А)**

***??? Верно ли, что***

**Г ) лучше, чем Б )**

***Две одинаковые буквы скорее имеют общего предка, чем две разные буквы***

***Две буквы «одинаковой гласности» скорее имеют общего предка, чем две буквы «разные гласности»***

А) --ПОДБЕРЕЗОВИК ПРЕДОСИНОВИЧКИ

Б) ПОДБЕРЕЗОВИК-- ПРЕДОСИНОВИЧКИ

В) ПО-ДБЕРЕЗОВИК-- ПРЕДОСИН-ОВИЧКИ

Г) П-ОДБЕРЕЗОВИК-- ПРЕД-ОСИНОВИЧКИ

Д) ПО-ДБЕРЕЗОВИ-К- ПРЕД-ОСИНОВИЧКИ

*??? Верно ли, что*

**Г ) лучше, чем Б )**

*=== НЕИЗВЕСТНО. Мы ничего не предположили о механизме удалений/вставок (насколько они вероятны по сравнению с заменами)*

# Вес выравнивания

```
A T - V V I - - T G S
G S M V L L E F S G T
0+2  +3+2+3    +2+7+2= 21
   -1       -2       = -3
```

**Score = $\Sigma$ m(i,j)-GapPen = 21 - 3 = 18**

PAM250 matrix recommended by Gonnet et al. Science, June 5, 1992

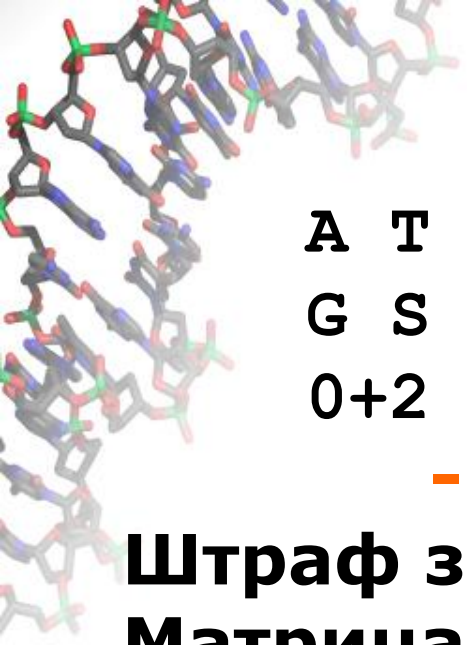
Values rounded to nearest integer

| | C | S | T | P | A | G | N | D | E | Q | H | R | K | M | I | L | V | F | Y | W |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C | 12 | 0 | 0 | -3 | 0 | -2 | -2 | -3 | -3 | -2 | -1 | -2 | -3 | -1 | -1 | -2 | 0 | -1 | 0 | -1 |
| S | 0 | 2 | 2 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | -2 | -2 | -1 | -3 | -2 | -3 |
| T | 0 | 2 | 2 | 0 | 1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | -1 | -1 | 0 | -2 | -2 | -4 |
| P | -3 | 0 | 0 | 8 | 0 | -2 | -1 | -1 | 0 | 0 | -1 | -1 | -1 | -2 | -3 | -2 | -2 | -4 | -3 | -5 |
| A | 0 | 1 | 1 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | -1 | -1 | 0 | -1 | -1 | -1 | 0 | -2 | -2 | -4 |
| G | -2 | 0 | -1 | -2 | 0 | 7 | 0 | 0 | -1 | -1 | -1 | -1 | -1 | -4 | -4 | -4 | -3 | -5 | -4 | -4 |
| N | -2 | 1 | 0 | -1 | 0 | 0 | 4 | 2 | 1 | 1 | 1 | 0 | 1 | -2 | -3 | -3 | -2 | -3 | -1 | -4 |
| D | -3 | 0 | 0 | -1 | 0 | 0 | 2 | 5 | 3 | 1 | 0 | 0 | 0 | -3 | -4 | -4 | -3 | -4 | -3 | -5 |
| E | -3 | 0 | 0 | 0 | 0 | -1 | 1 | 3 | 4 | 2 | 0 | 0 | 1 | -2 | -3 | -3 | -2 | -4 | -3 | -4 |
| Q | -2 | 0 | 0 | 0 | 0 | -1 | 1 | 1 | 2 | 3 | 1 | 2 | 2 | -1 | -2 | -2 | -2 | -3 | -2 | -3 |
| H | -1 | 0 | 0 | -1 | -1 | -1 | 1 | 0 | 0 | 1 | 6 | 1 | 1 | -1 | -2 | -2 | -2 | 0 | 2 | -1 |
| R | -2 | 0 | 0 | -1 | -1 | -1 | 0 | 0 | 0 | 2 | 1 | 5 | 3 | -2 | -2 | -2 | -2 | -3 | -2 | -2 |
| K | -3 | 0 | 0 | -1 | 0 | -1 | 1 | 0 | 1 | 2 | 1 | 3 | 3 | -1 | -2 | -2 | -2 | -3 | -2 | -4 |
| M | -1 | -1 | -1 | -2 | -1 | -4 | -2 | -3 | -2 | -1 | -1 | -2 | -1 | 4 | 2 | 3 | 2 | 2 | 0 | -1 |
| I | -1 | -2 | -1 | -3 | -1 | -4 | -3 | -4 | -3 | -2 | -2 | -2 | -2 | 2 | 4 | 3 | 3 | 1 | -1 | -2 |
| L | -2 | -2 | -1 | -2 | -1 | -4 | -3 | -4 | -3 | -2 | -2 | -2 | -2 | 3 | 3 | 4 | 2 | 2 | 0 | -1 |
| V | 0 | -1 | 0 | -2 | 0 | -3 | -2 | -3 | -2 | -2 | -2 | -2 | -2 | 2 | 3 | 2 | 3 | 0 | -1 | -3 |
| F | -1 | -3 | -2 | -4 | -2 | -5 | -3 | -4 | -4 | -3 | 0 | -3 | -3 | 2 | 1 | 2 | 0 | 7 | 5 | 4 |
| Y | 0 | -2 | -2 | -3 | -2 | -4 | -1 | -3 | -3 | -2 | 2 | -2 | -2 | 0 | -1 | 0 | -1 | 5 | 8 | 4 |
| W | -1 | -3 | -4 | -5 | -4 | -4 | -4 | -5 | -4 | -3 | -1 | -2 | -4 | -1 | -2 | -1 | -3 | 4 | 4 | 14 |

**Матрица весов замен *m(a, b)***

**Штраф за удаление символа δ = -1**

***GapPen – сумма штрафов за удаления***



# *Вес выравнивания*

```
A T - V V I — - T G S
G S M V L L E F S G T
0+2  +3+2+3    +2+7+2= 21
```

**-1 -2 = -3**

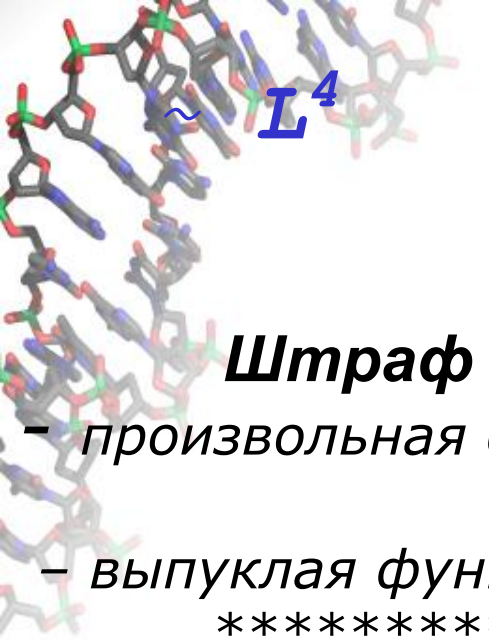
**Штраф за удаление символа: δ =-1**

**Матрица весов замен: m(a,b)**

***Score = Σ m(i,j)-GapPen =*** **21 - 3 =** ***18***

***GapPen – сумма штрафов за удаления.***

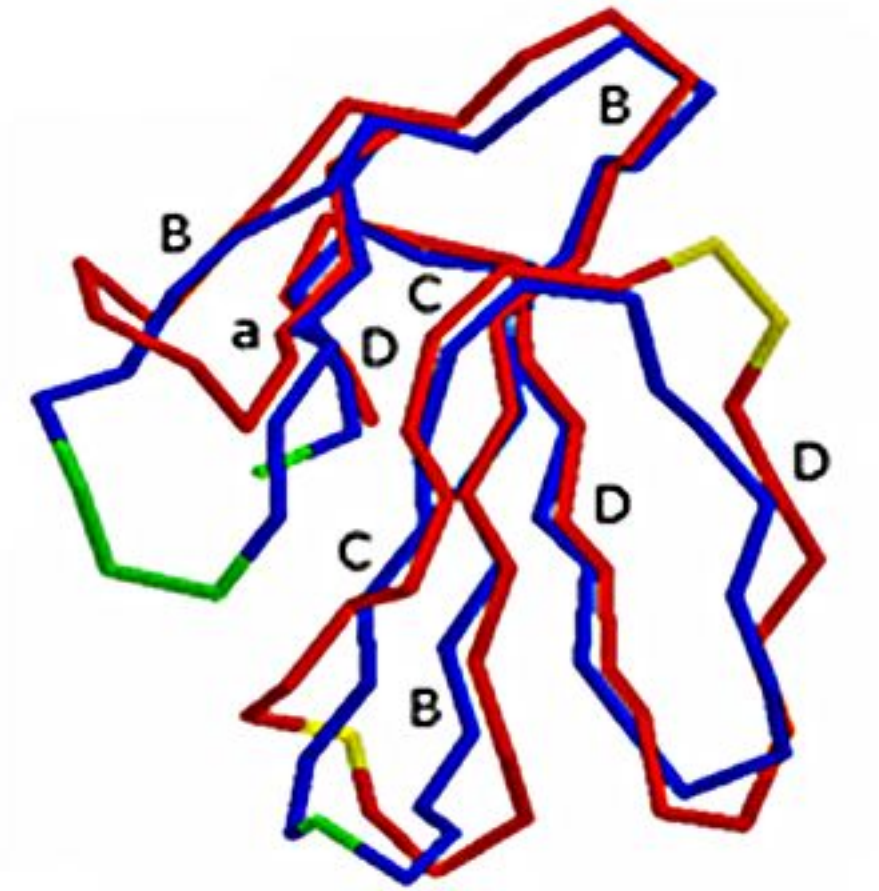
# *Score -> MAXIMUM*

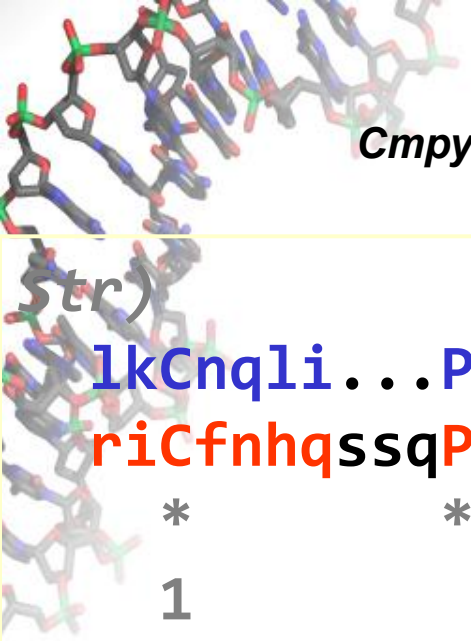


~ $L^4$

| ***Штраф за делецию f(L)*** | ***Время работы*** |
|---|---|
| - *произвольная функция* | ~ $L^4$ |
| – *выпуклая функция* | ~ $L^3$ |
| ********** | |
| – ***линейная f(L) = a + bL (Смит-Уотерман)*** | ~ $L^2$ |
| – *линейная f(L) = kL* | ~ $L^2$ |
| - *нулевая f(L) = 0* | ~<$L^2$ (зависит от п-стей) |

# Эталонные выравнивания





*Str)*

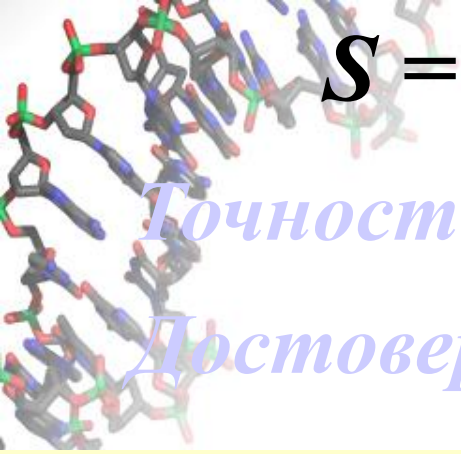
**40 сопоставлений**

```
lkCnqli...PPFWKTCPKGKNLCYKmtmraapmvPVKRGCidv
riCfnhqssqPQTTKTCSPGESSCYHkqwsdfrgtIIERGCg..
  *       ****************          ******
  1             16                    6
```

AlgSW)

```
    1             16                    6
    *       ****************          ******
lk..C...nqliPPFWKTCPKGKNLCYK...mtmraapmvPVKRGCidv
..riCfnhqssqPQTTKTCSPGESSCYHkqwsdfrgt...IIERGC..g
```

**35 сопоставлений**



$S = 40$ $I = 23$ $A = 35$

Точность $Acc = I/S = 23/40 = 0.58$

Достоверность $Conf = I/A = 23/35 = 0.66$

Str)

lkCnqli...PPFWKTCPKGKNLCYKmtmraapmvPVKRGCidv

riCfnhqssqPQTTKTCSPGESSCYHkqwsdfrgtIIERGCg..

* ****************** ******

1 16 6

AlgSW)

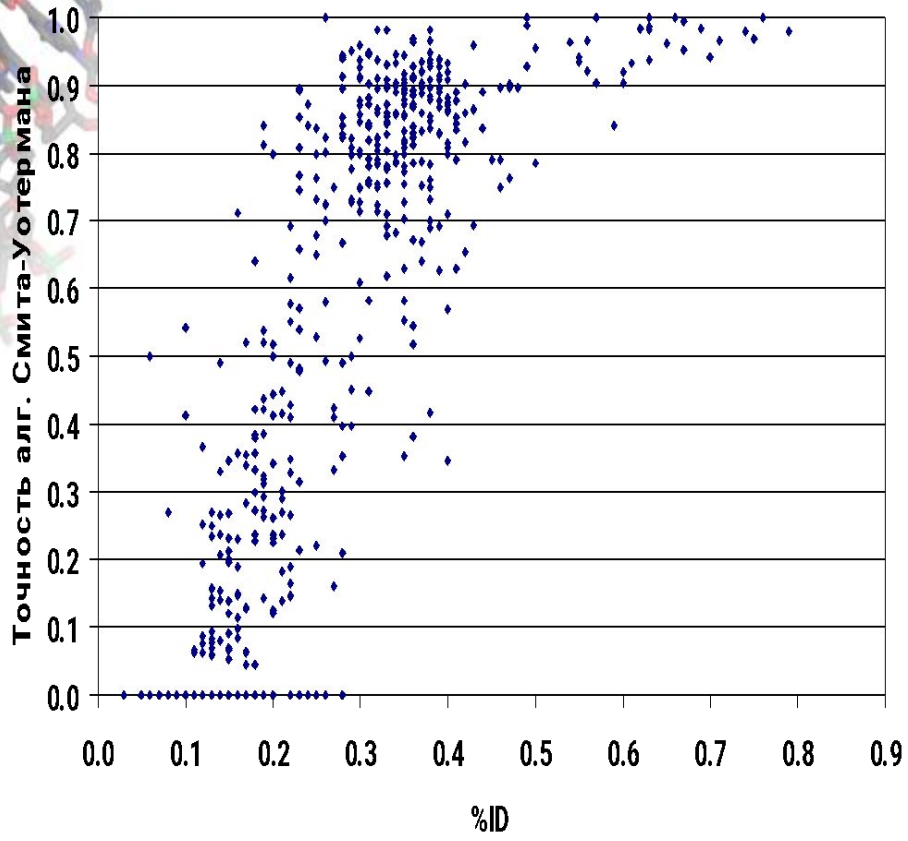
1 16 6

* ****************** ******

lk..C...nqliPPFWKTCPKGKNLCYK...mtmraapmvPVKRGCidv

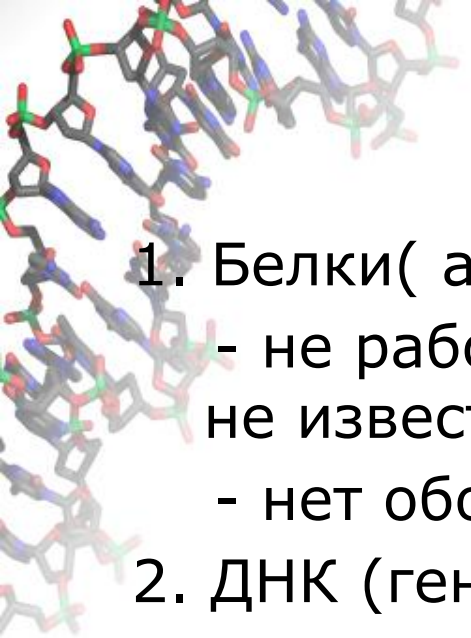
..riCfnhqssqPQTTKTCSPGESSCYHkqwsdfrgt...IIERGC..g

%ID



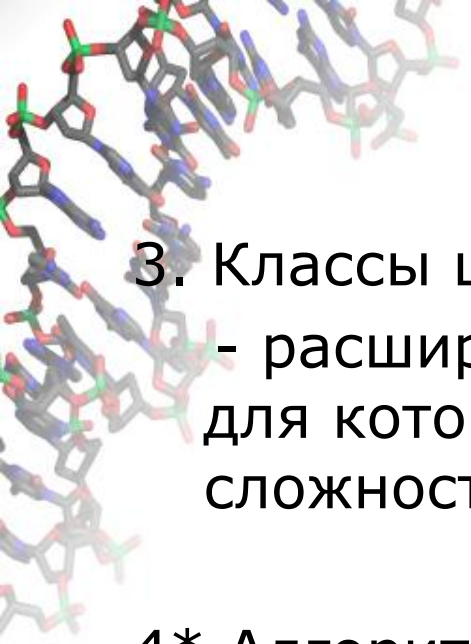

***Алгоритм Смита-Уотермана (SW) не может восстановить структурное выр-ние при ID<0.3***

| %ID | SW точность (acc) |
|---|---|
| < 0,1 | 0,07 |
| 0,1-0,3 | 0,306 |
| 0,3-0,4 | 0,818 |
| >0,4 | 0,893 |



# Проблемы:

1. Белки( алгоритм Смита-Уотермана):
   - не работает при слабом сходстве; причина этого не известна;
   - нет обоснования для штрафов за делеции
2. ДНК (геномы)
   - недостаток быстродействия
   - нет эталонных выравниваний



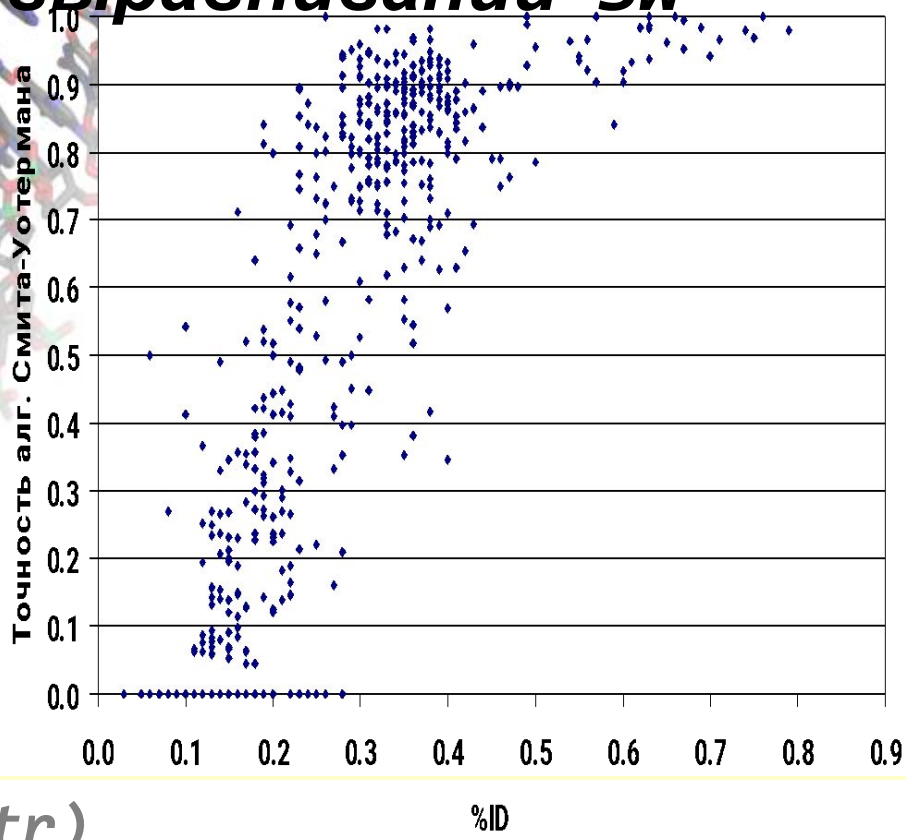
# Проблемы

3. Классы штрафных функций:

   - расширить классы штрафных функций делеций, для которых существуют алгоритмы данной сложности

4* Алгоритмы: анализ общих основ, выяснение границ применимости

# 1. Причины плохого качества выравниваний SW




***Острова*** **– безделеционные фрагменты выравниваний.**

***Вес острова*** **– сумма весов сопоставлений**

*Str)*

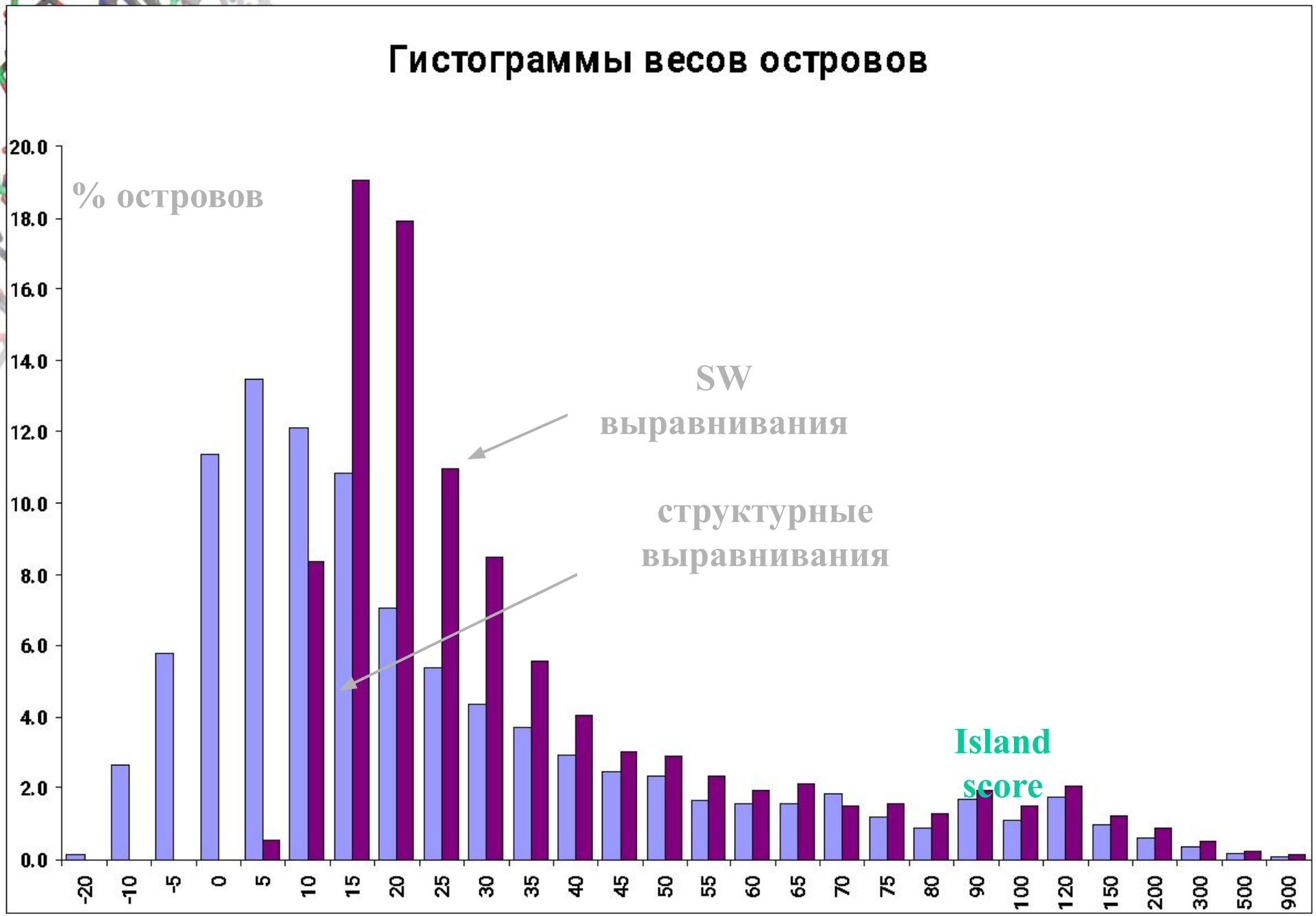
lkCnqli...PPFWKTCPKGKNLCYKmtmraapmvPVKRGCidv

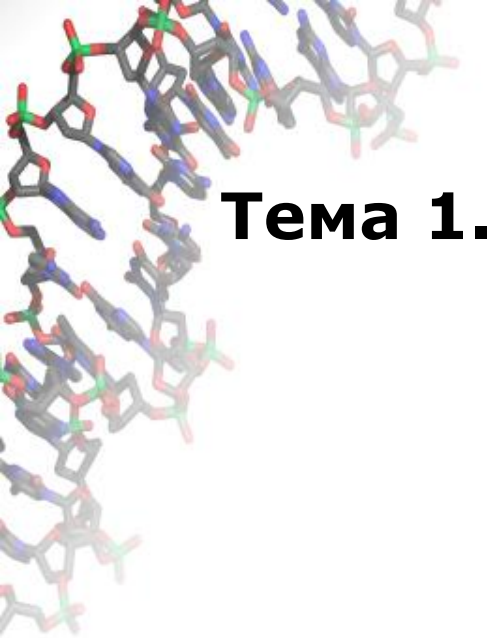
riCfnhqssqPQTTKTCSPGESSCYHkqwsdfrgtIIERGCg..

^^^^^^^ ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

Остров1 Остров 2

# 1. Причины плохого качества выравниваний SW





# Тема 1. Динамическое программирование

# Рекурсия для глобального выравнивания (δ(L)=kL)

- v, w - слова; a, b – буквы
- S(v, w) – вес оптимального выравнивания v, w.

- S(va, wb) = max{
  S(v, w) + m(a,b), // сопоставление последних букв
  S(v, wb) – k; // удаление посл. буквы в 1-м слове
  S(va, w) - k // удаление посл. буквы в 2-м слове

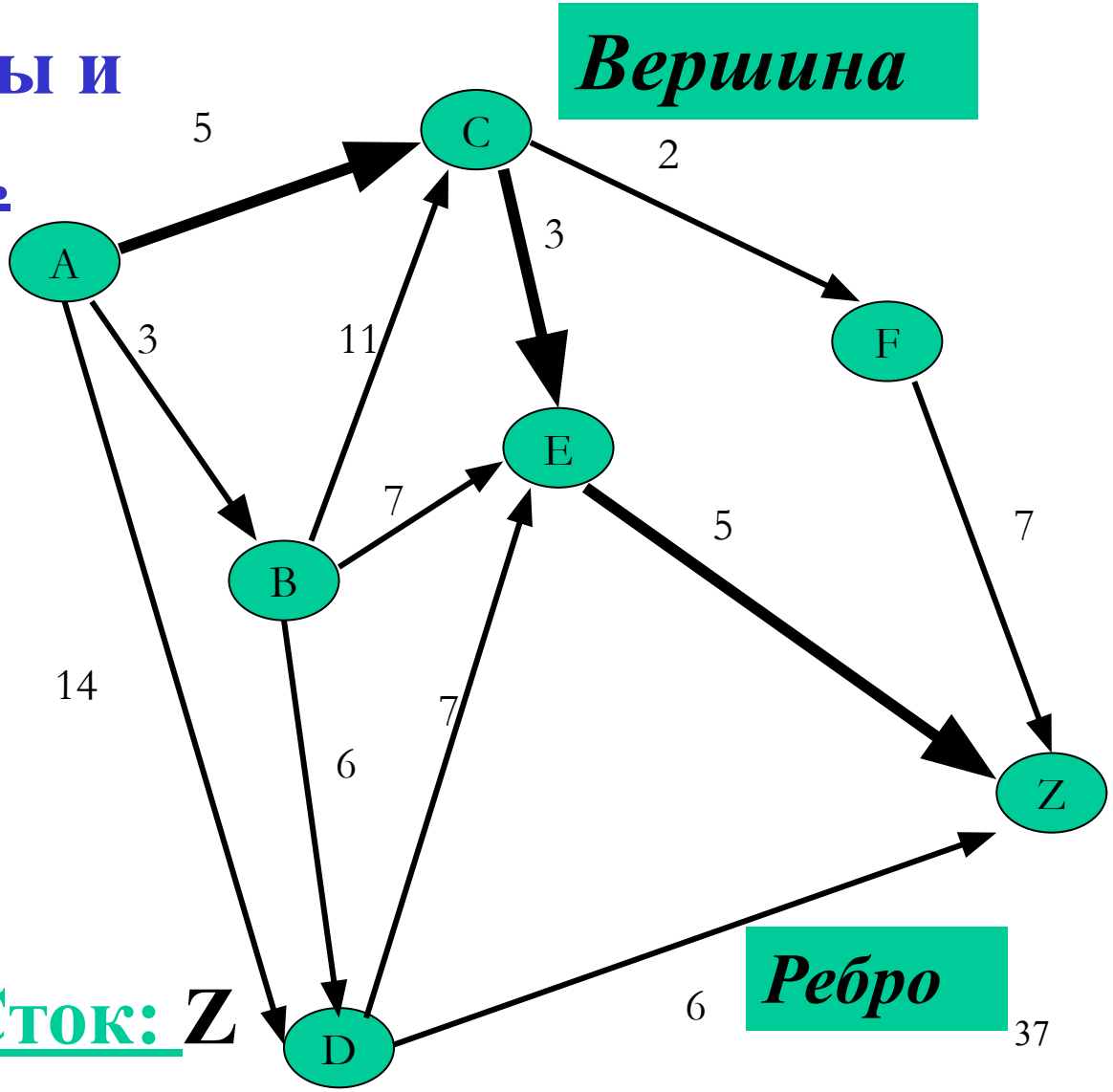
}

# Ориентированный ациклический граф с весами на ребрах

**Ребра направлены и снабжены весами.**

**Путь: ABCE**
**W(ABCE) =**
**= 3+11+ 3 = 17**

**Нет циклов**




**Источник: A;** **Сток: Z**

## Пути (примеры):

**BEZ = {(BE), (EZ)} (длина 2);**

**вес W(BEZ) = *7 + 5 = 12***

**BCEZ = {(BC), (CE), (EZ)} (длина 3);**

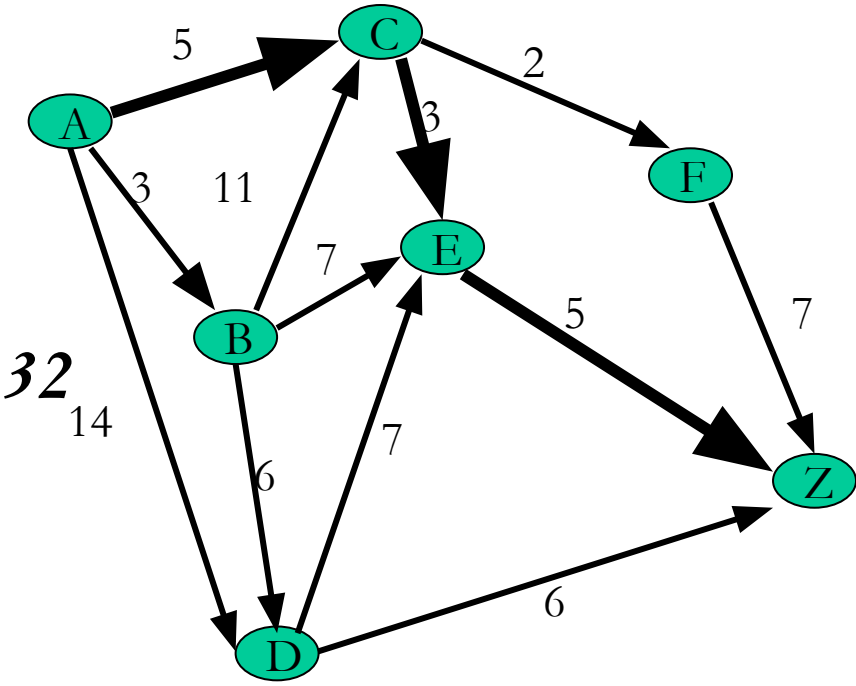
**W(BCEZ) = *11+ 3 +5 = 19***

***Полный путь* (длина 4); *:***

**ADBEZ =**

**={(AD), (DB), (BE), (EZ)}**

**W(ADBEZ) = *14+6+7 + 5 = 32***



***Полные пути –***

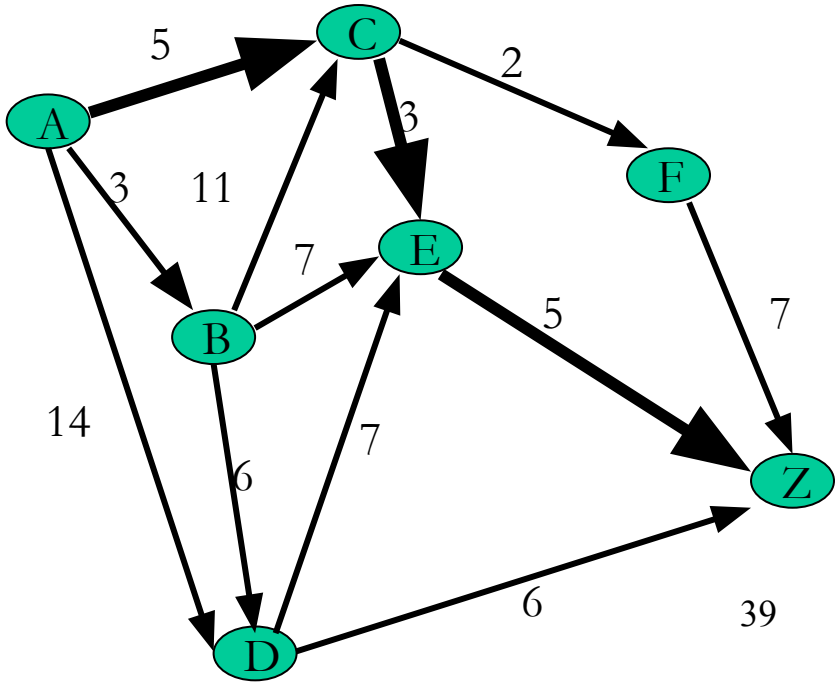
***пути из источника в сток (примеры):***

**ADEZ: длина = 3;**

**вес W(ADEZ) = *14+ 7 + 5 = 26;***

**ABCFZ: длина = 4;**

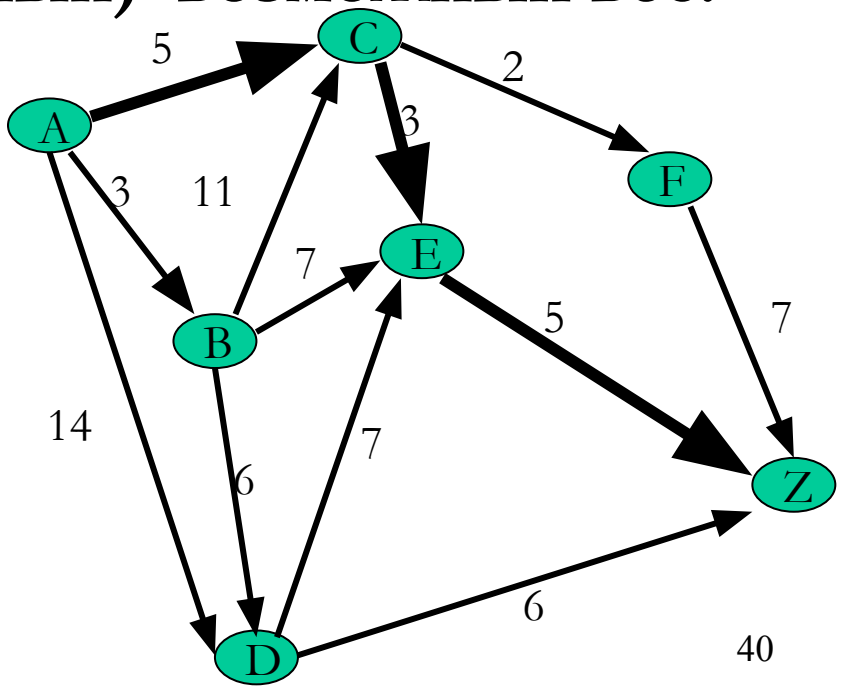
**вес W(ABCFZ) = *3+7+ 2 + 7 = 19***



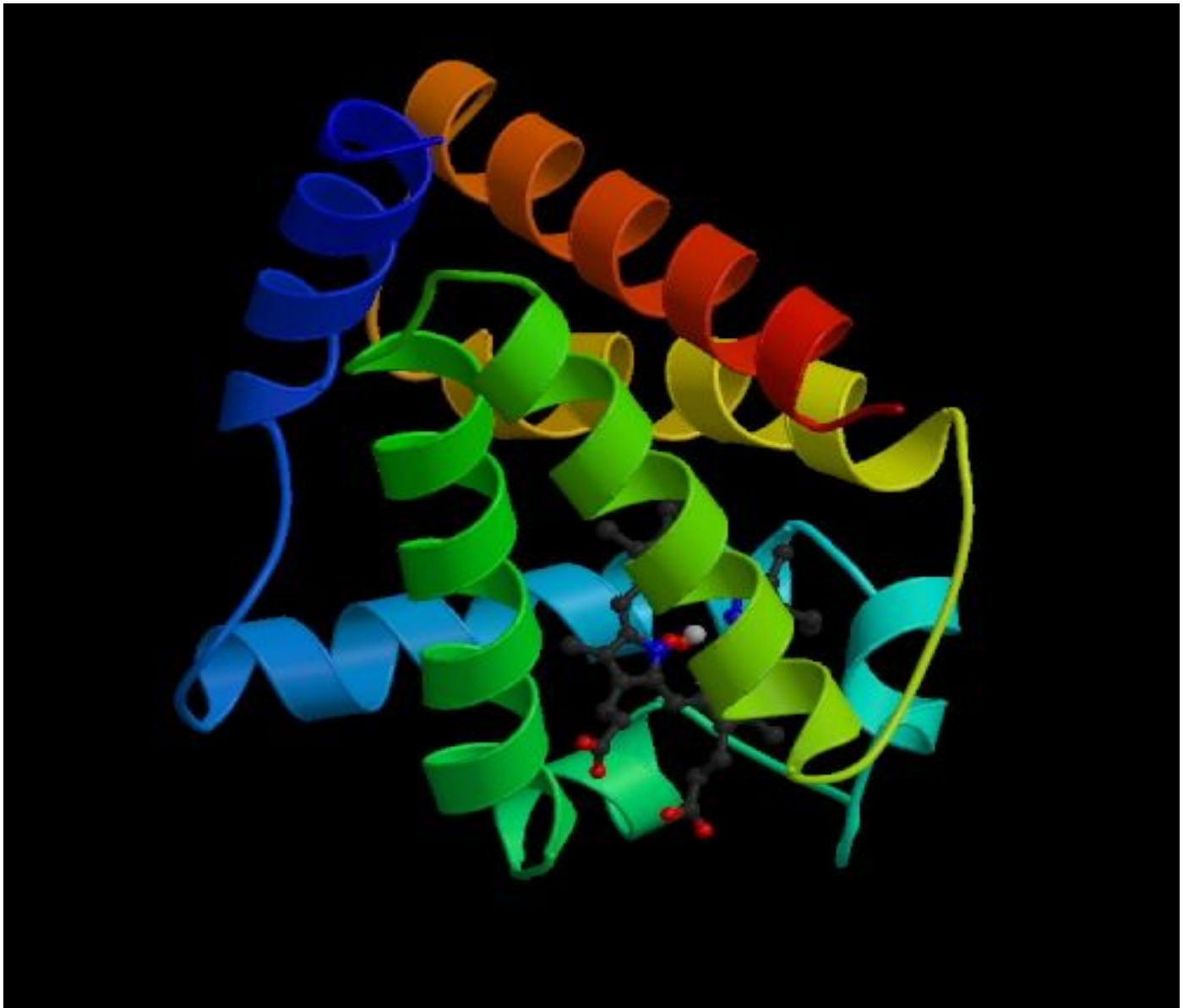
***ДАНО:*** **Ориентированный ациклический граф с весами на ребрах**

**G =< *V, E, W; A, Z*>**

***ЗАДАЧА 1 (задача Беллмана)*** **Найти оптимальный полный путь, т.е. полный путь, имеющий минимальный (максимальный) возможный вес.**



# Пример: предсказание 3D структуры белков (гемоглобин, код белка 1ash, цепь A)



RCSB PDB - 1ASH Sequence Report - Mozilla Firefox

Файл Правка Вид Журнал Закладки Инструменты Справка

http://www.rcsb.org/pdb/explore/remediatedSequence.do?structureId=1ASH

Самые популярные Начальная страница Лента новостей

RCSB PDB - 1ASH Sequence Report

**Tools** Hide

Download: Entries | Ligands
Compare Structures
FTP Services
File Formats
Services: RESTful | SOAP
Widgets

**PDB-101** Hide

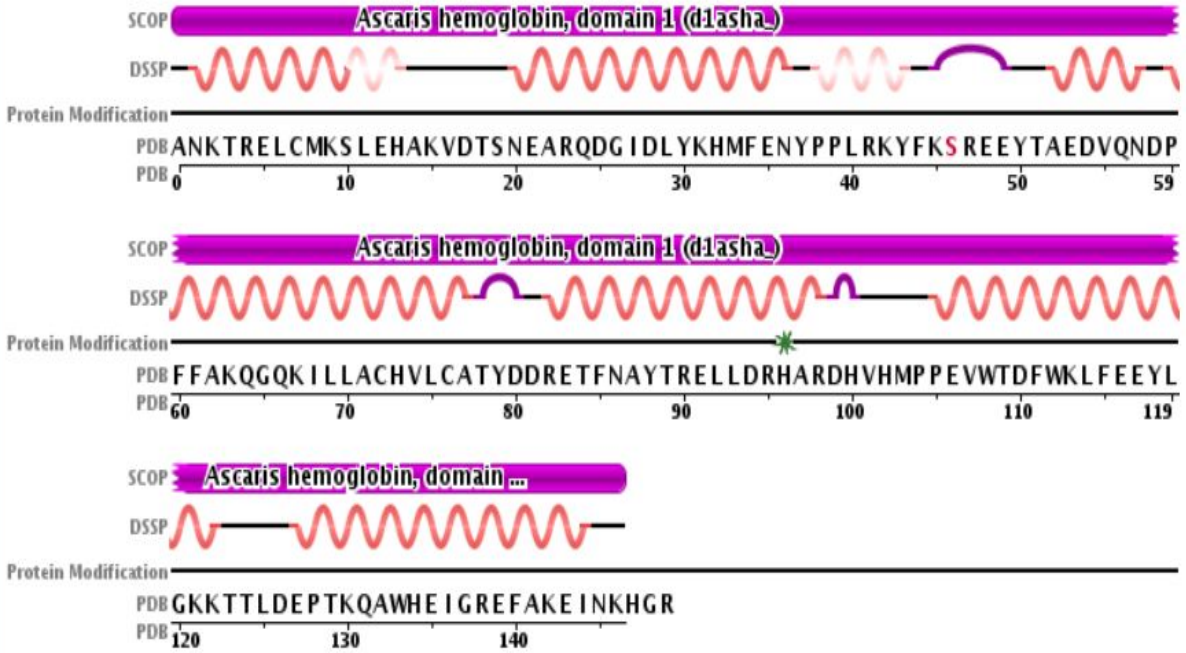
Structural View of Biology
Understanding PDB Data
Molecule of the Month
Educational Resources

**Help** Hide

Launch Help System
Display Settings
Video Tutorials
Glossary of Terms
PDB*Mobile* FAQ

Structural Feature: **Protein Modification**
[hide] [reference] [reference]

0148 1'-heme-L-histidine (chromoprotein, heme, iron, metalloprotein) *RESID*:**AA0329**
*PSI-MOD*:**MOD:00334**




**Protein Modification Legend**

✱ 1'-heme-L-histidine (chromoprotein, heme, iron, metalloprotein)

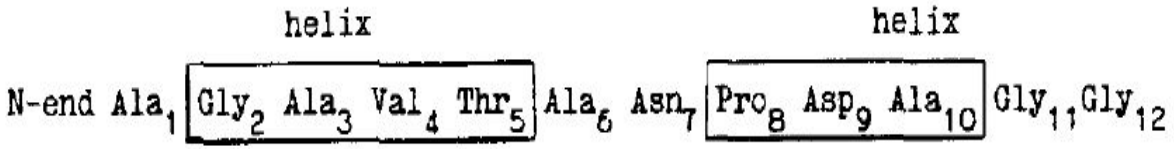
Готово

Fig. 3. Vertices of the Needleman-Wunsch graph (dots) and the path corresponding to the alignment presente

**Дано:** ***последовательность аминокислот***

**Надо:** ***где образуются спирали***

(a)




(b)

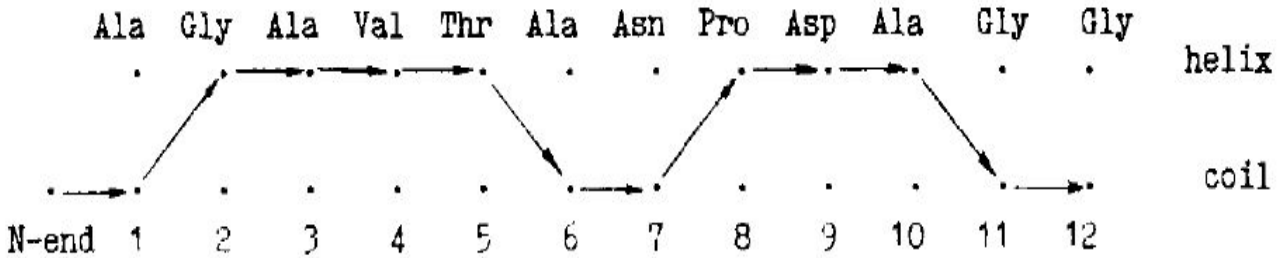
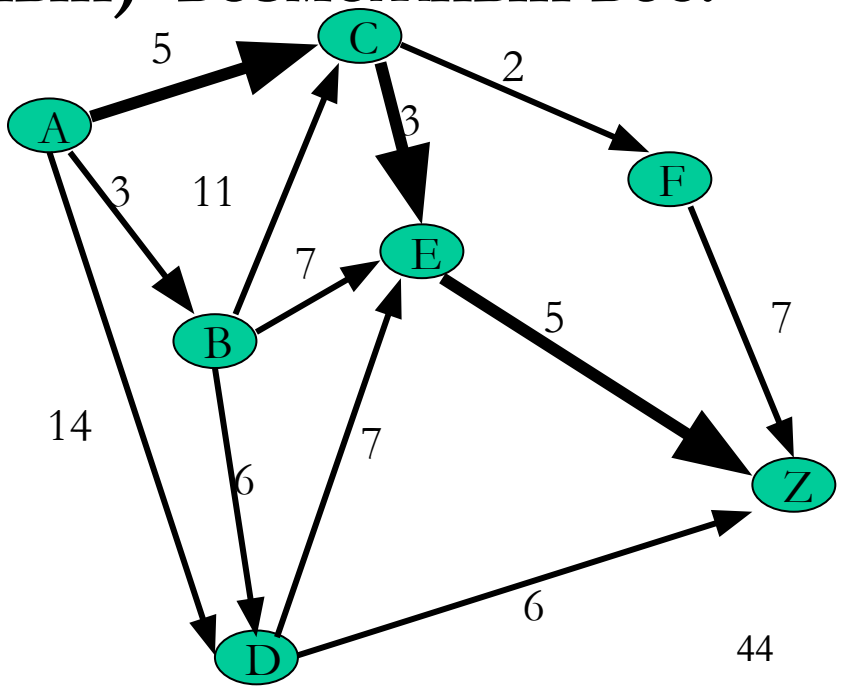



Fig. 4. (A) One of possible arrangements of helices in a polypeptide chain consisting of 12 amino acid tices of the graph describing the secondary structure (helical and coil regions) of a polypeptide chain an corresponding to the above arrangement of the secondary structure. In this example two states are possil unit: *coil* and *helix*. $\Phi_i(\text{coil}) \sim 0$, $\Phi_i(\text{helix}) = f(a_i)$, where $a_i$ is an amino acid at the $i$-th position in $U_i(\text{coil},\text{coil}) = U_i(\text{coil},\text{helix}) = U_i(\text{helix},\text{coil}) = 0$, while $U_i(\text{helix},\text{helix}) = \varepsilon_H$.

***ДАНО:*** **Ориентированный ациклический граф с весами на ребрах**

**G =< *V, E, W; A, Z*>**

***ЗАДАЧА 1 (задача Беллмана)*** **Найти оптимальный полный путь, т.е. полный путь, имеющий минимальный (максимальный) возможный вес.**



# Метод динамического программирования (Алгоритм Беллмана, 1953)

- Проход от стока к источнику:

  из W есть путь в V =>

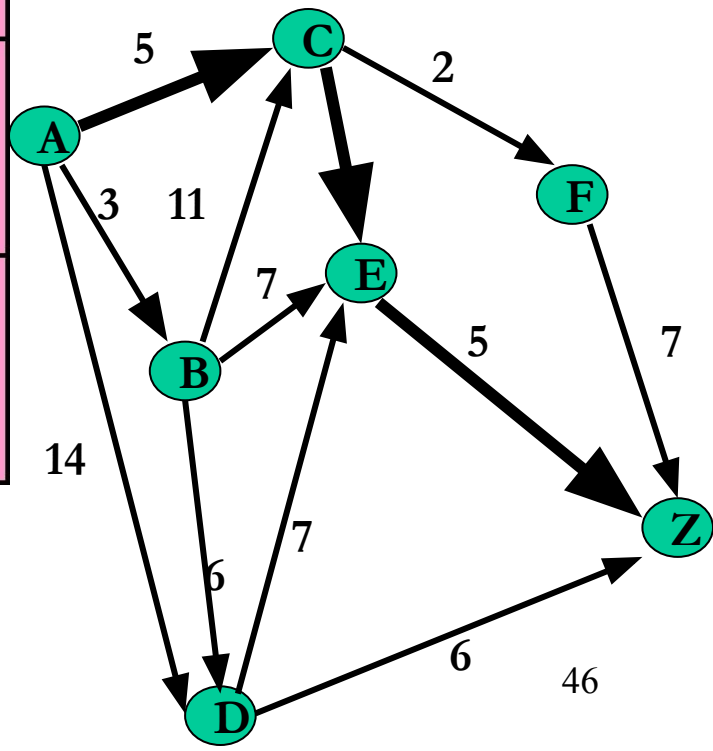
  => W обрабатывается позже, чем V.

- Рекуррентное уравнение

**BestW(A) = min{**
**W(AB) + BestW(B),**
**W(AC) + BestW(C),**
**W(AD) + BestW(D)**
**}**

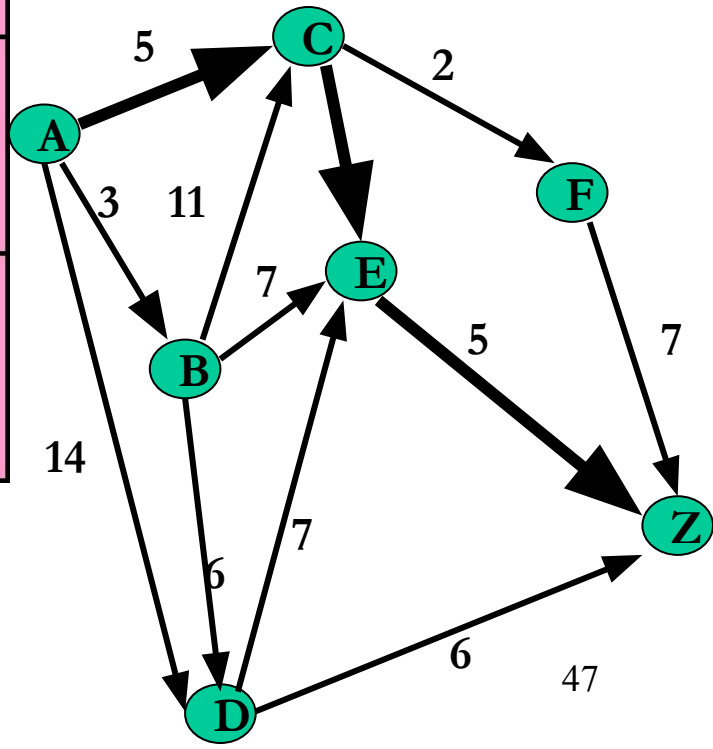
| Ранг | Вершина | Исх. ребра | Вес ребра | Вес след. верш. | Лучш вес для ребра | Лучш вес для верш | Куда идти |
|---|---|---|---|---|---|---|---|
| 0 | Z | xxx | 0 | 0 | 0 | 0 | xxx |
| 1 | F | Z | 7 | 0 | 7 | 7 | Z |
| 1 | E | Z | 5 | 0 | 5 | 5 | Z |
| 2 | D | E | 7 | 5 | 12 | 6 | Z |
| | | Z | 6 | 0 | 6 | | |
| 2 | C | E | 3 | 5 | 8 | 8 | E |
| | | F | 2 | 7 | 9 | | |
| 3 | B | C | 11 | 8 | 19 | 12 | D, E |
| | | D | 6 | 6 | 12 | | |
| | | E | 7 | 5 | 12 | | |
| 4 | A | B | | | | | |
| | | C | | | | | |
| | | D | | | | | |

**BestW(B) =**
**= min{**
**W(BC) + BestW(C),**
**W(BD) + BestW(D),**
**W(BE) + BestW(E),**
**}**



| Ранг | Вершина | Исх. ребра | Вес ребра | Вес след. верш. | Лучш вес для ребра | Лучш вес для верш | Куда идти |
|---|---|---|---|---|---|---|---|
| 0 | Z | xxx | 0 | 0 | 0 | 0 | xxx |
| 1 | F | Z | 7 | 0 | 7 | 7 | Z |
| 1 | E | Z | 5 | 0 | 5 | 5 | Z |
| 2 | D | E | 7 | 5 | 12 | 6 | Z |
| | | Z | 6 | 0 | 6 | | |
| 2 | C | E | 3 | 5 | 8 | 8 | E |
| | | F | 2 | 7 | 9 | | |
| 3 | B | C | 11 | 8 | 19 | 12 | D, E |
| | | D | 6 | 6 | 12 | | |
| | | E | 7 | 5 | 12 | | |
| 4 | A | B | 3 | 12 | 15 | 13 | C |
| | | C | 5 | 8 | 13 | | |
| | | D | 14 | 6 | 20 | | |

**BestW(B) =**
**= min{**
**W(BC) + BestW(C),**
**W(BD) + BestW(D),**
**W(BE) + BestW(E),**
**}**

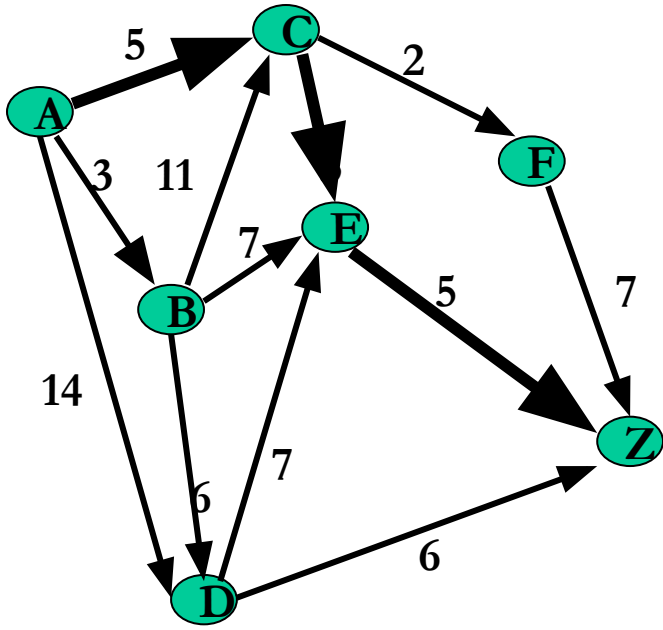



**Best Weight: 13**

**Best Path: ACEZ**

| Ранг | Вершина | Исх. ребра | Вес ребра | Вес след. верш. | Лучший вес для ребра | Лучший вес для вершины | Куда идти |
|---|---|---|---|---|---|---|---|
| 0 | Z | xxx | 0 | 0 | 0 | 0 | xxx |
| 1 | F | Z | 7 | 0 | 7 | 7 | Z |
| 1 | E | Z | 5 | 0 | 5 | 5 | Z |
| 2 | D | E | 7 | 5 | 12 | 6 | Z |
| | | Z | 6 | 0 | 6 | | |
| 2 | C | E | 3 | 5 | 8 | 8 | E |
| | | F | 2 | 7 | 9 | | |
| 3 | B | C | 11 | 8 | 19 | 12 | D, E |
| | | D | 6 | 6 | 12 | | |
| | | E | 7 | 5 | 12 | | |
| 4 | A | B | 3 | 12 | 15 | 13 | C |
| | | C | 5 | 8 | 13 | | |
| | | D | 14 | 6 | 20 | | |

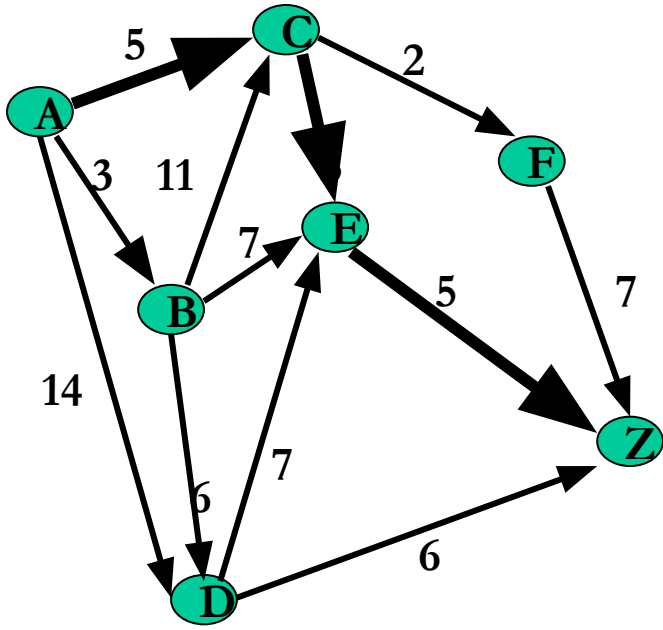
**BestW(A) =**
**= min{**
**W(AB) + BestW(B),**
**W(AC) + BestW(C),**
**W(AD) + BestW(D),**
**}**




**Для любой вершины T:**

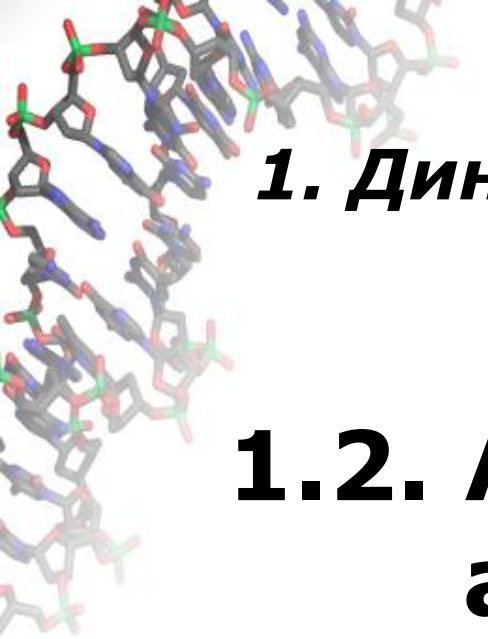
***BestW (T)* = min{**
**W(T $N_1$) + BestW($N_1$),**
**…..,**
**W(T $N_t$) + BestW($N_t$),**
**} где**
**$N_1$, ..., $N_t$ – все наследники T**

| Ранг | Вершина | Исх. ребра | Вес ребра | Вес след. верш. | Лучший вес для ребра | Лучший вес для вершины | Куда идти |
|---|---|---|---|---|---|---|---|
| 0 | Z | xxx | 0 | 0 | 0 | 0 | xxx |
| 1 | F | Z | 7 | 0 | 7 | 7 | Z |
| 1 | E | Z | 5 | 0 | 5 | 5 | Z |
| 2 | D | E | 7 | 5 | 12 | 6 | Z |
| | | Z | 6 | 0 | 6 | | |
| 2 | C | E | 3 | 5 | 8 | 8 | E |
| | | F | 2 | 7 | 9 | | |
| 3 | B | C | 11 | 8 | 19 | 12 | D, E |
| | | D | 6 | 6 | 12 | | |
| | | E | 7 | 5 | 12 | | |
| 4 | A | B | 3 | 12 | 15 | 13 | C |
| | | C | 5 | 8 | 13 | | |
| | | D | 14 | 6 | 20 | | |




**ВРЕМЯ РАБОТЫ ~ к-во РЕБЕР**

**ПАМЯТЬ ~ к-во ВЕРШИН**



*1. Динамическое программирование, графы и алгебра*

# 1.2. Алгебраическая основа алгоритма Беллмана

# Задача-подсказка

$$S =$$

$$= a_1 \cdot b_1 + a_1 \cdot b_2 + \ldots + a_1 \cdot b_{1000} +$$

$$+ a_2 \cdot b_1 + a_2 \cdot b_2 + \ldots + a_2 \cdot b_{1000} +$$

$$+ \quad \ldots \quad +$$

$$+ a_{1000} \cdot b_1 + a_{1000} \cdot b_2 + \ldots + a_{1000} \cdot b_{1000}$$

# Решение

$$S = a_1 \cdot (b_1 + b_2 + \ldots + b_{1000}) +$$
$$+ a_2 \cdot (b_1 + b_2 + \ldots + b_{1000}) +$$
$$+ \ldots$$
$$+$$
$$+ a_{1000} \cdot (b_1 + b_2 + \ldots + b_{1000}) =$$
$$= (a_1 + a_2 + \ldots + a_{1000}) \cdot (b_1 + b_2 + \ldots + b_{1000})$$

*** Алгоритм ***

**A =** $a_1 + a_2 + \ldots + a_{1000}$ // 999 операций

**B =** $b_1 + b_2 + \ldots + b_{1000}$ // 999

# Повторение: 1-й класс

☺

***Сочетательный закон (ассоциативность):***

| ***Сложение*** | ***Умножение*** |
|---|---|
| ***(a+b)+c = a+(b+c)*** | ***(a*b)*c = a*(b*c)*** |

***Переместительный закон (коммутативность):***

| ***Сложение*** | ***Умножение*** |
|---|---|
| ***a+b = b+a*** | ***a*b = b*a*** |

***Нейтральный элемент:***

| ***Сложение*** | ***Умножение*** |
|---|---|
| ***a+0 = 0+a =a*** | ***a*1 = 1*a = a*** |

***Обратные элементы (3-й класс☺) :***

| ***Сложение*** | ***Умножение*** |
|---|---|
| ***a+(-a) = 0*** | ***a*(1/a) = 1*** |

# Повторение: 1-й класс

☺

***Сочетательный закон (ассоциативность):***

| ***Сложение*** | ***Умножение*** |
|---|---|
| $(a+b)+c = a+(b+c)$ | $(a*b)*c = a*(b*c)$ |

***Переместительный закон (коммутативность):***

| ***Сложение*** | ***Умножение*** |
|---|---|
| $a+b = b+a$ | $a*b = b*a$ |

***Нейтральный элемент:***

| ***Сложение*** | ***Умножение*** |
|---|---|
| $a+0 = 0+a = a$ | $a*1 = 1*a = a$ |

- ***Обратные элементы (3-й класс☺) :***

| ***Сложение*** | ***Умножение*** |
|---|---|
| $a+(-a) = 0$ | $a*(1/a) = 1a$ |

- ***РАСПРЕДЕЛИТЕЛЬНЫЙ ЗАКОН (ДИСТРИБУТИВНОСТЬ)***

***умножение относительно сложения***

$(a+b)*c = a*c + b*c$  $a*(b+c) = a*b+a*c$

# Мультипликативные веса путей

**BEZ = {(BE), (EZ)} (длина 2);**

**вес W(BEZ) = *7 + 5 = 12***

***мультипликативный вес (м-вес)***

***WM(BEZ) = 7•5 = 35***

**BCEZ = {(BC), (CE), (EZ)} (длина 3);**

**W(BCEZ) = *11+ 3 +5 = 19***

***WM(BCEZ)=11•3•5=165***

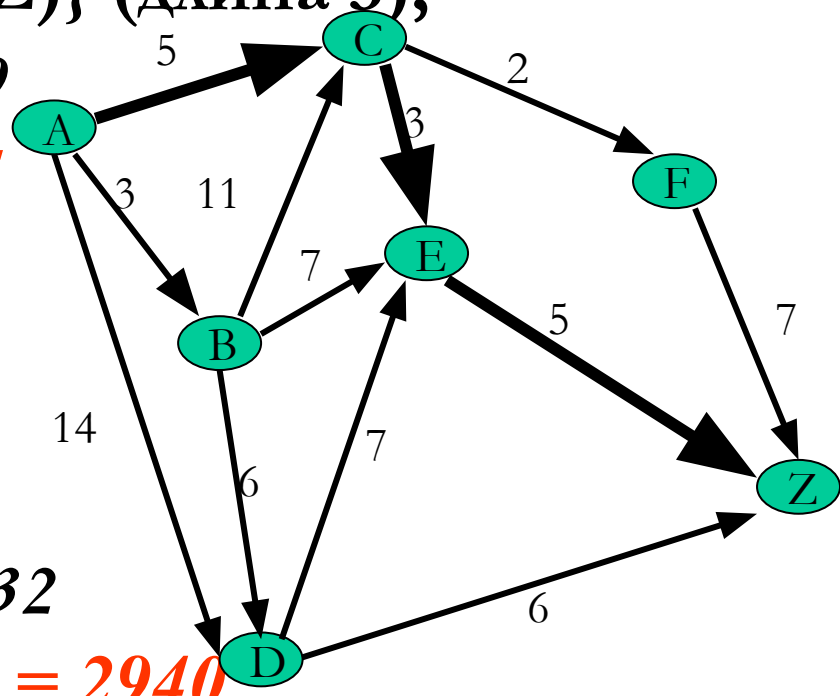
***Полный путь* (длина 4); :**

**ADBEZ =**

**={(AD), (DB), (BE), (EZ)}**

**W(ADBEZ) = *14+6+7 + 5 = 32***

***WM(ADBEZ) = 14•6•7•5 = 2940***

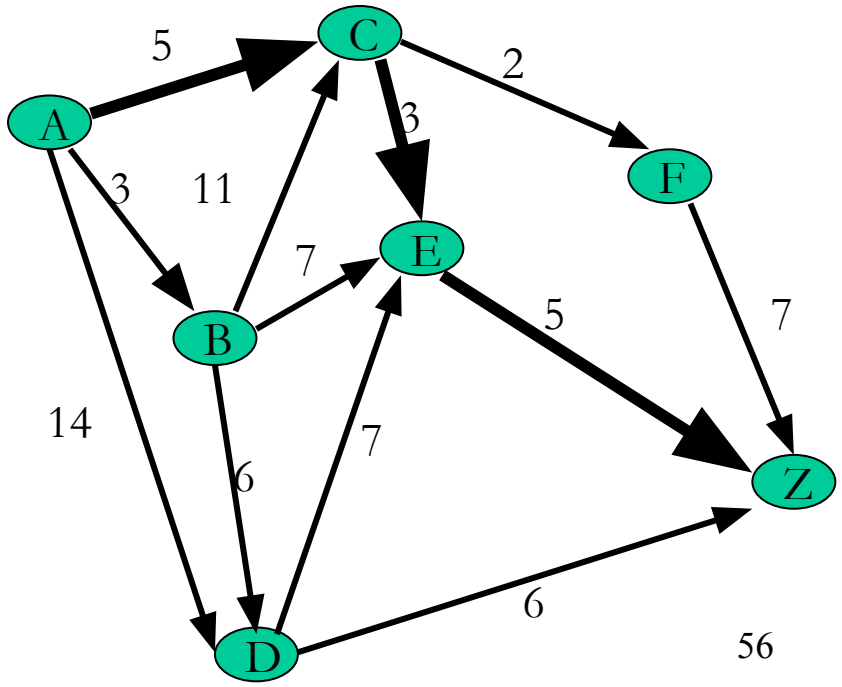


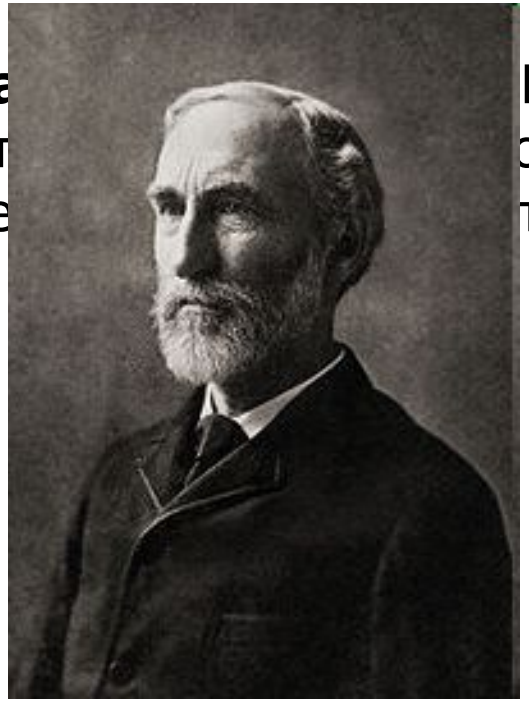
***ДАНО:*** **Ориентированный ациклический граф с весами на ребрах**

**G =< *V, E, W; A, Z*>**

***ЗАДАЧА 2 («задача Больцмана»)*** **Найти сумму мультипликативных весов всех полных путей.**

**Лю́двиг Больцман** (нем. Ludwig Eduard Boltzmann, 1844 - 1906), основатель статистической механики и молекулярно-кинетической теории





**Лю́двиг Больцман** (Ludwig Eduard Boltzmann, 1844 – 1906; Австро-Венгрия, Италия), основатель статистической механики и молекулярно-кинетической теории

**Эрнст Изинг** (Ernst Ising, 1900-1998, Германия-США) - физик, позже - педагог, автор модели Изинга (см. предсказание спиралей в белке и т. п.)

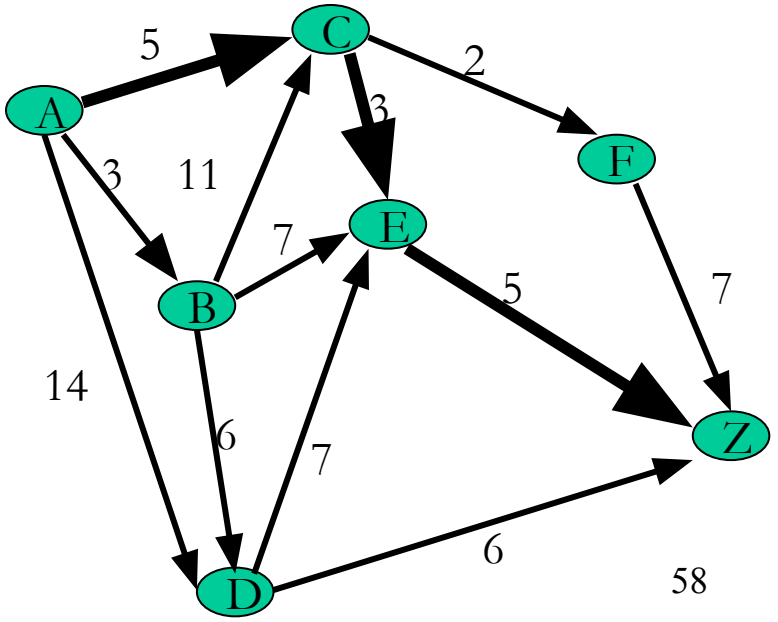
*Интерпретации:*

*1. Вероятность прохода лабиринта:*

***Вершины*** **– города;** ***Ребра*** **- дороги;**

***Вес ребра: вероятность перехода по ребру (сумма вероятностей выхода из вершины может быть меньше 1)***

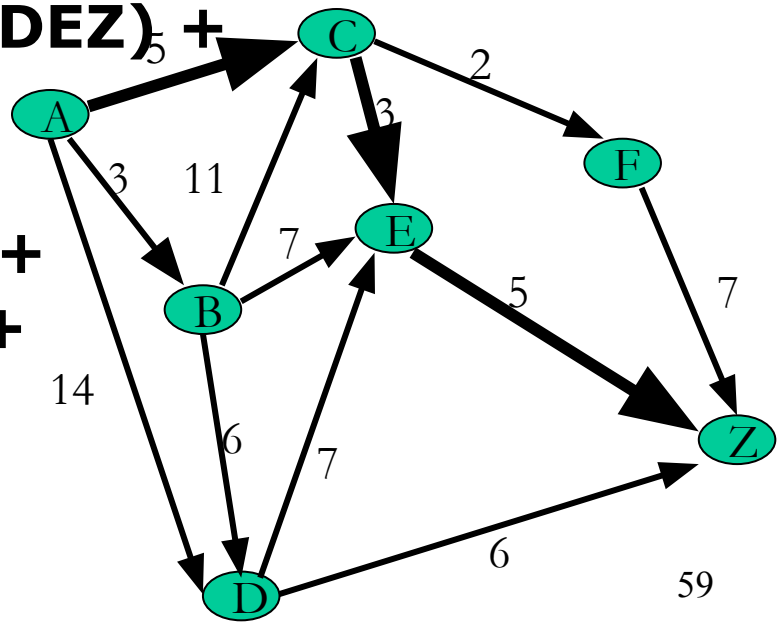
***2. Статистическая физика – без комментариев***



- Проход от стока к источнику:
  из W есть путь в V =>
  => W обрабатывается позже, чем V.
  Пример: вершина B.
- Пути из B в Z: BCEZ, BCFZ, BDZ, BDEZ, BEZ
- **Sum(B) = M(BCEZ) + M(BCFZ) +**
  **+ M(BDZ) + M(BDEZ) +**
  **+ M(BEZ) =**

**= W(BC)*M(CEZ) + W(BC)*M(CFZ) +**
**+W(BD)*M(DZ) + W(BD)*M(DEZ) +**
**+W(BE)* M(EZ) =**

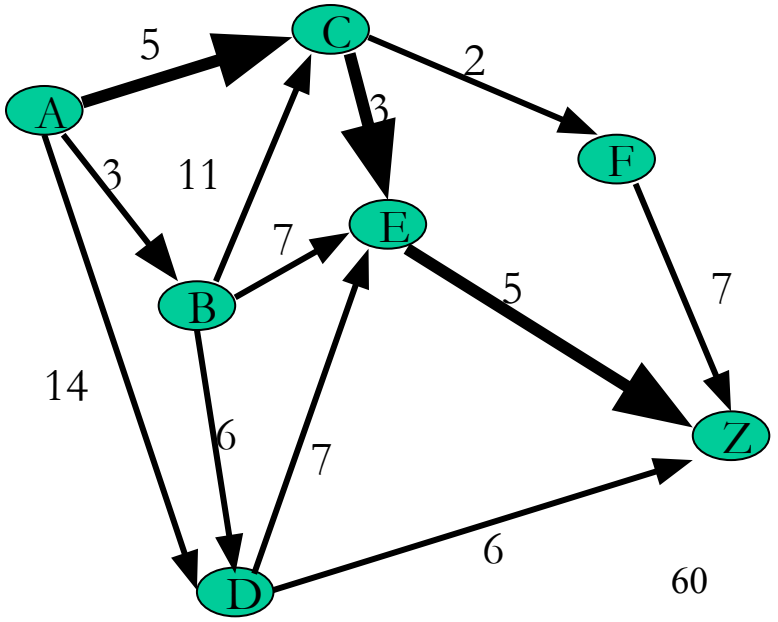
**= W(BC)*(M(CEZ) + M(CFZ)) +**
**+ W(BD)*(M(DZ) + M(DEZ)) +**
**+ W(BE)* M(EZ) = ....**



- Проход от стока к источнику:
  из W есть путь в V =>
  => W обрабатывается позже, чем V.
  Пример: вершина B.
- Пути из B в Z: BCEZ, BCFZ, BDZ, BDEZ, BEZ
- **Sum(B) =...**

**= W(BC)*(M(CEZ) + M(CFZ)) +**
**+ W(BD)*(M(DZ) + M(DEZ)) +**
**+ W(BE)* M(EZ) =**

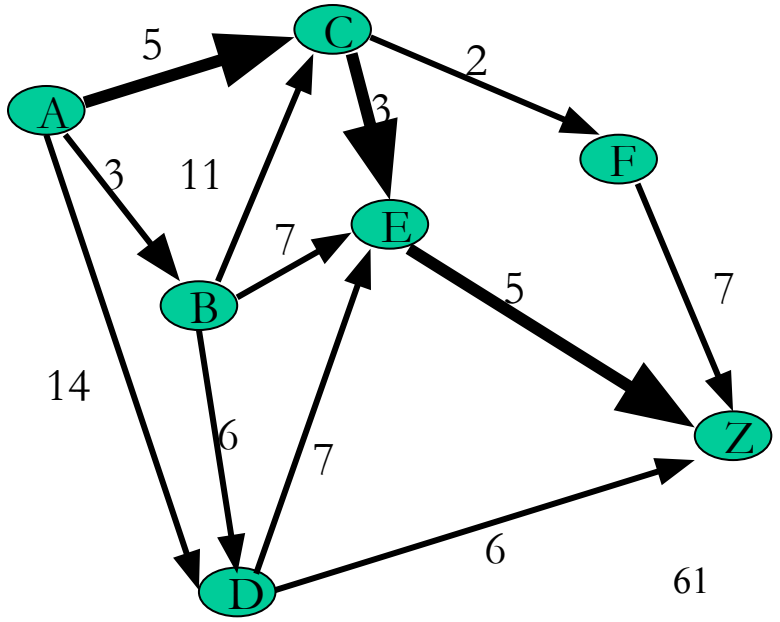
**= W(BC)* Sum(C) +**
**+ W(BD)* Sum(D) +**
**+ W(BE)* Sum(E)**



- Проход от стока к источнику:
  из W есть путь в V =>
  => W обрабатывается позже, чем V.
- Пример: вершина B.
- Пути из B в Z: BCEZ, BCFZ, BDZ, BDEZ, BEZ
- **Sum(B) = M(BCEZ) + M(BCFZ) +**
  **+ M(BDZ) + M(BDEZ) +**
  **+ M(BEZ) =**

- Рекуррентное уравнение (сумма м-весов):

**Sum(A) =**
**W(AB)*Sum(B) +**
**+ W(AC)*Sum(C) +**
**+ W(AD)*Sum(D)**
**}**

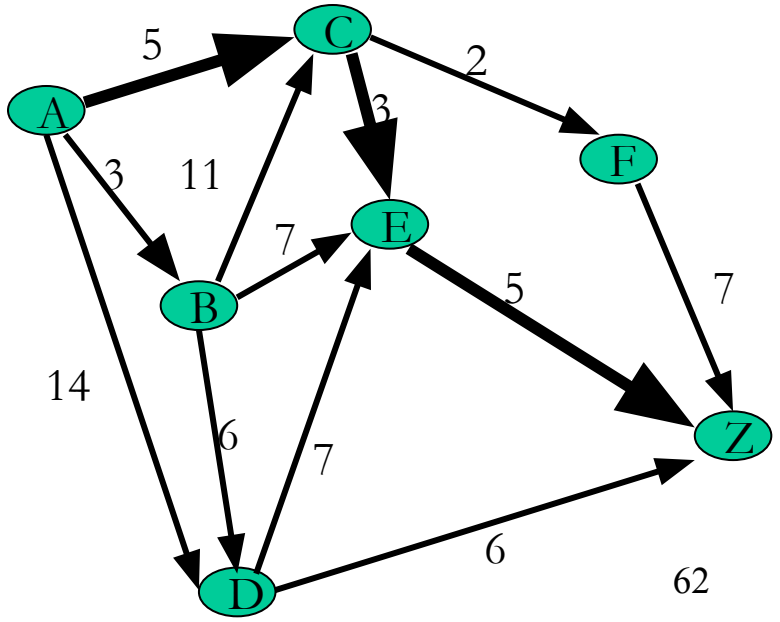


- Проход от стока к источнику:
  из W есть путь в V =>
  => W обрабатывается позже, чем V.
- Рекуррентное уравнение (минимальный путь)

**BestW(A) = min{**
**W(AB) + BestW(B),**
**W(AC) + BestW(C),**
**W(AD) + BestW(D)**
**}**

- Рекуррентное уравнение (сумма м-весов):

**Sum(A) =**
**W(AB)*Sum(B) +**
**+ W(AC)*Sum(C) +**
**+ W(AD)*Sum(D)**
**}**



# Что использовали?

***Сочетательный закон (ассоциативность):***

***Сложение*** ***Умножение***

$(a+b)+c = a+(b+c)$ $(a*b)*c = a*(b*c)$

***Переместительный закон (коммутативность):***

***Сложение*** ***Умножение***

$a+b = b+a$ $a*b = b*a$

***Нейтральный элемент:***

***Сложение*** ***Умножение***

$a+0 = 0+a = a$ $a*1 = 1*a = a$

- ***Обратные элементы (3-й класс☺) :***

***Сложение*** ***Умножение***

$a+(-a) = 0$ $a*(1/a) = 1a$

- ***РАСПРЕДЕЛИТЕЛЬНЫЙ ЗАКОН (ДИСТРИБУТИВНОСТЬ)***

***умножение относительно сложения***

$(a+b)*c = a*c + b*c$ $a*(b+c) = a*b+a*c$

# Что использовали?

***Сочетательный закон (ассоциативность):***

***Сложение*** ***Умножение***

$(a+b)+c = a+(b+c)$ $(a*b)*c = a*(b*c)$

***Переместительный закон (коммутативность):***

***Сложение*** ***Умножение***

$a+b = b+a$ $a*b = b*a$

***Нейтральный элемент:***

***Сложение*** ***Умножение***

$a+0 = 0+a = a$ $a*1 = 1*a = a$

***Обратные элементы (3-й класс☺) :***

***Сложение*** ***Умножение***

$a+(-a) = 0$ $a*(1/a) = 1a$

- ***РАСПРЕДЕЛИТЕЛЬНЫЙ ЗАКОН (ДИСТРИБУТИВНОСТЬ)***

***умножение относительно сложения***

$(a+b)*c = a*c + b*c$ $a*(b+c) = a*b+a*c$

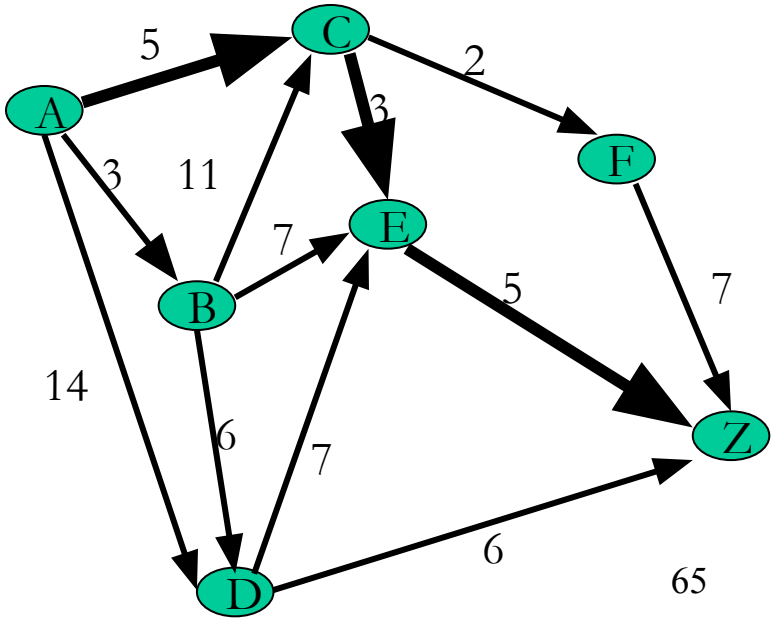
- Sum(B) =
- = M(BCEZ) + M(BCFZ) + M(BDZ) + M(BDEZ) + M(BEZ) =

= W(BC)*M(CEZ) + W(BC)*M(CFZ) +
+W(BD)*M(DZ) + W(BD)*M(DEZ) +
+W(BE)* M(EZ) =

= W(BC)*(M(CEZ) + M(CFZ)) +
+ W(BD)*(M(DZ) + M(DEZ)) +
+ W(BE)* M(EZ) =

= W(BC)*(M(CEZ) + M(CFZ)) +
+ W(BD)*(M(DZ) + M(DEZ)) +
+ W(BE)* M(EZ) =

= W(BC)* Sum(C) +
+ W(BD)* Sum(D) +
+ W(BE)* Sum(E)



# Что использовали?

***Сочетательный закон (ассоциативность):***

***Сложение*** $(a+b)+c = a+(b+c)$

***Умножение*** $(a*b)*c = a*(b*c)$

***Переместительный закон (коммутативность):***

***Сложение*** $a+b = b+a$

***Умножение*** $a*b = b*a$

***Нейтральный элемент:***

***Сложение*** $a+0 = 0+a = a$

***Умножение*** $a*1 = 1*a = a$

- ***Обратные элементы (3-й класс☺) :***

***Сложение*** $a+(-a) = 0$

***Умножение*** $a*(1/a) = 1a$

- ***РАСПРЕДЕЛИТЕЛЬНЫЙ ЗАКОН (ДИСТРИБУТИВНОСТЬ)***

***умножение относительно сложения***

$(a+b)*c = a*c + b*c$ $a*(b+c) = a*b+a*c$

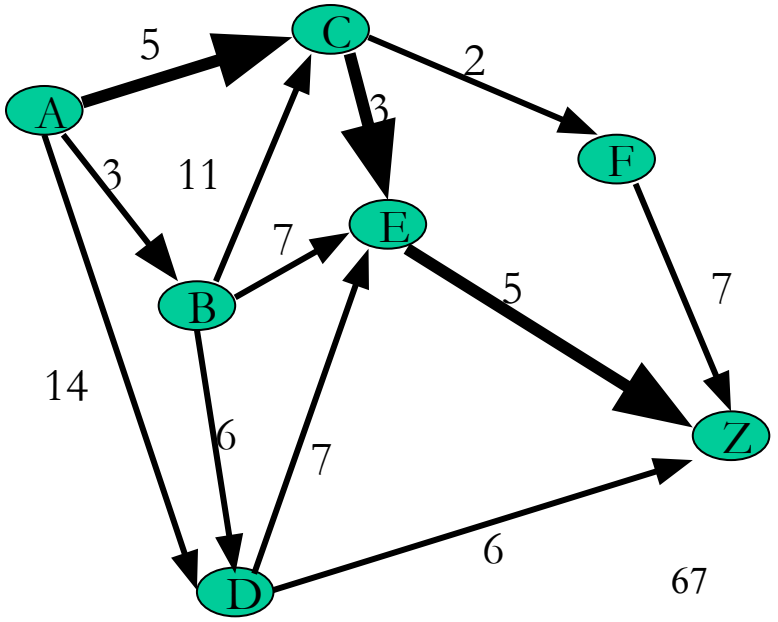
- Sum(B) =
- = M(BCEZ) + M(BCFZ) + M(BDZ) + M(BDEZ) + M(BEZ) =

= W(BC)*M(CEZ) + W(BC)*M(CFZ) +
+W(BD)*M(DZ) + W(BD)*M(DEZ) +
+W(BE)* M(EZ) =

= W(BC)*(M(CEZ) + M(CFZ)) +
+ W(BD)*(M(DZ) + M(DEZ)) +
+ W(BE)* M(EZ) =

= W(BC)*(M(CEZ) + M(CFZ)) +
+ W(BD)*(M(DZ) + M(DEZ)) +
+ W(BE)* M(EZ) =

= W(BC)* Sum(C) +
+ W(BD)* Sum(D) +
+ W(BE)* Sum(E)



# Что использовали?

***Сочетательный закон (ассоциативность):***

***Сложение*** $(a+b)+c = a+(b+c)$

***Умножение*** $(a*b)*c = a*(b*c)$

***Переместительный закон (коммутативность):***

***Сложение*** $a+b = b+a$

***Умножение*** $a*b = b*a$

***Нейтральный элемент:***

***Сложение*** $a+0 = 0+a = a$

***Умножение*** $a*1 = 1*a = a$

- ***Обратные элементы (3-й класс☺) :***

***Сложение*** $a+(-a) = 0$

***Умножение*** $a*(1/a) = 1a$

- ***РАСПРЕДЕЛИТЕЛЬНЫЙ ЗАКОН (ДИСТРИБУТИВНОСТЬ)***

***умножение относительно сложения***

$(a+b)*c = a*c + b*c$ $\quad a*(b+c) = a*b+a*c$

# Что использовали?

***Сочетательный закон (ассоциативность):***

***Сложение*** $(a+b)+c = a+(b+c)$

***Умножение*** $(a*b)*c = a*(b*c)$

***Переместительный закон (коммутативность):***

***Сложение*** $a+b = b+a$

***Умножение*** $a*b = b*a$

***Нейтральный элемент:***

***Сложение*** $a+0 = 0+a =a$

***Умножение*** $a*1 = 1*a = a$

- ***Обратные элементы (3-й класс☺) :***

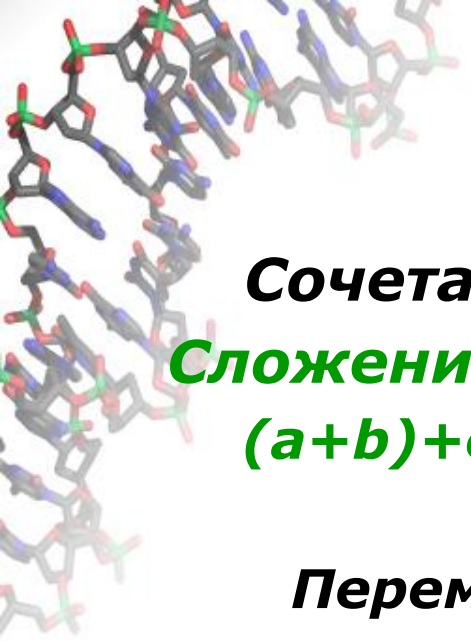
***Сложение*** $a+(-a) = 0$

***Умножение*** $a*(1/a) = 1a$

- ***РАСПРЕДЕЛИТЕЛЬНЫЙ ЗАКОН (ДИСТРИБУТИВНОСТЬ)***

***умножение относительно сложения***

$(a+b)*c = a*c + b*c$ $a*(b+c) = a*b+a*c$



# Это называется полукольцо

☺

***Сочетательный закон (ассоциативность):***

***Сложение*** ***Умножение***

$(a+b)+c = a+(b+c)$ $(a*b)*c = a*(b*c)$

***Переместительный закон (коммутативность):***

***Сложение***

$a+b = b+a$

***Нейтральный элемент:***

***Умножение***

$a*1 = 1*a = a$

***РАСПРЕДЕЛИТЕЛЬНЫЙ ЗАКОН (ДИСТРИБУТИВНОСТЬ)***

***умножение относительно сложения***

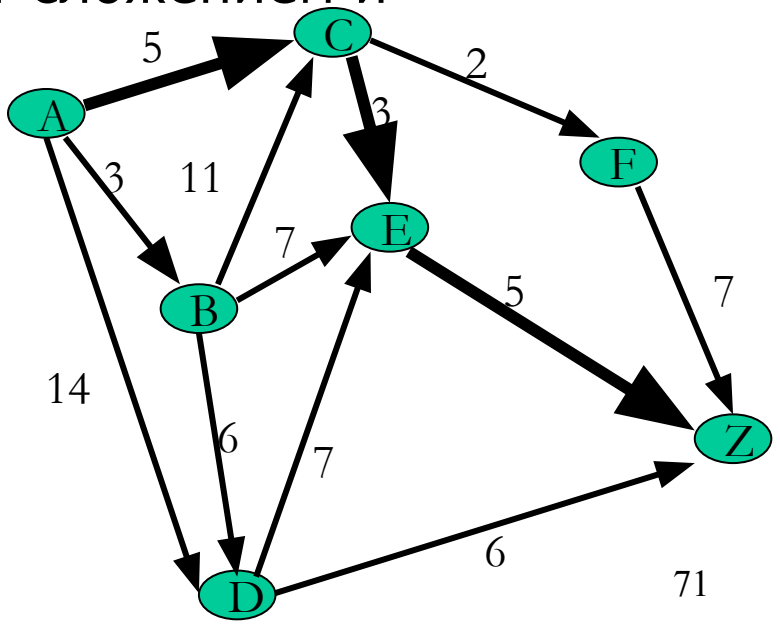
$(a+b)*c = a*c + b*c$ $a*(b+c) = a*b+a*c$

*Полукольцо A* – это множество, на котором заданы две бинарные всюду определенные операции **+** и ***** («сложение» и «умножение»), удовлетворяющие следующим свойствам:

- операции **+** и ***** ассоциативны;
- операция **+** коммутативна, коммутативность операции ***** не обязательна;
- в *A* есть правый нейтральный элемент относительно операции *****;
- Операции и обычно называют сложением и умножением.

**+ - «целевая» операция**

*** - «соединительная» операция**



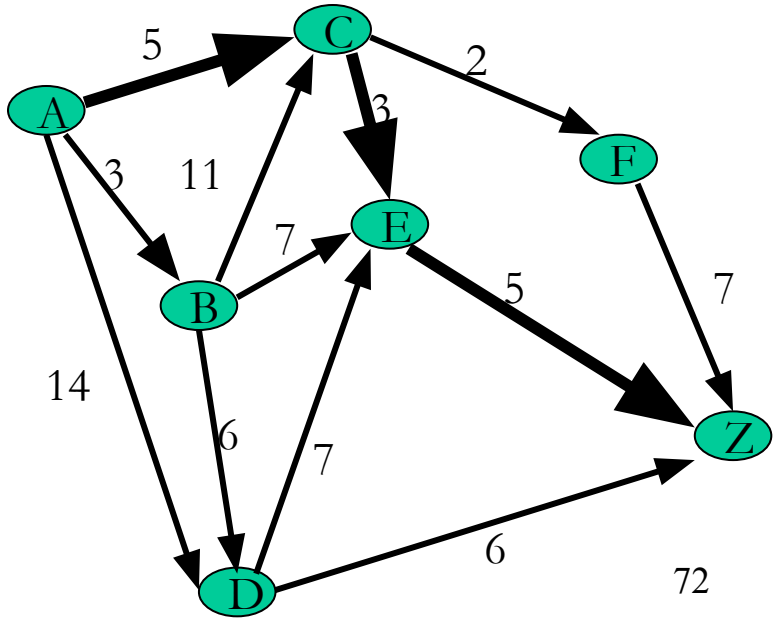
## *Примеры полуколец.*

Первая операция – аналог сложения («целевая операция»), вторая – аналог умножения («соединяющая операция»):

- на числах: {+, x}, {max, +}; {max, min};
- на множествах: {∪, ∩}
- на множествах слов: {∪, •}
- на матрицах: {+, x}.

**+ - «целевая» операция**

*** - «соединительная» операция**

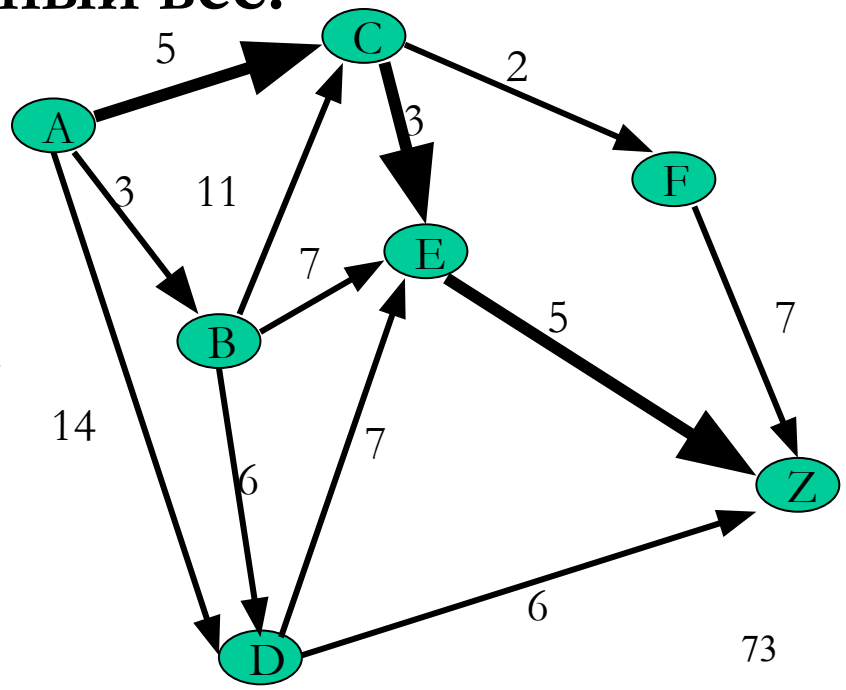


***ДАНО:*** **Ориентированный ациклический граф с весами на ребрах**

**G =< *V, E, W; A, Z*>**

***ЗАДАЧА 1*** **Найти оптимальный полный путь, т.е. полный путь, имеющий минимальный (максимальный) возможный вес.**

***ЗАДАЧА 2*** **Найти сумму мультипликативных весов всех полных путей.**



## Метод динамического программирования (Алгоритм Беллмана)

- Проход от стока к источнику:

  из W есть путь в V =>

  => W обрабатывается позже, чем V.

- Рекуррентное уравнение (минимальный путь)

```
BestW(A) = min{
        W(AB) + BestW(B),
        W(AC) + BestW(C),
        W(AD) + BestW(D)
}
```

- Рекуррентное уравнение (сумма м-весов):

```
Sum(A) =
          W(AB)*Sum(B) +
      +  W(AC)*Sum(C) +
      +  W(AD)*Sum(D)
}
```

***ДАНО:*** **Ориентированный ациклический граф с весами на ребрах**

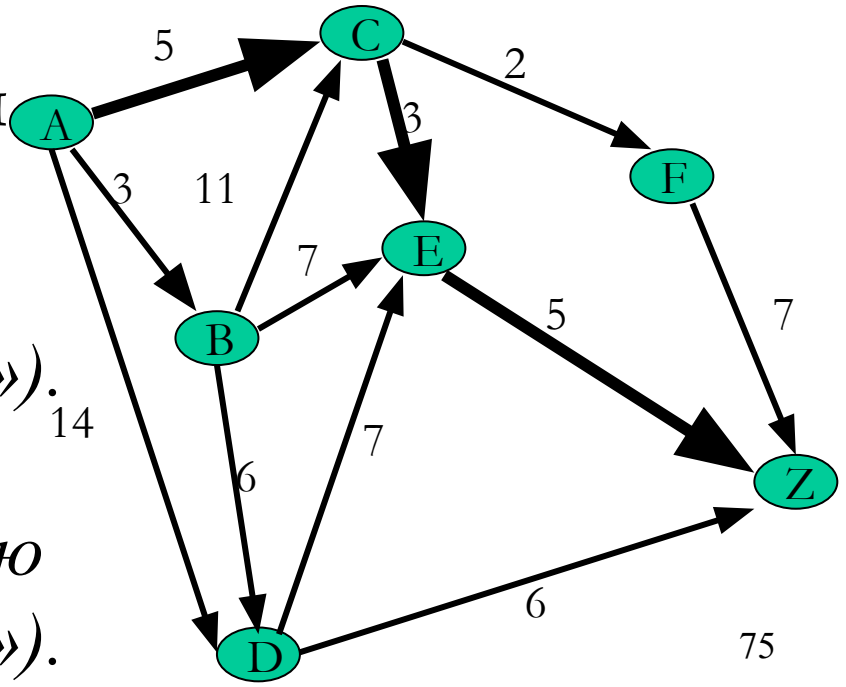
**G =< *V, E, W; A, Z*>;**

***веса W(e) – элементы полукольца K с операциями*** + и *.

***ЗАДАЧА 3*** **Найти сумму мультипликативных весов всех полных путей**

*Операция * («умножение») определяет веса путей («соединительная операция»).*

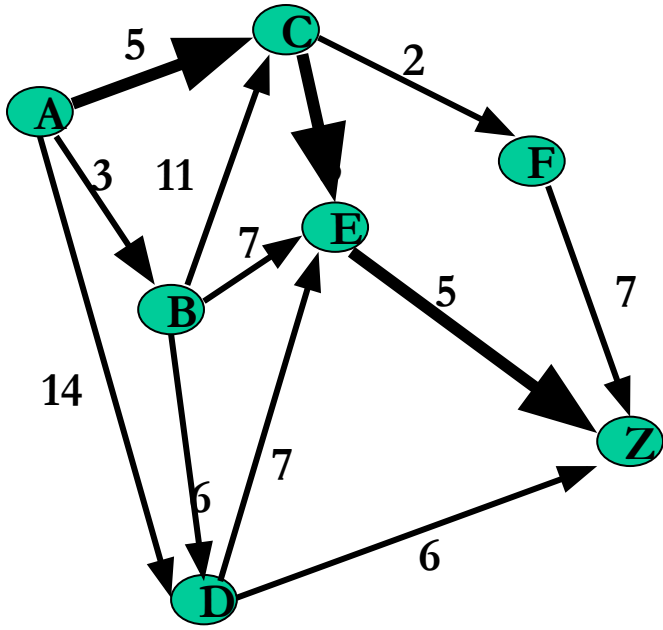
*Операция + («сложение») определяет целевую функцию («соединительная операция»).*



***ДАНО:*** **Ориентированный ациклический граф с весами на ребрах**

$G =< V, E, W; A, Z >;$

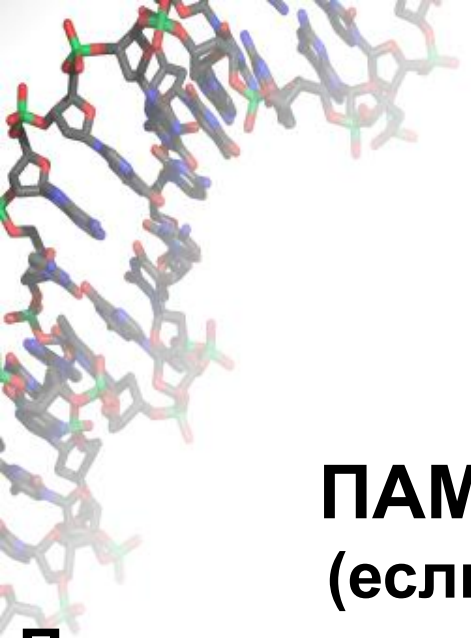
***веса W(e) – элементы полукольца K с операциями*** + и *.




***ЗАДАЧА 3*** **Найти сумму мультипликативных весов всех полных путей.**

**ВРЕМЯ РАБОТЫ ~ к-во РЕБЕР**

**ПАМЯТЬ ~ к-во ВЕРШИН**



## Замечание 1.

# Память

**ВРЕМЯ РАБОТЫ ~ к-во РЕБЕР**
**ПАМЯТЬ ~ к-во ВЕРШИН**

## ПАМЯТЬ МОЖЕТ БЫТЬ МЕНЬШЕ !

**(если в графе можно выделить «слои»)**

**Пример: нахождение веса оптимального выравнивания (но не самого выравнивания !)**

***Space ~ L1 = SQRT(|Vertex|)***

**!! Выравнивание тоже можно найти с памятью *Space ~ L1* и временем *Time ~ L1*L2*, но для этого нужны новые идеи**

[Hirschberg D.S. Algorithms for the Longest Common Subsequence Problem. // Journal of the ACM . 1977. Vol. 24 , N.4. P. 664 – 675. ]

## Замечание 2.
## Различие между *min* и суммой: argmin

Рекуррентное уравнение (минимальный путь)

**BestW(V) = min{**
**W(VB) + BestW(B),**
**W(VC) + BestW(C),**
**W(VD) + BestW(D) }**

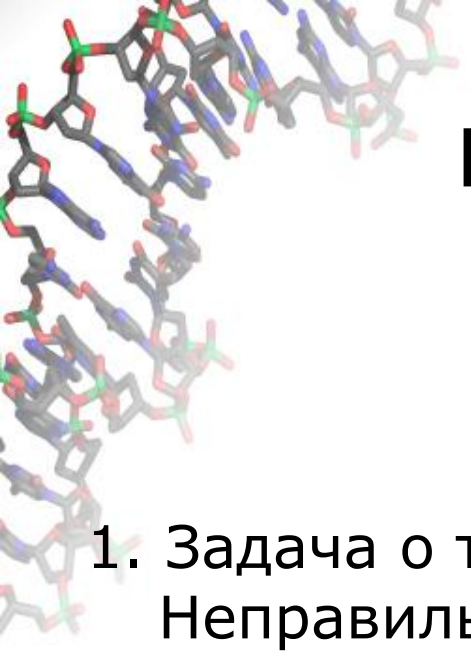
Рекуррентное уравнение (сумма Больцмана)

**Sum(V) = Σ{**
**W(VB) * BestW(B),**
**W(VC) * BestW(C),**
**W(VD) * BestW(D) }**

Операция *min* предполагает не только получение числа, но и (неявно) выбор одного из операндов. Поэтому при работе с *min* мы **кроме значения *веса «оптимального» пути***

**находим и *сам оптимальный путь*.**

Для этого при вычислении значения **BestW(V) = min{…}** мы запоминаем дополнительно **argmin{…}** – наследника (-ков) вершины *V*, на котором (-рых) минимум достигается.
Примеры были раньше.

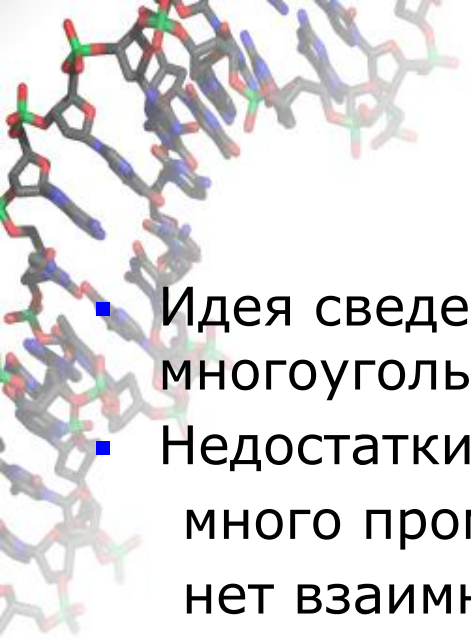


# Раздел 3
# Гиперграфы: знакомство

**Пока без слайдов** ☹

## Развернутый план

1. Задача о триангуляции выпуклого треугольника. Неправильное решение. Сведение задачи к **нескольким** подзадачам меньшего размера. Невозможность моделирования этого с помощью задач на ориентированных графах.
- 2. Понятие гиперграфа. Гиперребро. Гиперпуть. Вес гиперребра. Вес гиперпуть.
- 3. Задача Больцмана для гиперграфов. Рекурсия и алгоритм решения. Понятие ранга вершины для гиперграфов.



# 3.1. Задача о триангуляции (рисунок на доске)

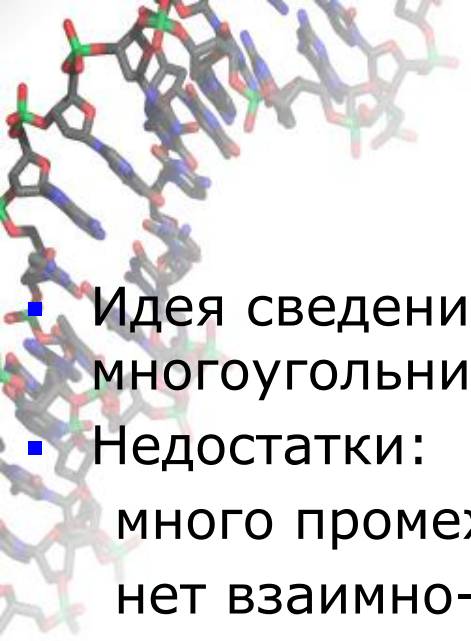
- Идея сведения: провести диагональ, разбить на два многоугольника меньшего размера
- Недостатки:

  много промежуточных задач

  нет взаимно-однозначного соответствия между структурами и последовательностью сведений

**!!!! Сведения образуют не последовательность, а дерево!!!**

- ☐ **НЕ СВОДИТСЯ К ЗАДАЧЕ НА ГРАФЕ !!!**



# Задача о триангуляции
## (рисунок на доске)

- Идея сведения: провести диагональ, разбить на два многоугольника меньшего размера
- Недостатки:

  много промежуточных задач

  нет взаимно-однозначного соответствия между структурами и последовательностью сведений

**!!!! Сведения образуют не последовательность, а дерево!!!**

- ☐ **НЕ СВОДИТСЯ К ЗАДАЧЕ НА ориентированном ГРАФЕ**

- **Сводится к задаче на ориентированном ГИПЕРГРАФЕ!!**

# Задача о триангуляции
## (рисунок на доске)

- Дан выпуклый многоугольник. Каждой диагонали приписан вес – положительное число.
- Триангуляция – это разбиение многоугольника на треугольники непересекающимися диагоналями.
- Вес триангуляции – сумма весов входящих в нее диагоналей.
- Требуется: найти триангуляцию минимального веса.
- Идея: использовать метод динамического программирования (сведение к более простым задачам того же типа).

# 3.2. Понятие гиперграфа

Определение 1. *Граф G* – это пара $<\mathcal{V},\mathcal{C}>$, где $\mathcal{V}$ – это множество вершин, $\mathcal{C}$ – множество ребер .

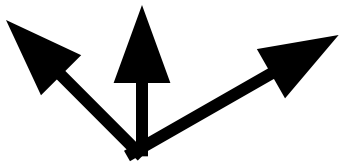
*Ребро* – это пара *<V, W>*, где *V* – начальная вершина ребра, *W*- конечная вершина ребра

*V* *W* →

Определение 2. ***Гиперграф*** *Y* – это пара $<\mathcal{V},\mathcal{H}>$, где $\mathcal{V}$ – это множество вершин, $\mathcal{H}$ – множество гиперребер.

***Гиперребро*** – это пара *<V, <W1, ..., Wk>*, где *V* – начальная вершина ребра, *<W1, ..., Wk>*, - упорядоченный набор конечных вершин гиперребра

*W1 W2 W3*



V

# ❓3.2. Понятие гиперграфа

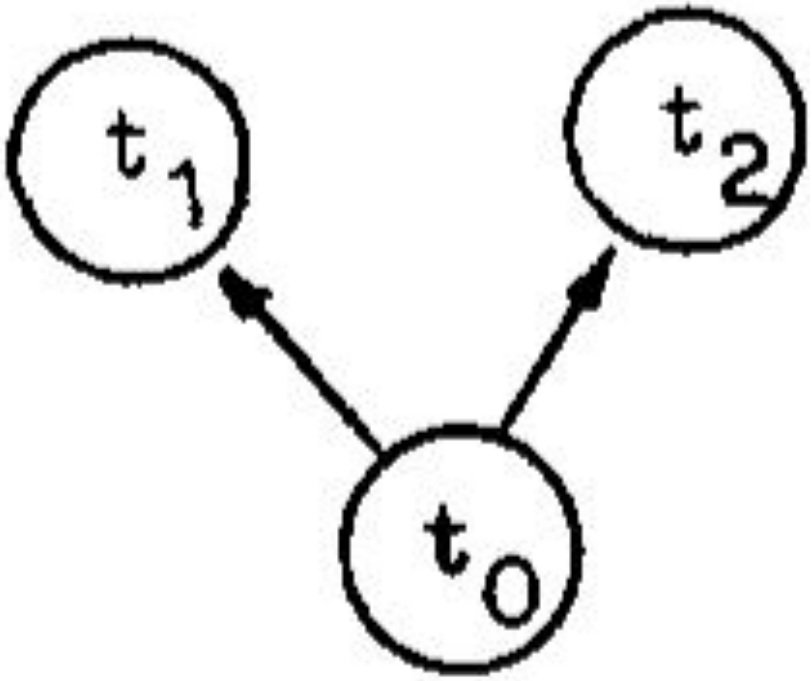
Определение 3. *Путь* в графе $G=<\mathcal{V},\mathcal{C}>$ – это <u>простая цепь</u>, узлы которой помечены вершинами графа *G,* такая что ….

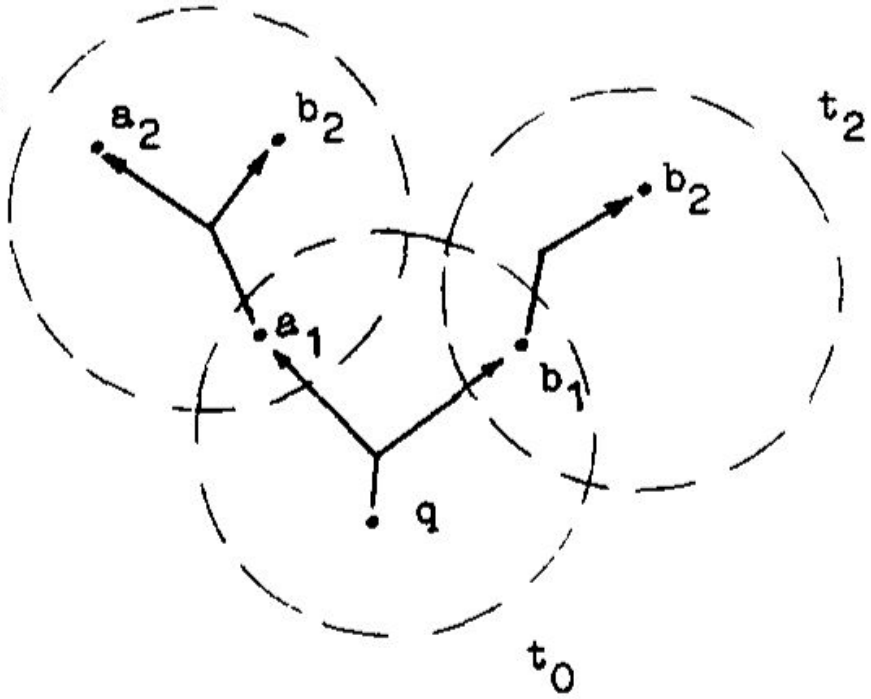
*Начальная вершина* пути – это вершина, которой помечена первый узел цепи, *конечная вершина* – вершина, которой помечен последний узел цепи.

Определение 4. *Гиперпуть* в гиперграфе $Y=<\mathcal{V},\mathcal{H}>,$ > – это <u>упорядоченное дерево</u>, узлы которой помечены вершинами графа *G,* такое что ….

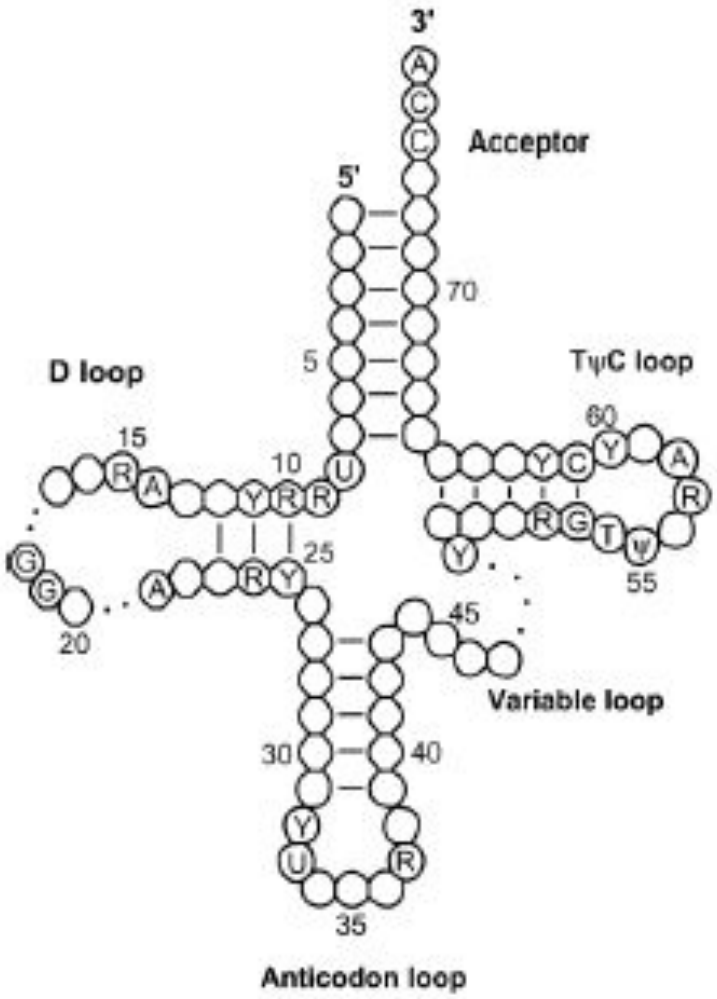
*Начальная вершина* пути – это вершина, которой помечен корень дерева, *конечн**ые** вершин**ы*** – это вершины, которыми помечены листья дерева.

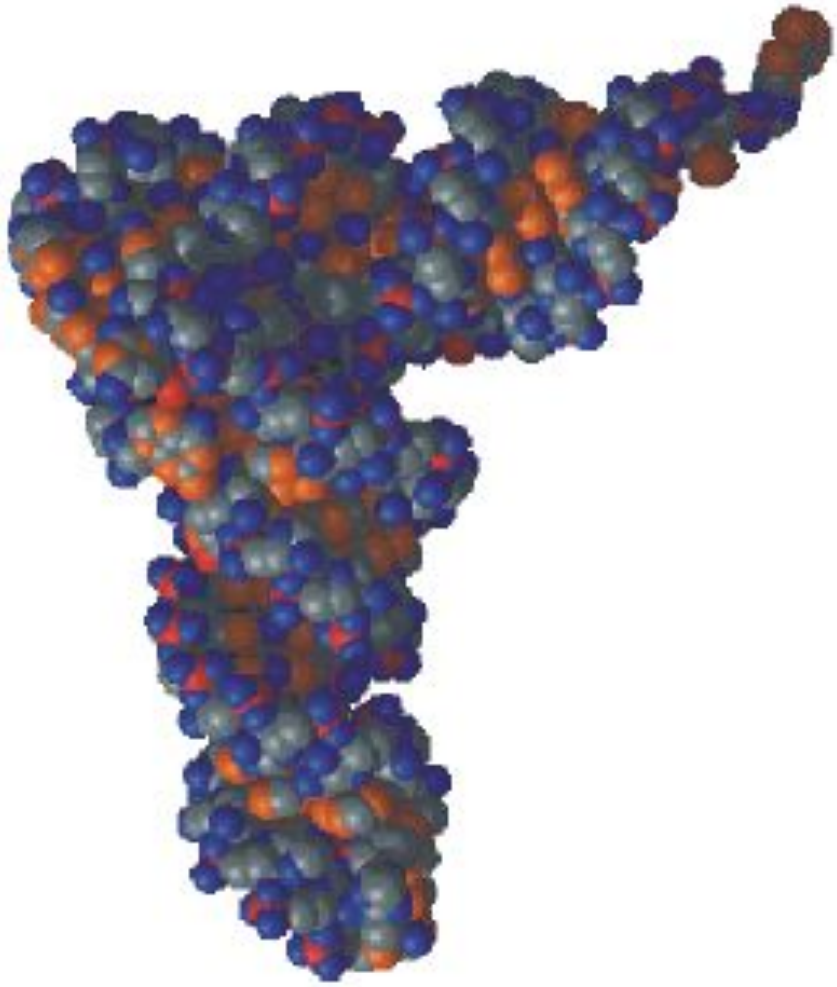
# Гиперпуть



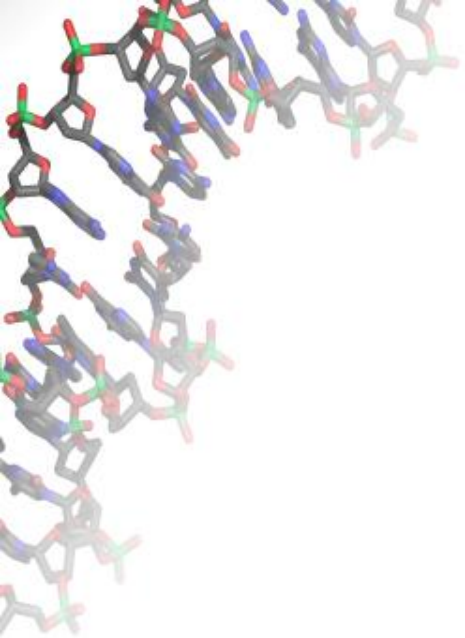



*Вторичная структура РНК.*



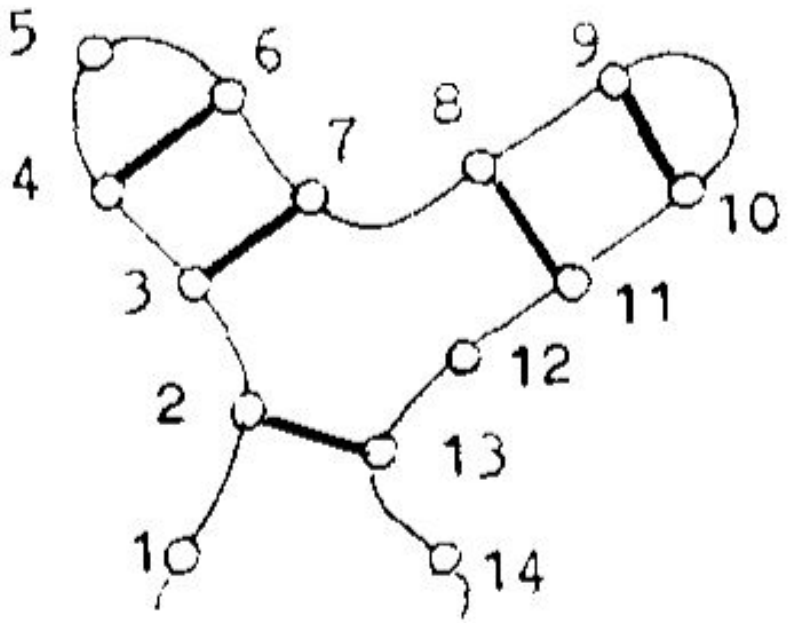



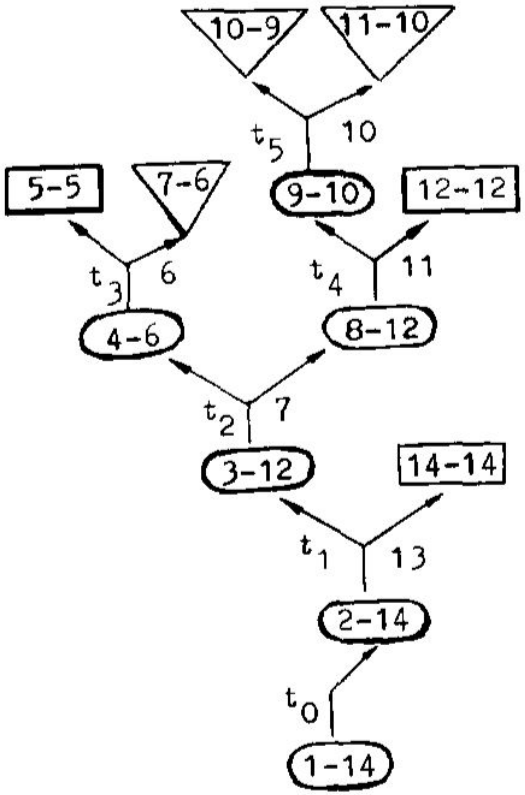
From Paul Higgs

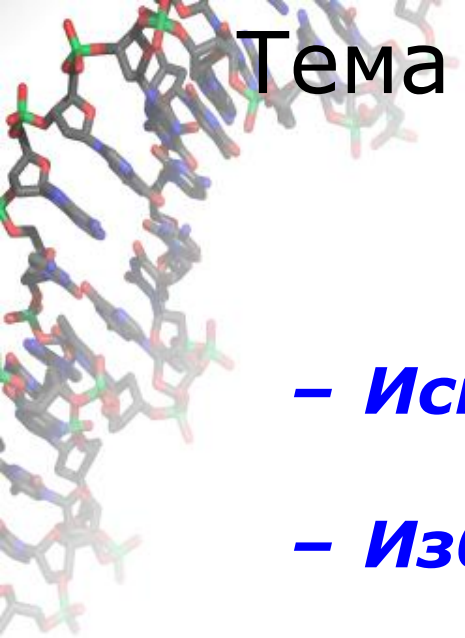


*3. Выравнивание последовательностей РНК с заданной вторичной структурой.*

*Пример: РНК и гиперпуть*



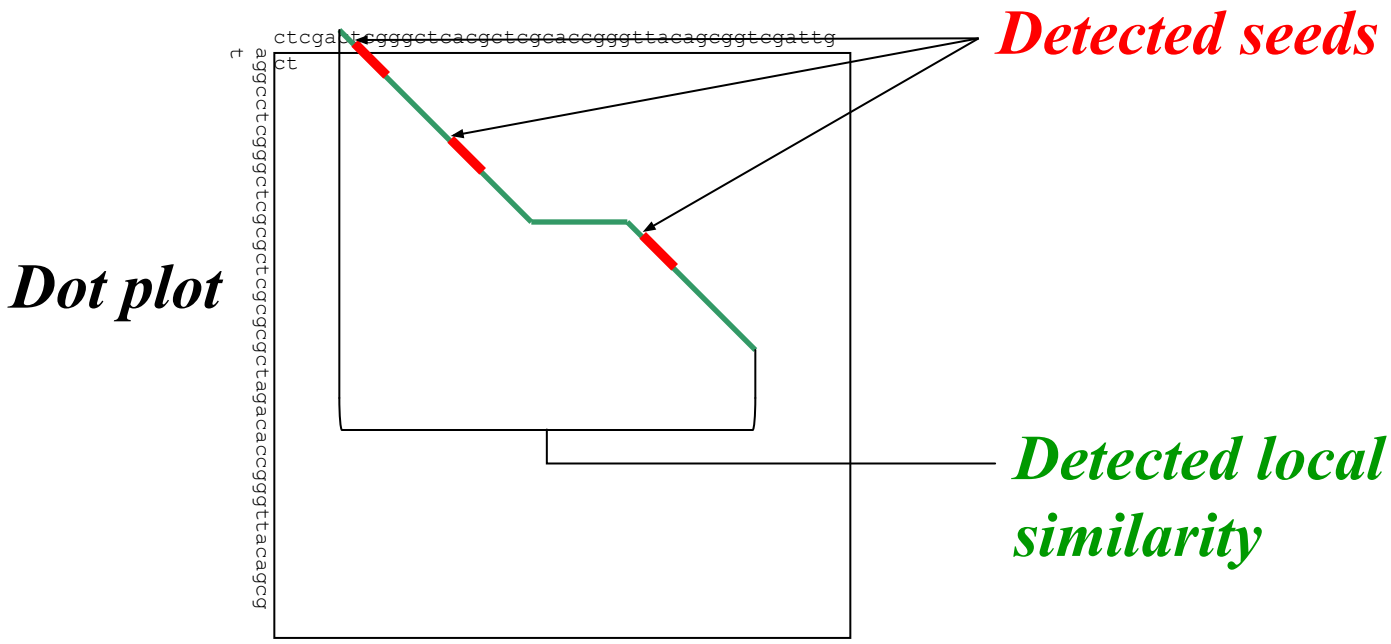


# Тема 4. Поиск локальных сходств

- ***Использование затравок (seed)***
- ***Избирательность и чувствительность***
- ***Типы затравок (seed model)***

# Затравки: фильтрация пространства поиска

- Сначала ищем небольшие и легко диагностируемые участки сходства («затравочные сходства», *seed similarities*).
- Далее ищем сходства только в окрестностях затравочных сходств (одного или нескольких).



# «Классическая затравка» (пример: 6 совпадений подряд)

- Точные совпадения :

```
ATCAGT
||||||
ATCAGT
```

Затравка («затравочное слово», описание затравочных сходств) : ######

***Вес* : 6** [количество #]

- **Пример : 16 совпадений из 20**

```
      ######
ATCAGTGCAATGCTCATGAA
|||.|.|||||||:||.|||
ATCGGCGCAATGCGCAAGAA
```

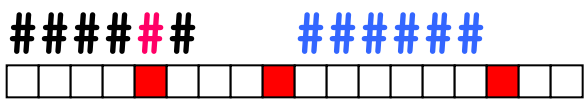
# Затравка ловит сходство
## (затавка соответствует сходству)

- Затравка ##### □ *seed*

Затравочное сходство (… выравнивание)

ATGCAA

ATGCAA




Затравка соответствует[1] сходству[10] в позиции 10

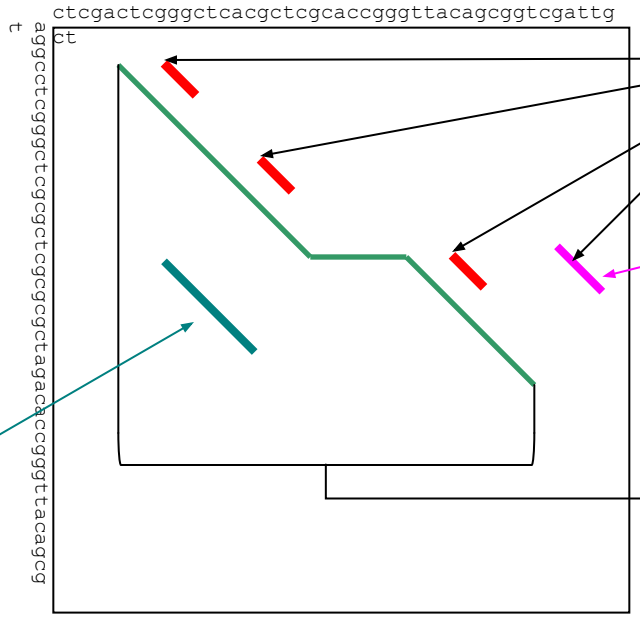
Затравка не соответствует сходству в позиции 1

Затравка ловит сходство

# Недостатки подхода

```
##☹### [16 of 20!]
ATCAGTGCGATGCTCATGAA
|||.||||:|||:||.|||
ATCGGTGCGGTGCGCAAGAA
```

```
######
TCAGTGCAATGCTCATGAA
:|::::||||||:::..::
CCGACACAATGCGTGACCC
```

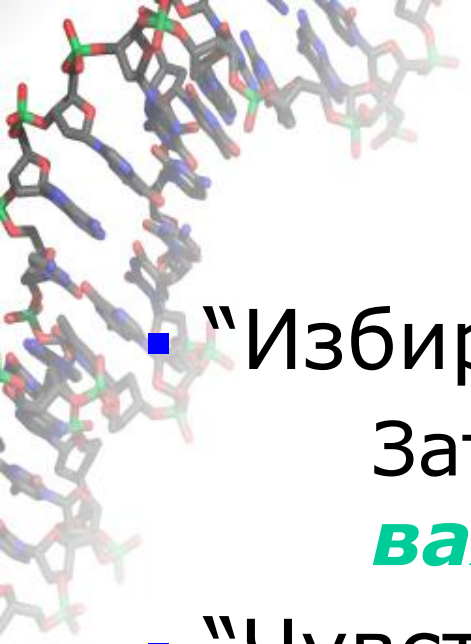



***Найденные затравки***

**Случайное сходство**

***Detected local similarity***

**Пропущенное сходство: не содержит затравок**



# Две проблемы

- “Избирательность”

  Затравка может НЕ быть частью ***важного (для нас) сходства***

- “Чувствительность”

  ***Важное (для нас) сходство*** *может НЕ содержать ни одной затравки*

**Нужно уточнить:**

- **Что такое «важное сходство»?**

# Что может быть мерой избирательности и чувствительности

- Избирательность затравки: ~ $4^{-weight}$

  вероятность ее обнаружения при сравнении независимых случайных последовательностей

- Чувствительность затравки:

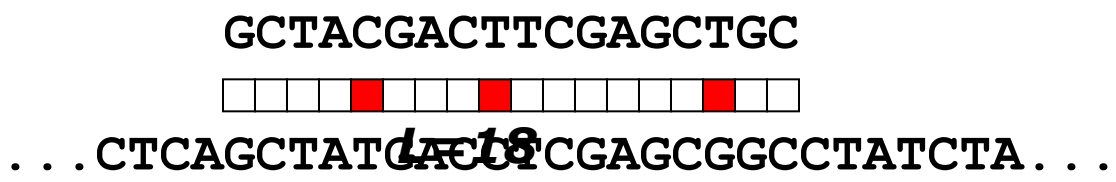
  ***вероятность*** *того, что затравка попадет в* ***важное сходство****.*

**Нужно уточнить:**

- **Что такое «важное сходство»?**
- **Каково распределение вероятностей для важных сходств?**

# Множество важных [целевых] выравниваний и их вероятности

- Выравнивания фиксированной длины без удалений




- Вероятностная модель: ***Бернулли*** ;

  *Случайные выравнивания:* *Prob(match) =0.25*

  *Целевые выравнивания:* *Prob(match) >> 0.25*

*Обобщения: Марковские модели, скрытые марковские модели (сегодня не рассматриваем)*

# Разреженные затравки

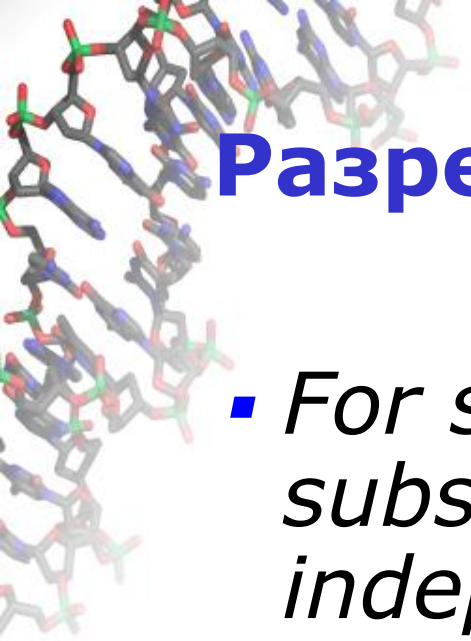
## Ma, Tromp, Li 2002 (PatternHunter)

- Затравка: **###--#-##**
  - ‘**#**’ : должно быть совпадение
  - ‘**-**’ : «джокер» (“все равно, что” )

***Вес* : 6** [количество #]

- **Пример:**

```
###--#-##
ATCAGTGCAATGCTCAAGA
|||||.||.||||:||||||
ATCAGCGCGATGCGCAAGA
```

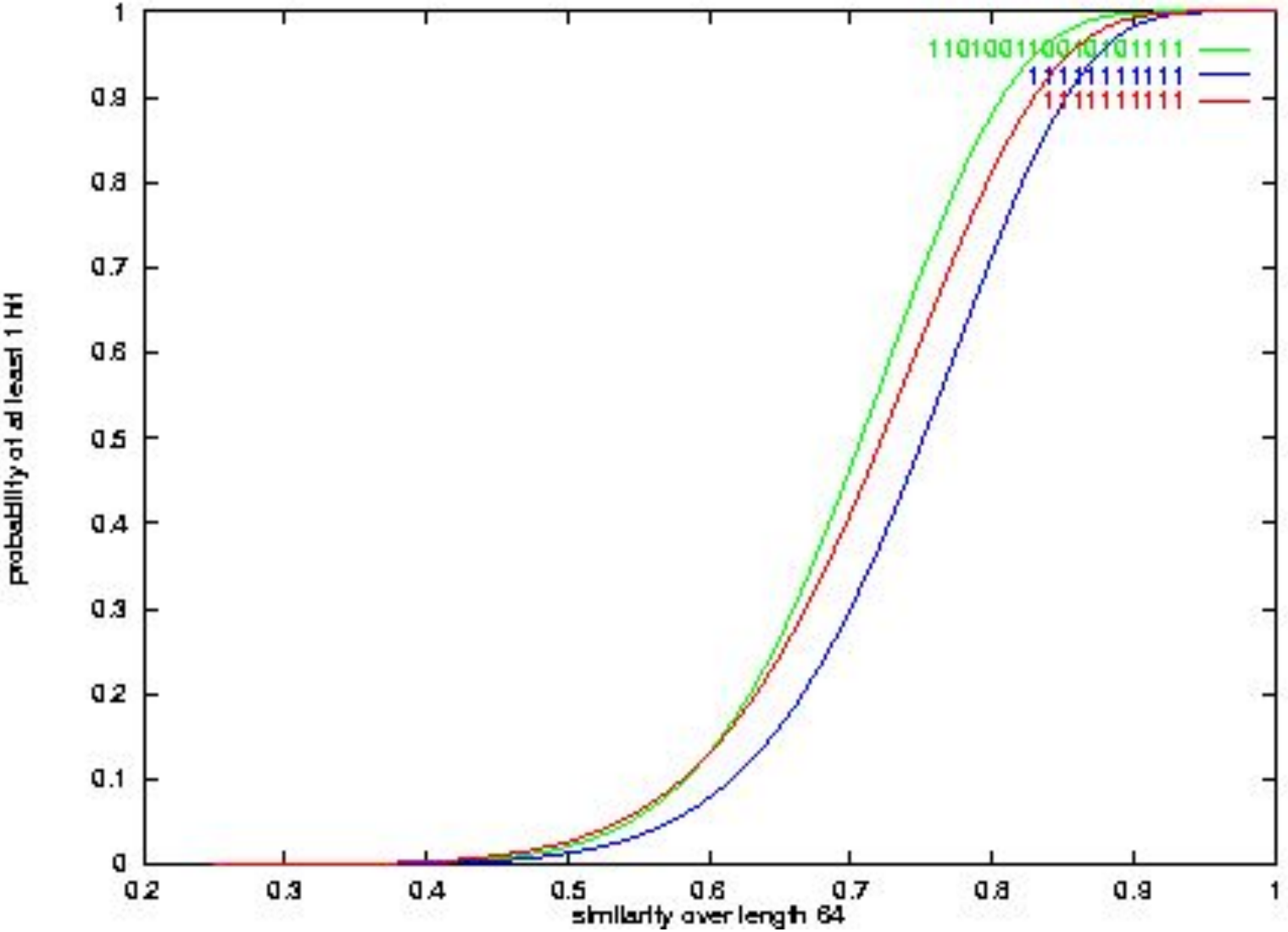


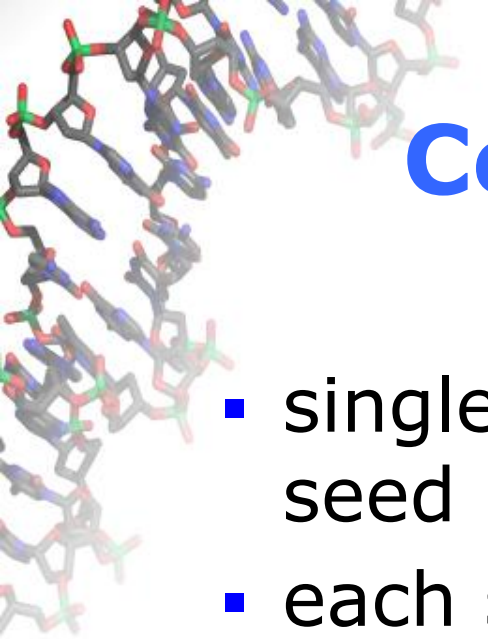
# Разреженные затравки: в чем преимущество?

- *For spaced seeds, hits at subsequent positions are "more independent events"*
- *For contiguous vs. spaced seeds of the same weight, the expected number of hits is (basically) the same but the probabilities of having **at least one hit** are very different*

# Sensitivity: PH weight 11 seed vs BLAST 11 & 10

[after Ma, Tromp and Li]

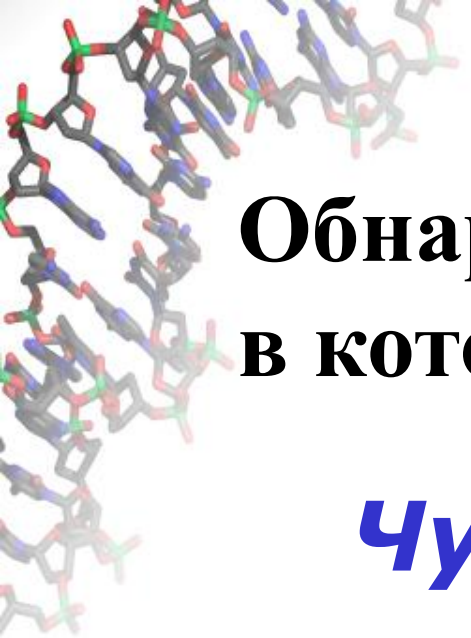




# Семейства затравок

- single filter based on several distinct seed patterns
- each seed pattern detects a part of interesting similarities but together they detect [almost] all of them

- Li, Ma, Kisman, Tromp 2004 (PatternHunter II)
- Kucherov, Noe, Roytberg, 2005
- Sun, Buhler, RECOMB 2004



# Пример: ВСЕ (18,3)

**Обнаружить все сходства длины 18, в которых не более 3 несовпадений**

***Чувствительность = 1.0***

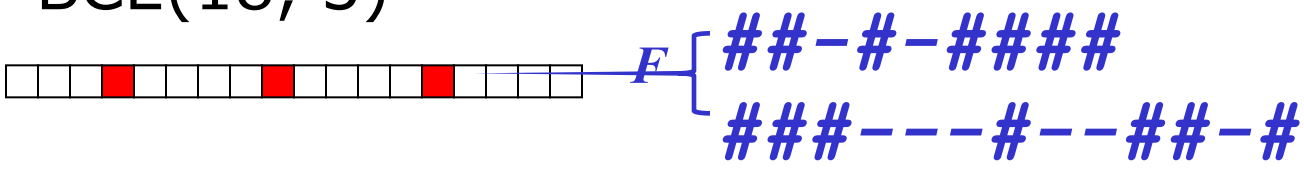
***Избирательность***

*(вероятность случайного появления затравочного сходства)* ***-> MIN***

# Пример: BCE (18,3)

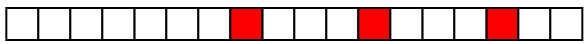
**Обнаружить все сходства длины 18, в которых не более 3 несовпадений**

Множественная затравка *F* решает проблему BCE(18, 3)




Затравка *F* состоит из двух простых затравок, каждая из них имеет вес 7

# Пример: ВСЕ (18.3)



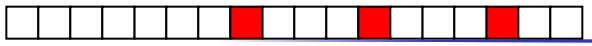
###---#--##-#

###---#--##-#

##-#-####

###---#--##-#

w=7



###-##---#-###

##-##-#####

###-####--##

###-##---#-###

##----####-###

###---#-#-##-##

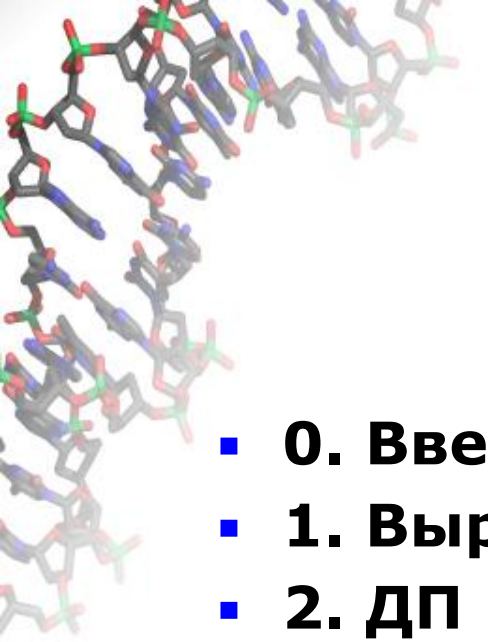
###-#-#-#-----###

w=9

# Пример: BCE (18.3). Избирательности

Избирательность семейства затравок – вероятность встретить хотя бы одну из них в случайном месте ($p(match) = 1/4$)

| | | |
|---|---|---|
| `####` | **w=4** | **~39. $10^{-4}$** |
| `###-##` | **w=5** | **~9.8 $10^{-4}$** |
| `##-#-####`<br>`###---#--##-#` | **w=7** | **~1.2 $10^{-4}$** |
| `##-##-#####`<br>`###-####--##`<br>`###-##---#-###`<br>`##----####-###`<br>`###---#-#-##-##`<br>`###-#-#-#-----###` | **w=9** | **~0.23 $10^{-4}$** |



# *СПАСИБО за ВНИМАНИЕ*

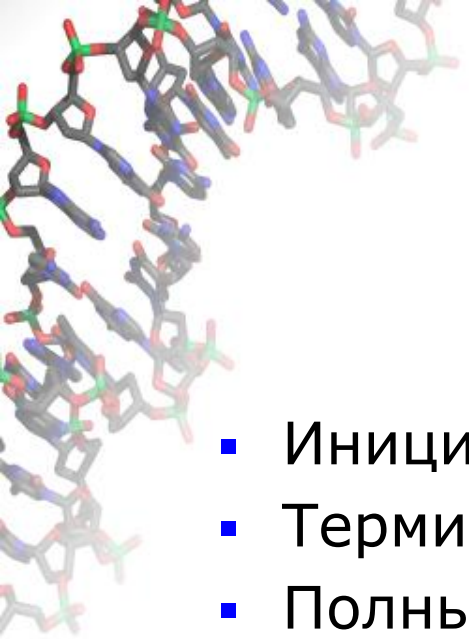
- **0. Введение**
- **1. Выравнивания**
- **2. ДП и алгебра**
- **3. Гипернрафы и РНК**
- **4. Разреженные затравки**

***Чего не было:***

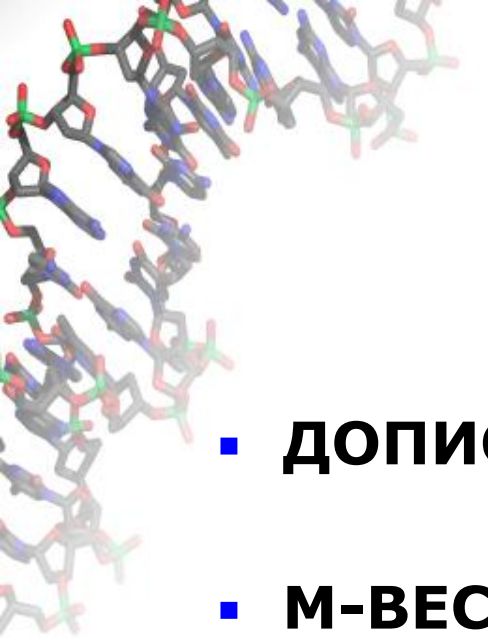
**Сравнительная геномика**

**Разработка лекарств**

**Клеточные автоматы….**

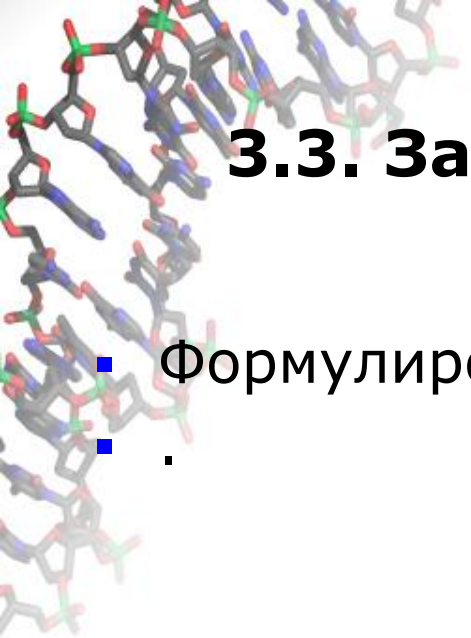


- Инициальный (гипер) путь
- Терминальный (гипер) путь
- Полный (гипер) путь



# Вес гиперпути

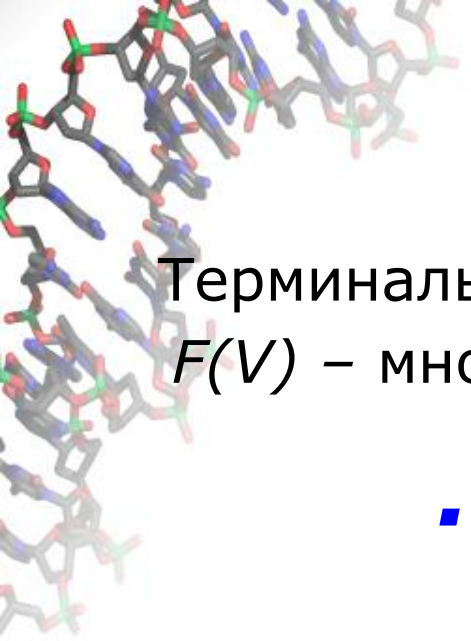
- **ДОПИСАТЬ !!!**
- **М-ВЕС НАД ПОЛУКОЛЬЦОМ**



# 3.3. Задача Больцмана для гиперграфов.

.

- Формулировка задачи Больцмана.
- .



# ❓Подход к решению

Терминальная сумма Больцмана вершины *V:*

*F(V)* – множество всех терминальных гиперпутей с начальной вершиной *V.*

- ***Sum(V) = Σ{M(T)| T∈ F(V) }***

- Идея: Найти терминальные суммы Больцмана для всех вершин. Вершины перебираются в порядке возрастания рангов.
- Уточнить: что такое ранг вершины в гиперграфе (= максимальная высота гиперпути с данной начальной вершиной)
- Пока считаем ранги известными

# Терминальные суммы Больцмана для гиперребер

Терминальная сумма Больцмана гиперребра *y:*

*FF(y)* – множество всех терминальных гиперпутей с начальной вершиной *V.*

- $S(y) = \Sigma\{M(T) \mid T \in Fr(y)\}$

- *Start(V)* – множество всех гиперребер с начальной вершиной *V*.

- Утверждение.
  - $Sum(V) = \Sigma\{S(y) \mid y \in Start(V)\}$

# Терминальные суммы Больцмана для гиперребер: рекурсия

Утверждение.

Пусть *y = <V, <W1, …, Wk>* - гиперребро. Тогда

***S(y) = W(y)*Sum(W1)*…* Sum(Wk)***

Доказательство. Пусть *T ∈ Fr(y),*

*Ti – поддерево T с корнем в узле, соответствующем i-й конечной вершине гиперрбра y* – начального гиперребра дерева T.

Тогда:

1) ***Ti ∈ F(Wi)***

2) существует взаимно-однозначное соответствие между деревьями

***T ∈ Fr(y)*** и **наборами <*T1, …, Tk*>,**

где ***Ti ∈ F(Wi)*, i =1,…, k**

=>

# Терминальные суммы Больцмана для гиперребер: рекурсия

2) существует взаимно-однозначное соответствие между дере-вьями $T \in Fr(y)$ и наборами $<T1, \ldots, Tk>$, где $Ti \in F(Wi)$, i =1,…, k

=>

$$S(y) = \Sigma\{M(T) | T \in Fr(y)\} =$$

$$= \Sigma \ldots \Sigma\{W(y)*M(T1)*\ldots*M(Tk) | T1 \in F(w1, \ldots, F(Wk)\}$$

=

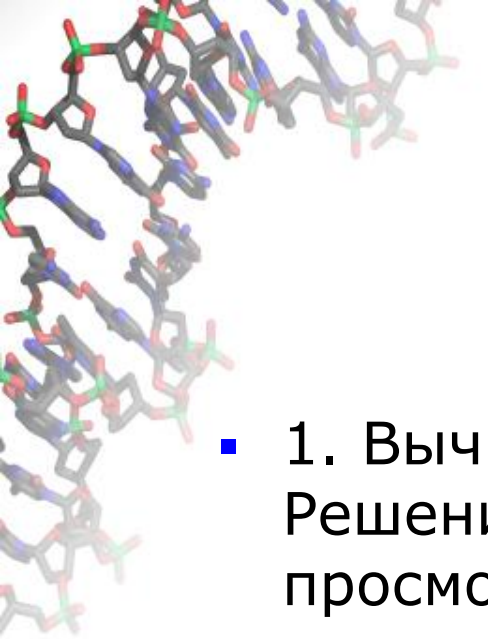
$$= W(y)*\Sigma \ldots \Sigma\{M(T1)*\ldots*M(Tk) | T1 \in F(w1, \ldots, F(Wk)\}$$

=

**[СУММА ПРОИЗВЕДЕНИЙ = ПРОИЗВЕДЕНИЕ СУММ]**

$$= W(y)* \Sigma\{M(T1) | T1 \in F(w1\}* \ldots$$

$$\ldots* \Sigma\{M(T1) | T1 \in F(w1\} =$$



## Осталось:

- 1. Вычисление рангов вершин гиперграфа. Решение задачи Больцмана, когда порядок просмотра вершин гиперграфа неизвестен.
- 2. Вычисление специальных сумм Больцмана.
- 3. Разбор примеров.
- 4. Решение задачи про триангуляцию.