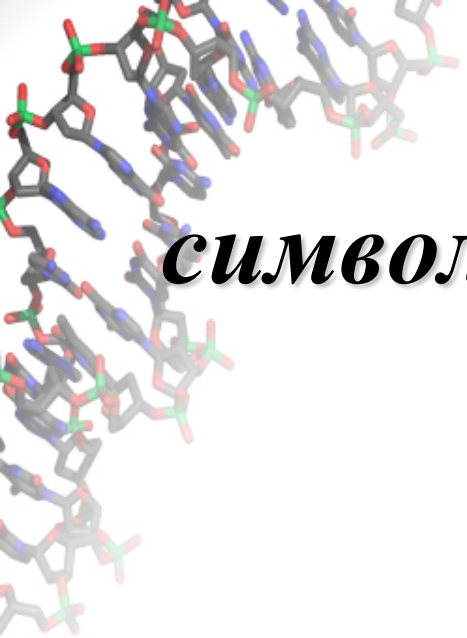


Анализ символьных последовательностей

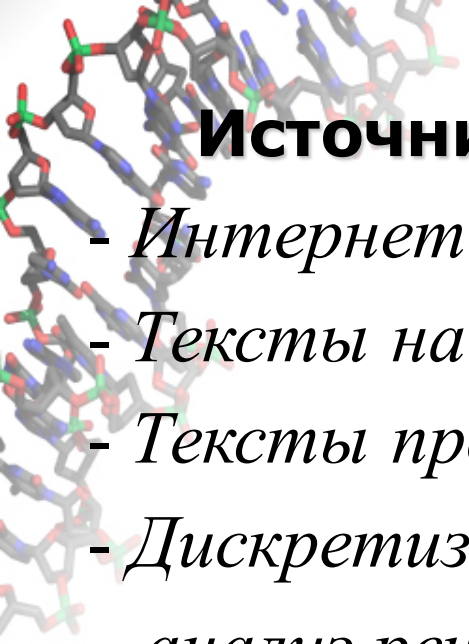
по мотивам биоинформатики

М.А. Ройтберг

Занятие 1

Яндекс

13 февраля 2017

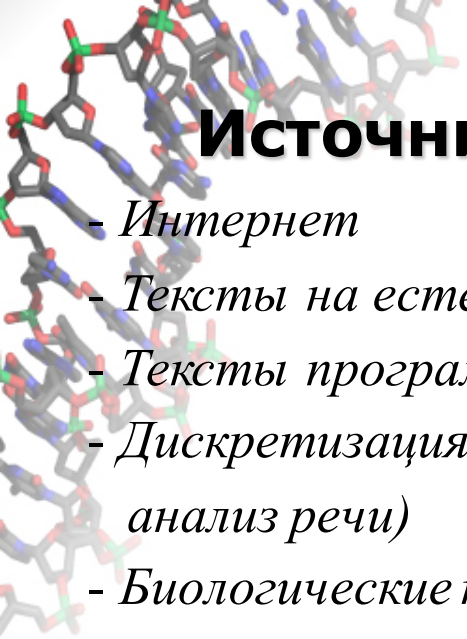


Источники последовательностей (текстов)

- *Интернет*
- *Тексты на естественных языках (лингвистика)*
- *Тексты программ (программирование)*
- *Дискретизация сигналов (технические приложения, анализ речи)*
- *Биологические полимеры – ДНК, РНК, белки (биоинформатика)*

Различия:

- *Длины текстов;*
- *Размеры корпуса текстов в целом;*
- *Размеры исследуемых групп текстов;*
- *Размеры алфавитов*

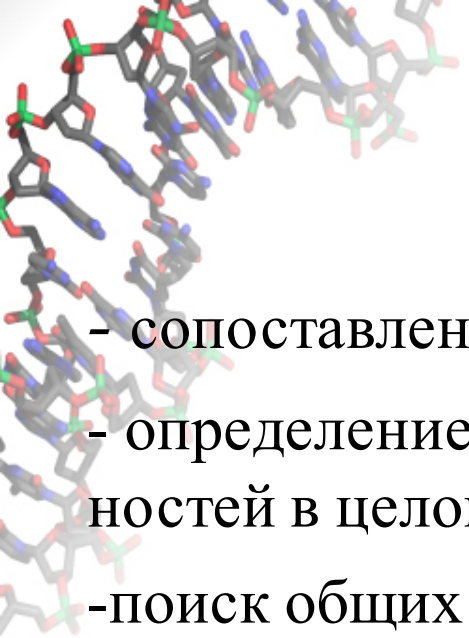


Источники последовательностей (текстов)

- *Интернет*
- *Тексты на естественных языках (лингвистика)*
- *Тексты программ (программирование)*
- *Дискретизация сигналов (технические приложения, анализ речи)*
- *Биологические полимеры – ДНК, РНК, белки (биоинформатика)*

Различия:

- *Длины текстов;*
- *Размеры корпуса текстов в целом;*
- *Размеры исследуемых групп текстов;*
- *Размеры алфавитов*
- ***Постановки задач.***



Основные задачи

- сопоставление в целом (парное, множественное);

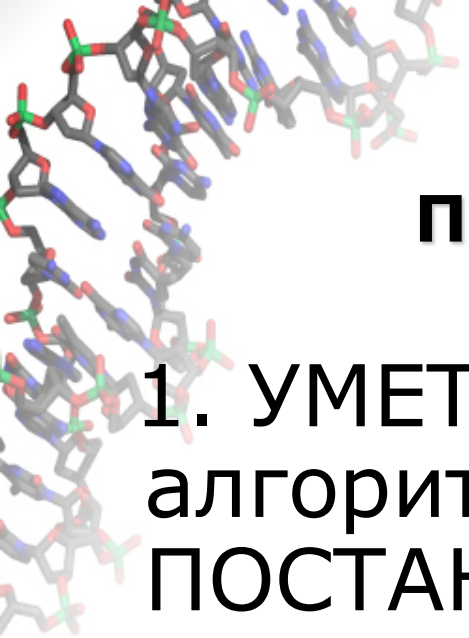
- определение количественной меры сходства последовательностей в целом;

-поиск общих мотивов (в двух и нескольких последовательностях); поиск в базах данных;

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

- разбиение последовательности на «однородные» участки;

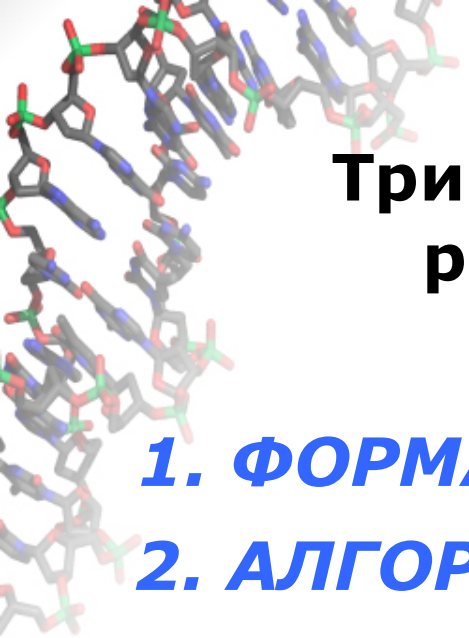
- определение статистической значимости результатов сравнения и поиска.



При анализе данных требуется:

1. УМЕТЬ ПРИДУМЫВАТЬ не только алгоритмы, но и ФОРМАЛЬНЫЕ ПОСТАНОВКИ ЗАДАЧ.

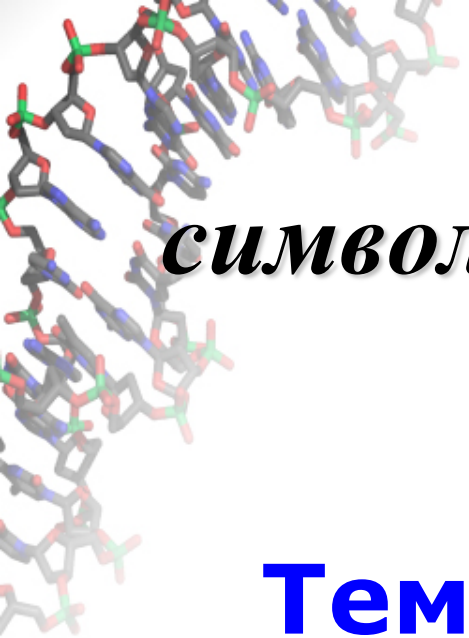
2. ПРОВЕРЯТЬ, насколько ФОРМАЛЬНАЯ ПОСТАНОВКА ЗАДАЧИ соответствует СОДЕРЖАТЕЛЬНОЙ ПОСТАНОВКЕ.



Три вещи, которые нужно уточнить, решая содержательную задачу

1. ***ФОРМАЛЬНАЯ ПОСТАНОВКА ЗАДАЧИ.***
2. ***АЛГОРИТМ РЕШЕНИЯ***
3. ***ПРОВЕРКА – АДЕКВАТНО ЛИ ТО, ЧТО ПОЛУЧАЕТСЯ (ИСХОДНОЙ СОДЕРЖАТЕЛЬНОЙ ЗАДАЧЕ).***

Как правило, уточнение этих вещей происходит взаимосвязано и итеративно

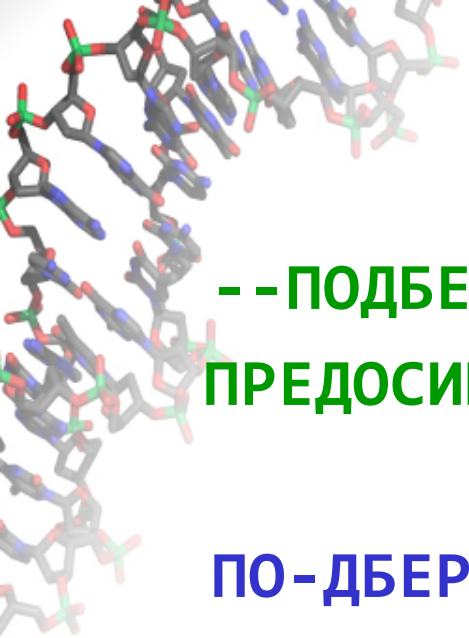


Анализ символьных последовательностей

по мотивам биоинформатики

Тема 0 (вводный пример)

Точность и достоверность выравниваний.



Варианты выравниваний

--ПОДБЕРЕЗОВИК
ПРЕДОСИНОВИЧКИ

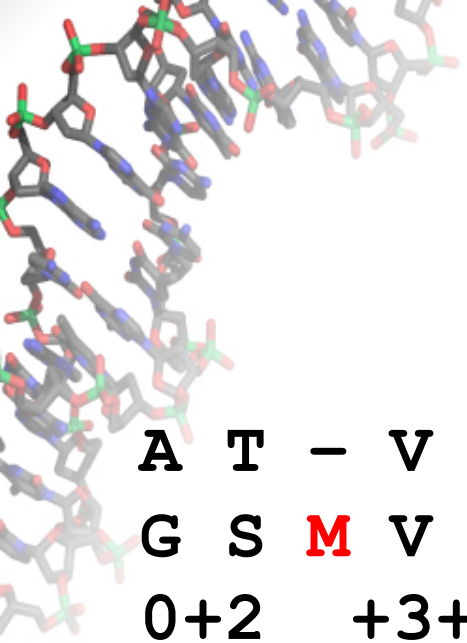
ПОДБЕРЕЗОВИК--
ПРЕДОСИНОВИЧКИ

ПО-ДБЕРЕЗОВИК--
ПРЕДОСИН-ОВИЧКИ

П-ОДБЕРЕЗОВИК--
ПРЕД-ОСИНОВИЧКИ

П-ОДБЕРЕЗОВИ-К-
ПРЕД-ОСИНОВИЧКИ

Какой вариант выбрать?



Формализация:

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
```

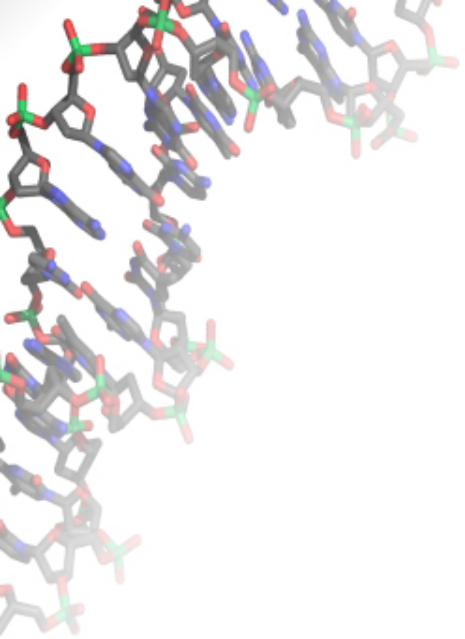
Штрафы за удаление:

фрагмента = 0;

символа = -1

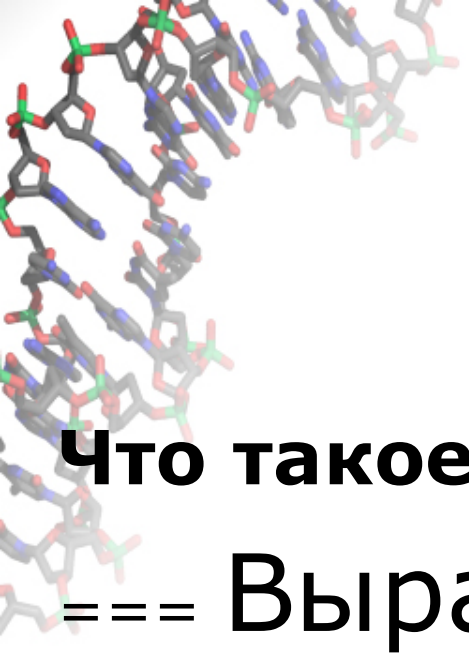
Score = Σ m(i,j)-GapPen = **21 - 3 =** **18**

Строим оптимальное выравнивание



Правильно ли оптимальное выравнивание?

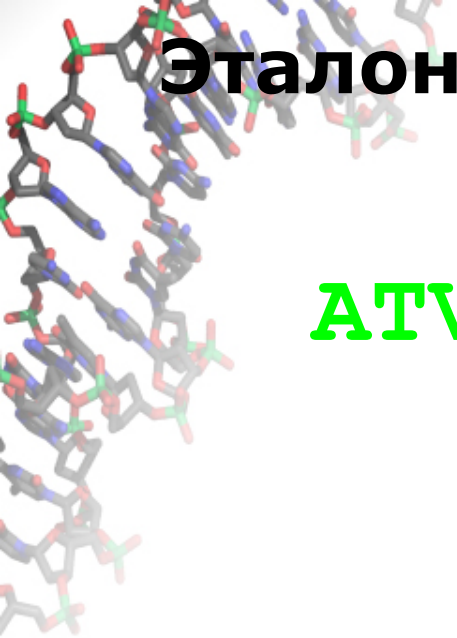
Что такое «правильное выравнивание»?



Правильно ли оптимальное выравнивание?

Что такое «правильное выравнивание»?

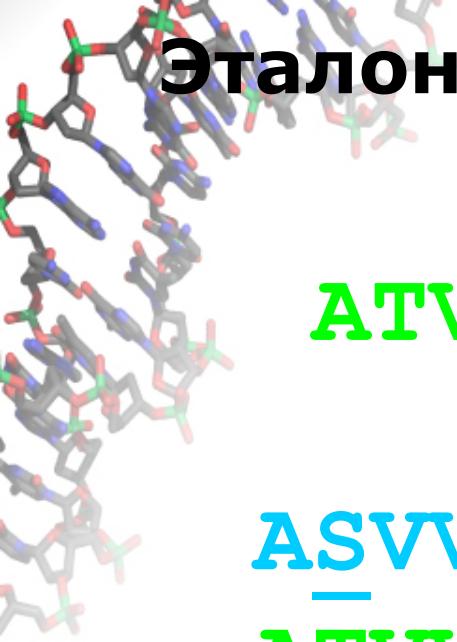
=== Выравнивание, отражающее **эволюционный процесс** (не обязательно биологический).



Эталон: эволюционное выравнивание

ASVVLDFTGT

ATVVITGS GSMVLLEFSGT



Эталон: эволюционное выравнивание

ASVVLDFTGT

ATVVITGS GSMVLLEFSGT

ASVVLDFTGT AS-VVLDFTGT

ATVVI--TGS GSMVLLEFSGT

Эталон: эволюционное выравнивание

ASVVLDFTGT

ATVVITGS GSMVLLEFSGT

ASVVLDFTGT AS-VVLDFTGT

ATVVI--TGS GSMVLLEFSGT

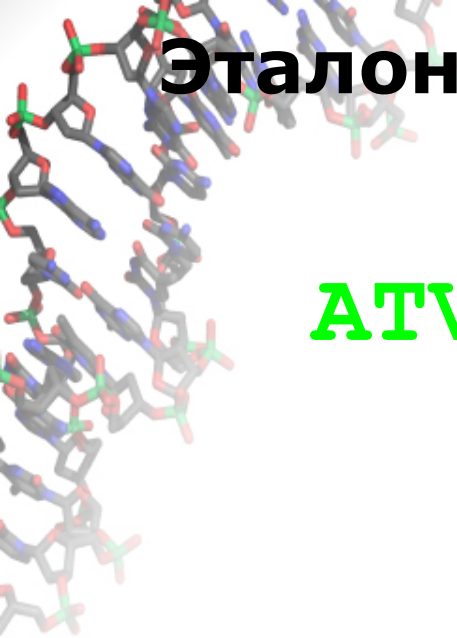
AS-VVLDFTGT AS-VVLDFTGT

AT-VVI--TGS GSMVLLEFSGT

AS-VVLDFTGT

AT-VVI--TGS

GSMVLLEFSGT



Эталон: эволюционное выравнивание

ASVVLDFTGT

ATVVITGS GSMVLLEFSGT

AT-VVI—-TGS

GSMVLLEFSGT

Варианты выравниваний

--ПОДБЕРЕЗОВИК
ПРЕДОСИНОВИЧКИ

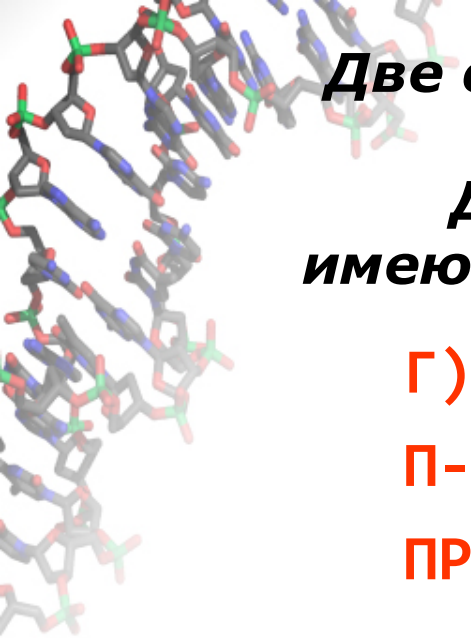
ПОДБЕРЕЗОВИК--
ПРЕДОСИНОВИЧКИ

ПО-ДБЕРЕЗОВИК--
ПРЕДОСИН-ОВИЧКИ

П-ОДБЕРЕЗОВИК--
ПРЕД-ОСИНОВИЧКИ

П-ОДБЕРЕЗОВИ-К-
ПРЕД-ОСИНОВИЧКИ

Какой вариант выбрать?



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

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

Г)	Д)
П-ОДБЕРЕЗОВИК--	П-ОДБЕРЕЗОВИ-К-
ПРЕД-ОСИНОВИЧКИ	ПРЕД-ОСИНОВИЧКИ

При любом штрафе за удаление символа

Д) лучше, чем Г)

(удаление – посимвольное)

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

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

Г)

П-ОДБЕРЕЗОВИК--

ПРЕД-ОСИНОВИЧКИ

Д)

П-ОДБЕРЕЗОВИ-К-

ПРЕД-ОСИНОВИЧКИ

НО:

Д) – «неправильно» («эволюционно»)

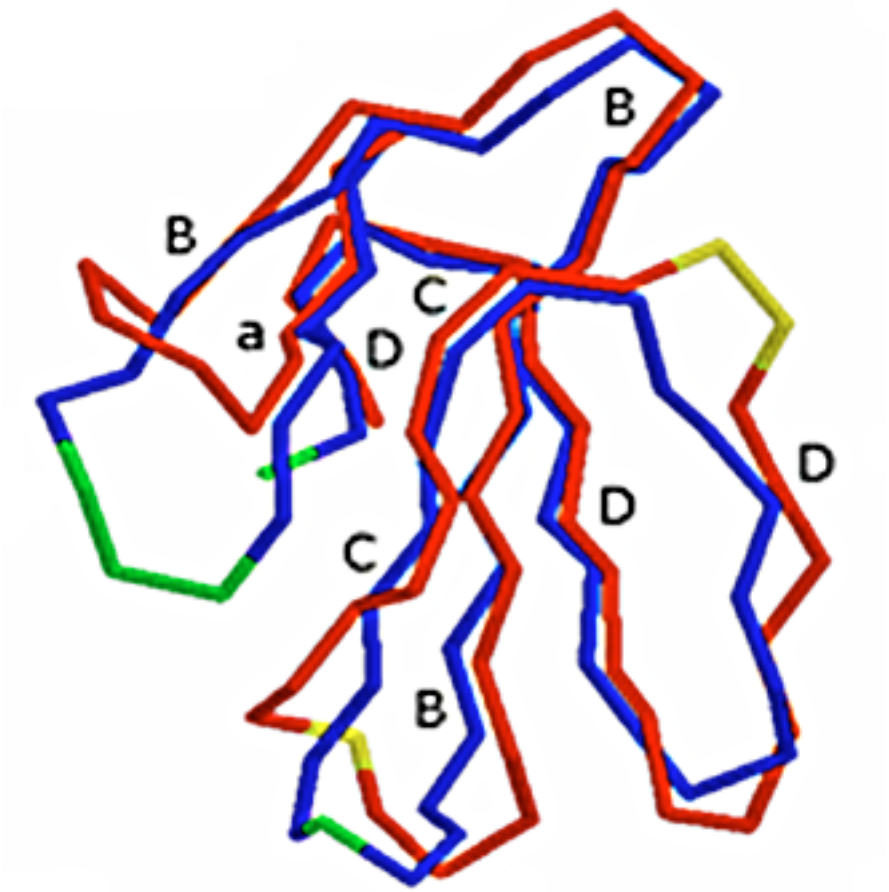
ПОД -БЕРЕЗ-ОВИ-К

ПРЕД-ОСИН -ОВИ-К

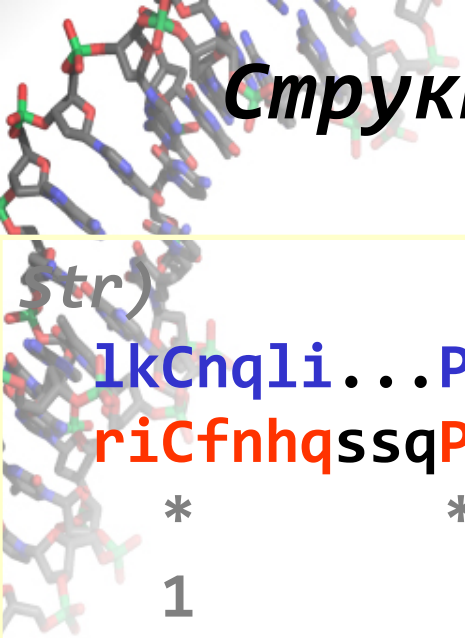
ПРЕД-ОСИН -ОВИ-Ч-ЕК

ПРЕД-ОСИН -ОВИ-Ч- КИ

Приближение: структурные выравнивания



Структурное и алгоритмическое выравнивания



str)

40 сопоставлений

lkCnqli...PPFWKTCPKGKNLCYKmtmraapmvPVKRGCidv

riCfnhqssqPQTTKTCSPGESSCYHkqwsdfrgtIIERGCg..

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

1 16 6

AlgSW)

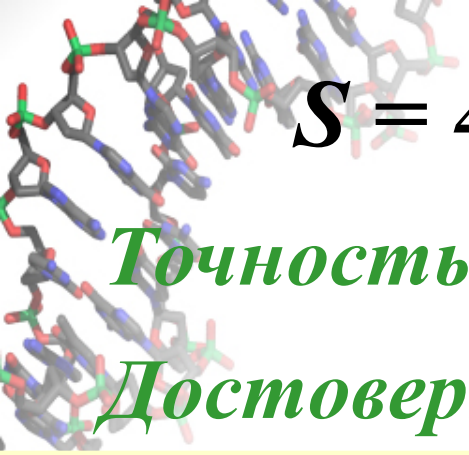
1 16 6

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

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

..riCfnhqssqPQTTKTCSPGESSCYHkqwsdfrgt...IIERGC..g

35 сопоставлений

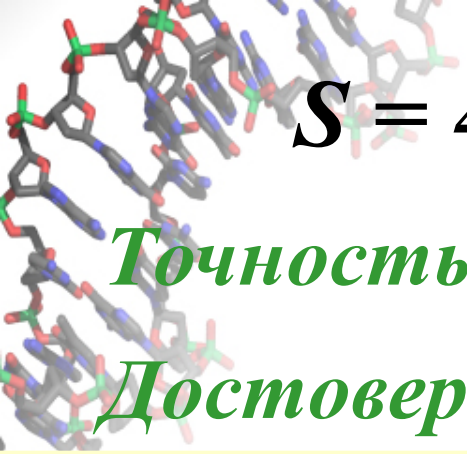


$S = 40$ $I = 23$ $A = 35$

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
```



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

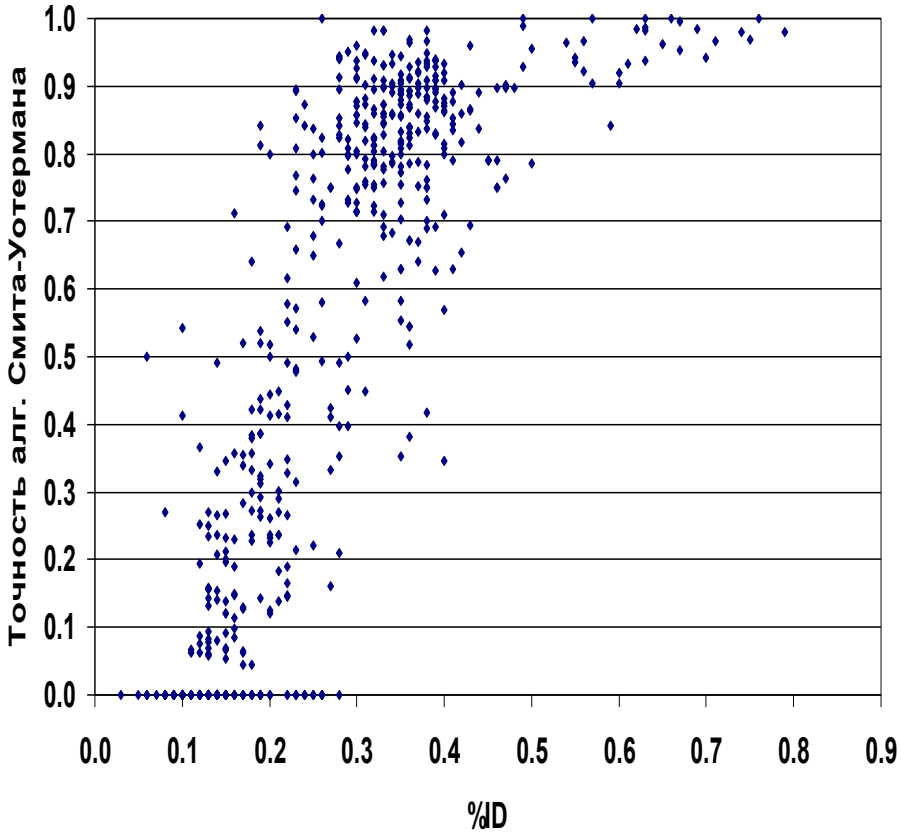
Точность $Acc = I/S = 23/40 = 0.58$

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

```
Str)                                                40 сопоставлений
lkCnqli...PpfwKTCpkGknlCYkmtmraapmvpvkRGCidv
riCfnhqssqPqttKTCspGessCYhkqwsdfrgtiieRGCg..
  *        ****************                ******
  1              16                         6
                                           10 совпадений
```

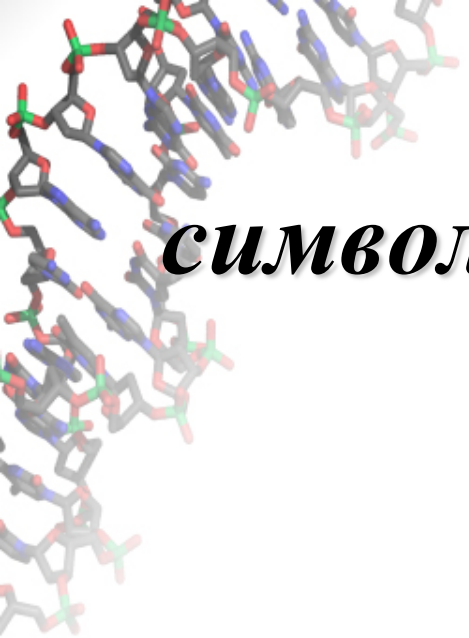
$S = 40$ $M = 10$

$\%Id = M/S = 0.25$



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

%ID	SW точность (acc)
< 0,1	0,037
0,1-0,3	0,306
0,3-0,4	0,818
>0,4	0,893



Анализ символьных последовательностей

по мотивам биоинформатики

Тема 1

Пути в ориентированных ациклических графах (повторение и не только)

1.1. Повторение

Эпиграф. Задача.

$$
\begin{aligned}
S = & \\
= & \; a_1 \cdot b_1 + a_1 \cdot b_2 + \ldots + a_1 \cdot b_{1000} \\
+ & \\
+ & \; a_2 \cdot b_1 + a_2 \cdot b_2 + \ldots + a_2 \cdot b_{1000} \\
+ & \\
+ & \qquad\qquad\qquad \ldots \\
+ & \\
+ & \; a_{1000} \cdot b_1 + a_{1000} \cdot b_2 + \ldots + a_{1000} \cdot b_{1000}
\end{aligned}
$$

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

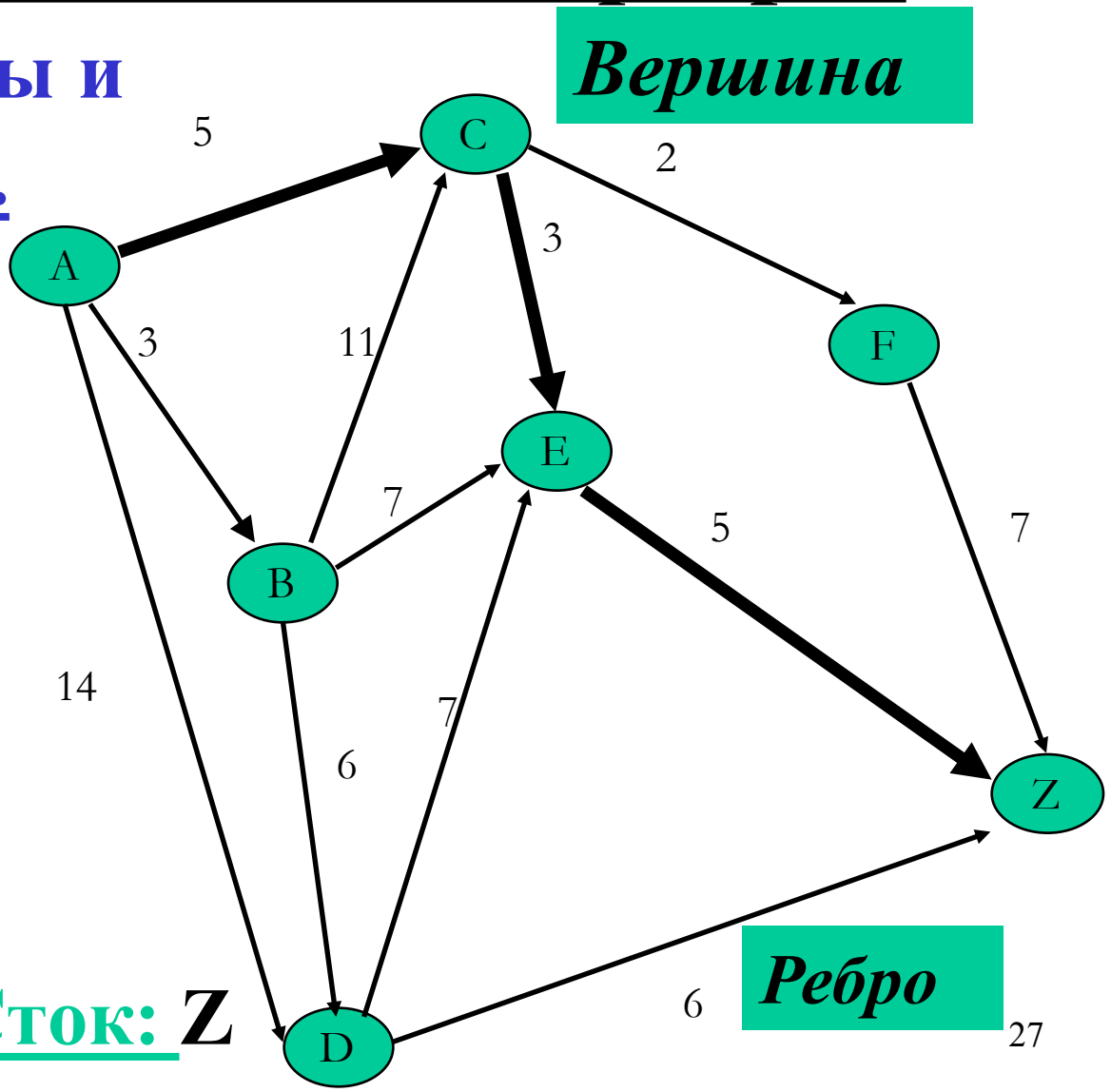
$B = b_1 + b_2 + \ldots + b_{1000}$ // 999

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

мулитипликативный 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

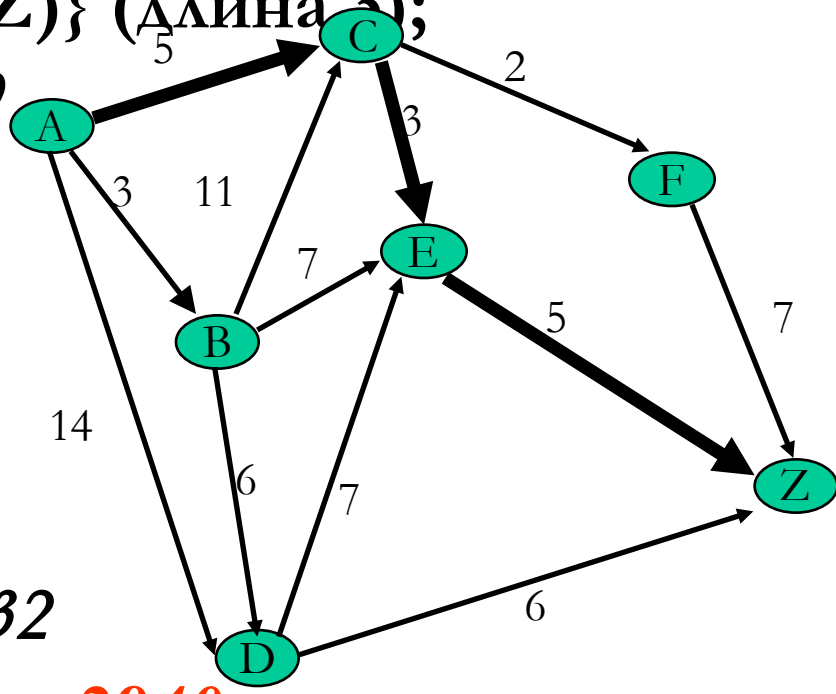
Полный путь **(длина 4); :**

ADBEZ =

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

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

WM(ADBEZ) = 14•6•7•5 = 2940



Полные пути –

пути из источника в сток (примеры):

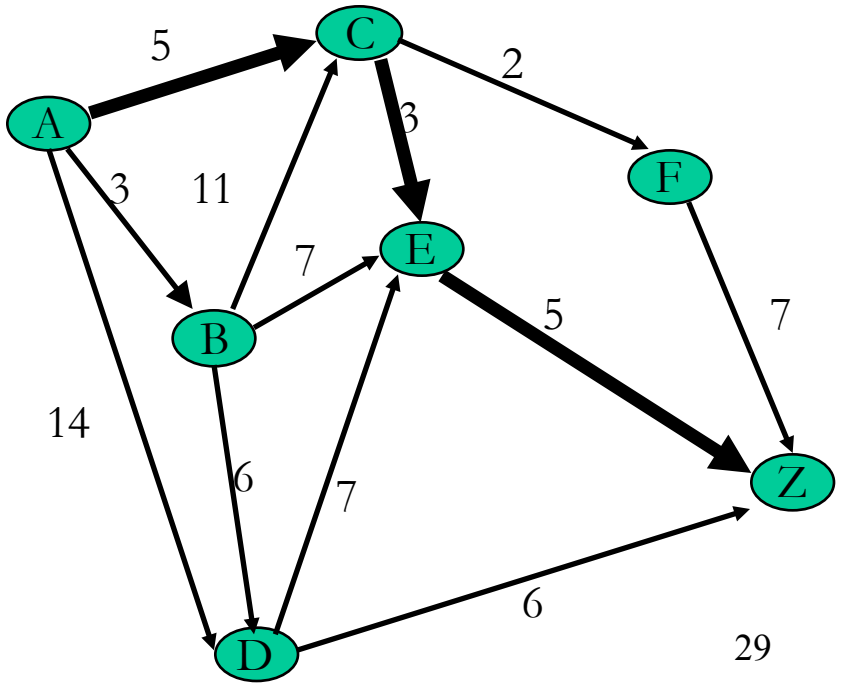
ADEZ: длина = 3; вес W(ADEZ) = *14+ 7 + 5 = 26;*

м-вес* WM(ADEZ) = *14•7•5=490

ABCFZ: длина = 4;

W(ABCFZ) = *3+7+ 2 + 7 = 19*

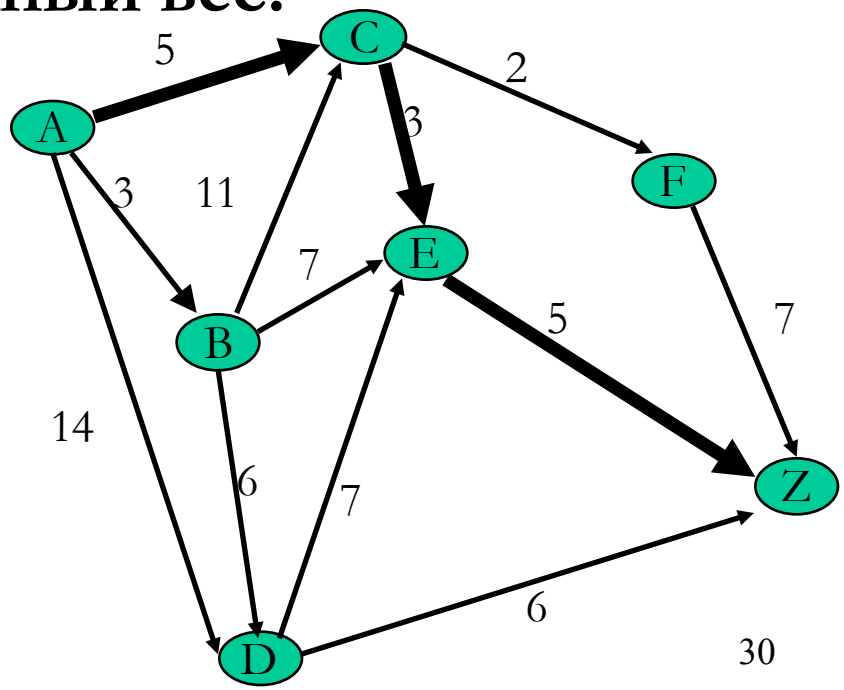
WM(ADEZ) = *3•7•2•7=294*



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.**



Интерпретации:

1. *“Экономика”:* *Вершины* – **города;** *Ребра* - **дороги;**

   Веса: *длина, время, стоимость и т.п.*

2. *“Физика” : Вершины – состояния процесса;*

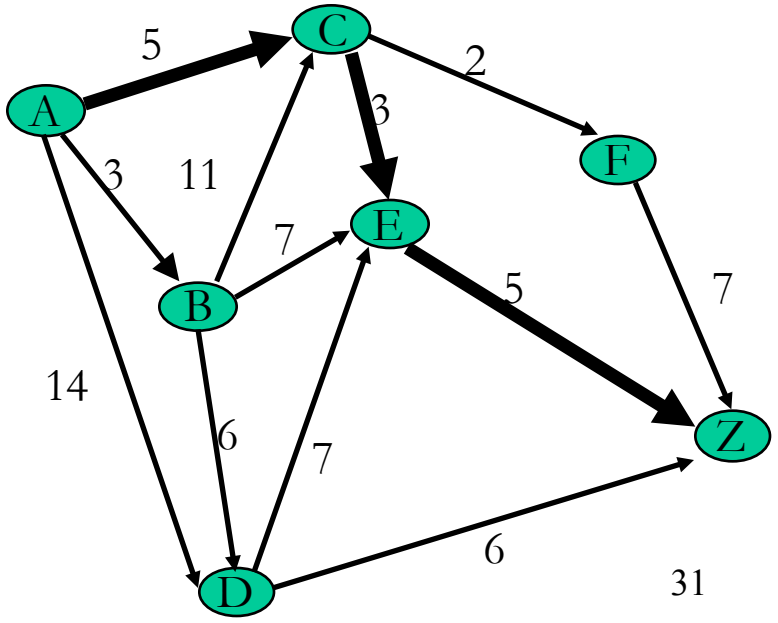
   Ребра: **возможные переходы между состояниями;**

   Веса: **энергии или вероятности переходов.**

3. *“Информатика”:* *Вершины – (промежуточные) задачи.*

   Ребро из A в B – **«выбор»,** *после которого решение задачи A сводится к решению задачи B.*

   Веса: «стоимость» выбора.



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),

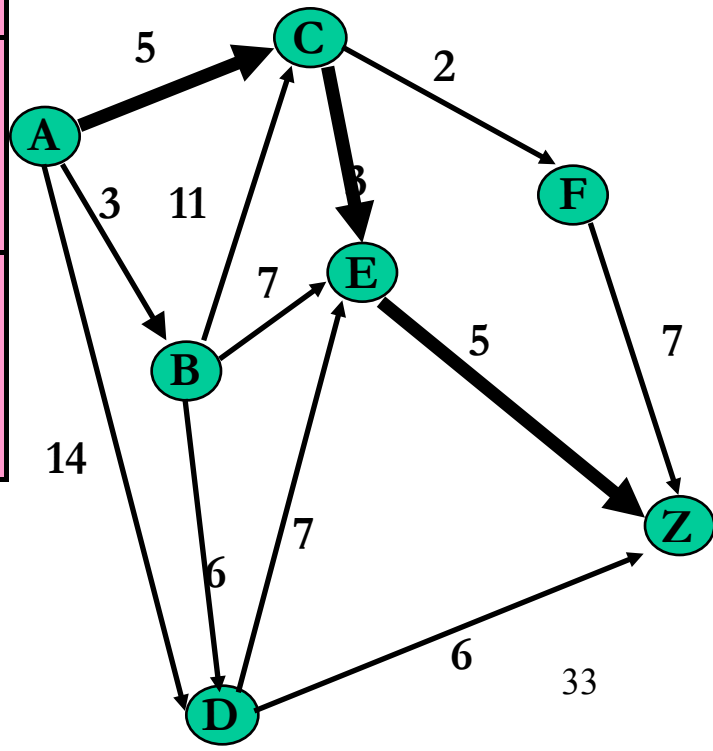
W(AC) + BestW(C),

W(AD) + BestW(D)

}

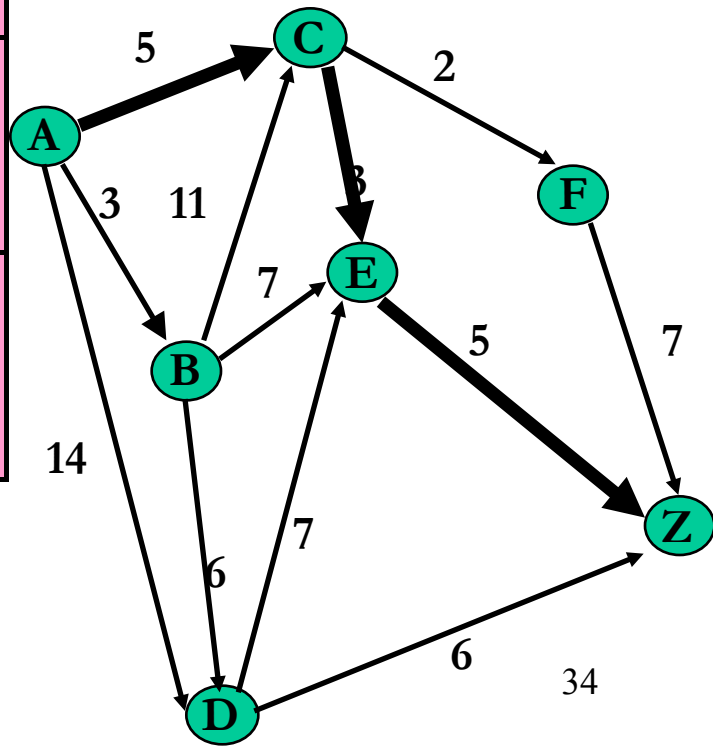
Ранг	Вершина	Исх. ребра	Вес ребра	Вес след. верш.	Лучш вес для ребра	Лучш вес для верш	Куда идти
0	Z	xxx	0	0	0	0	xxx
1	F	Z	7	0	7	7	Z
1	E	Z	5	0	5	5	Z
2	D	E	7	5	12	6	Z
		Z	6	0	6		
2	C	E	3	5	8	8	E
		F	2	7	9		
3	B	C	11	8	19	12	D, E
		D	6	6	12		
		E	7	5	12		
4	A	B					
		C					
		D					

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),
}

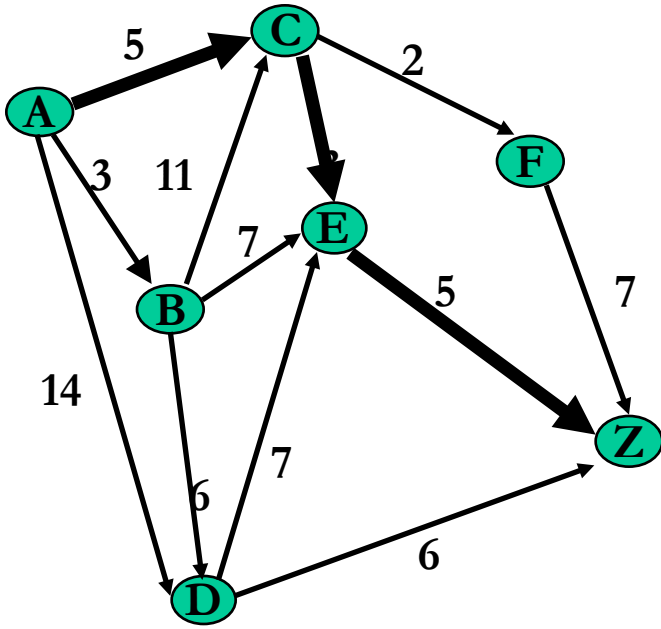


Best Weight: 13

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

1.2. *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

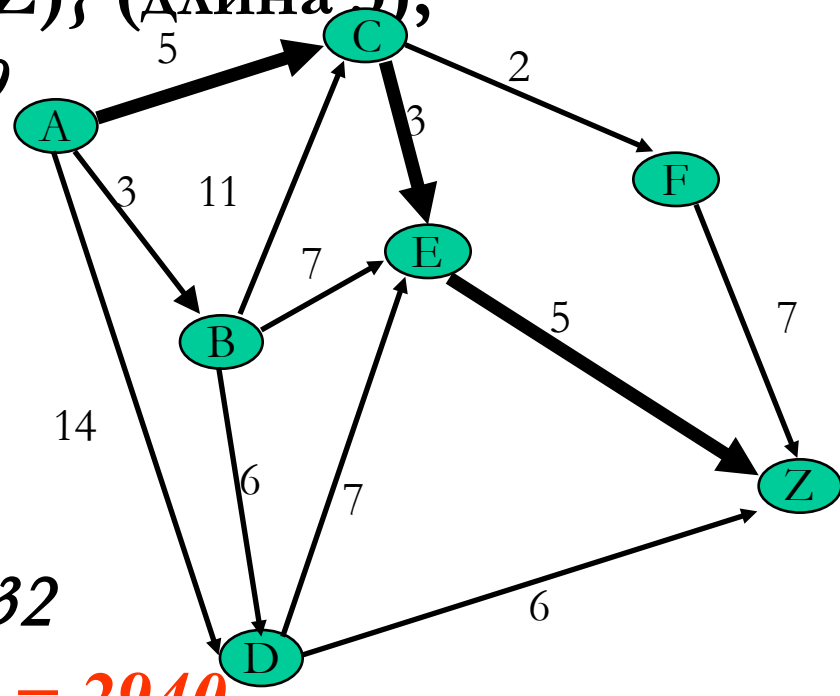
Полный путь **(длина 4); :**

ADBEZ =

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

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

WM(ADBEZ) = 14•6•7•5 = 2940

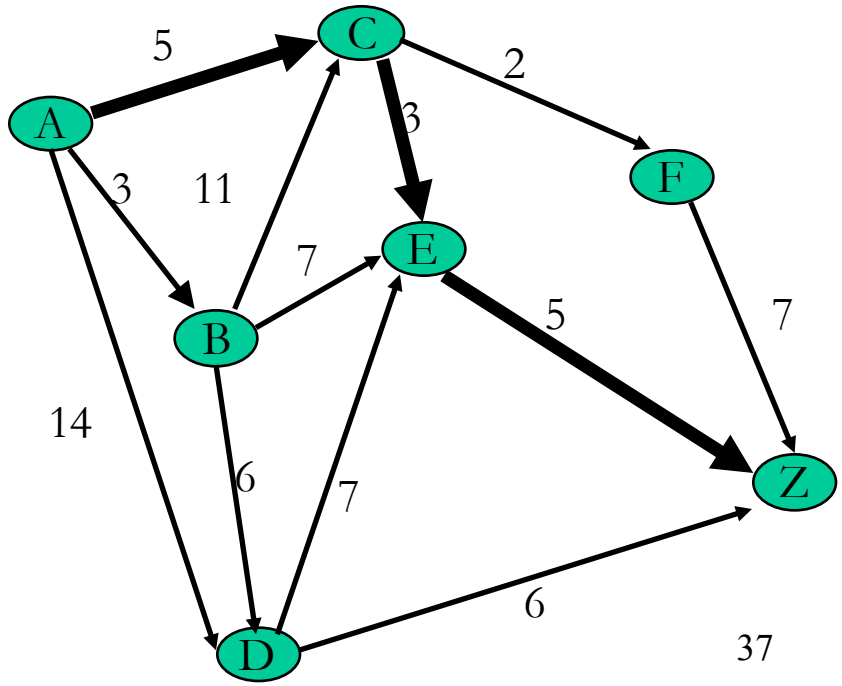


ДАНО: **Ориентированный ациклический граф с весами на ребрах**

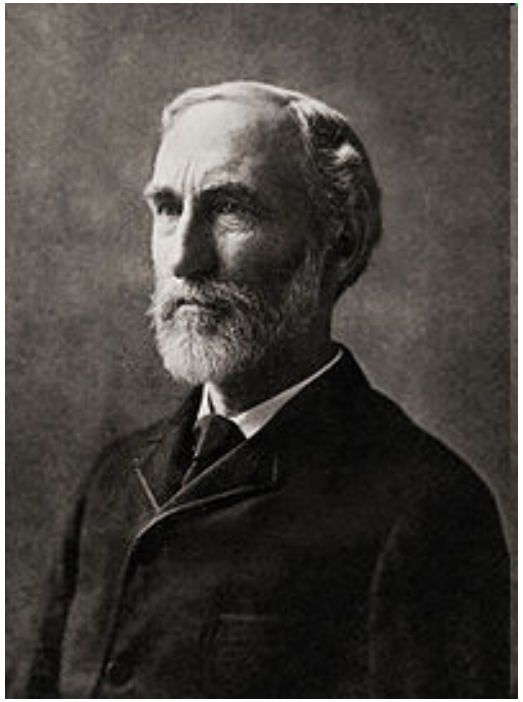
$G =< V, E, W; A, Z >$

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

Людвиг 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



Джозайя Уиллард Гиббс (*Josiah Willard Gibbs*; 1839 – 1903, США) — математик, физик и физикохимик, один из создателей статистической физики и математической теории термодинамики

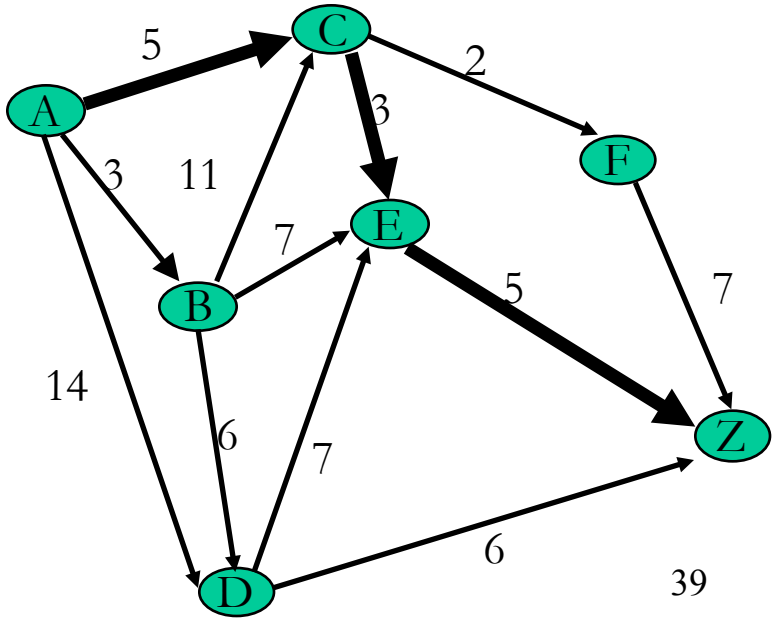
Эрнст Изинг (Ernst Ising, 1900-1998, Германия-США) - физик, позже - педагог, автор модели Изинга (см. предсказание спиралей в белке и т. п.)

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*



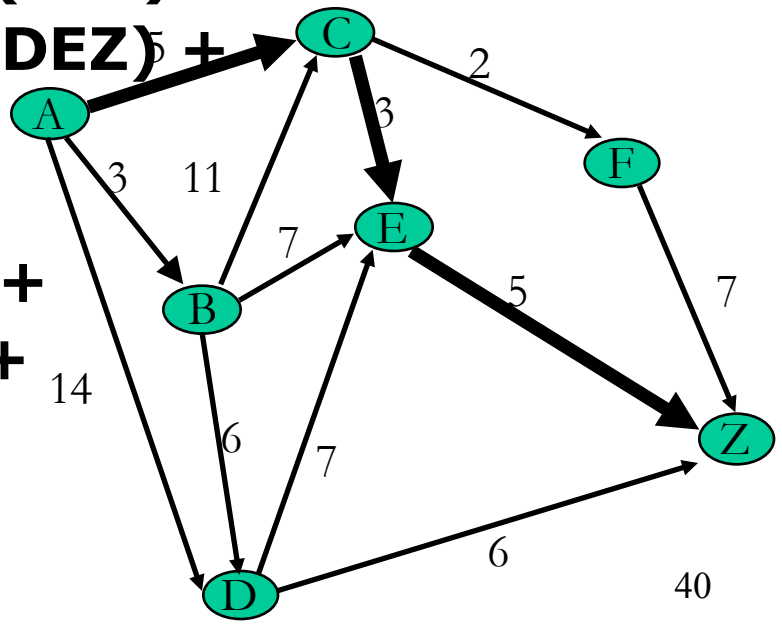
- Проход от стока к источнику:
  из W есть путь в V =>
  => W обрабатывается позже, чем V.

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 =>

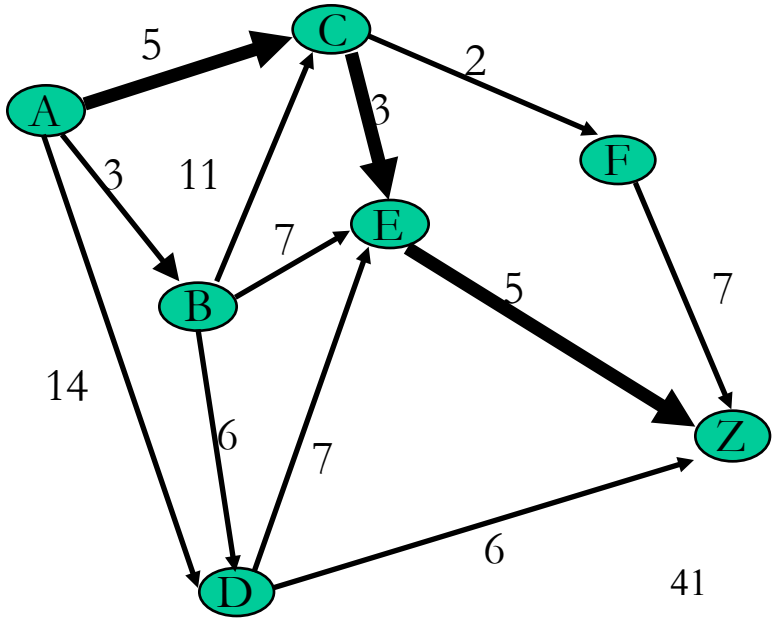
  => W обрабатывается позже, чем V.

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) =

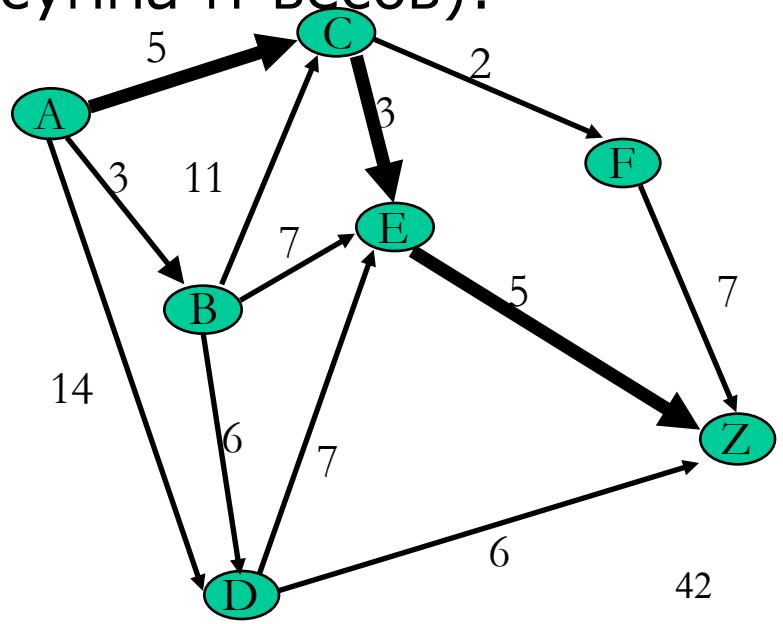
= W(BC)* Sum(C) +
+ W(BD)* Sum(D) +
+ W(BE)* Sum(E)



- 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)
}

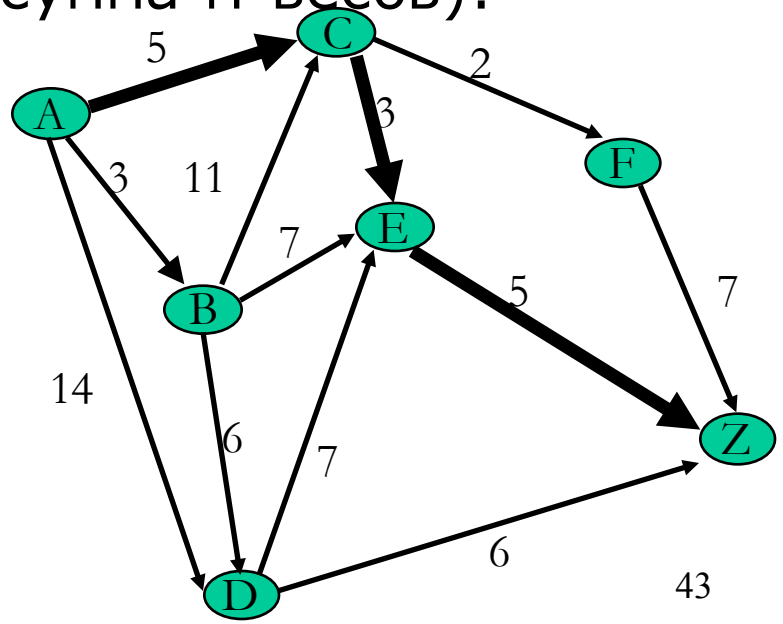


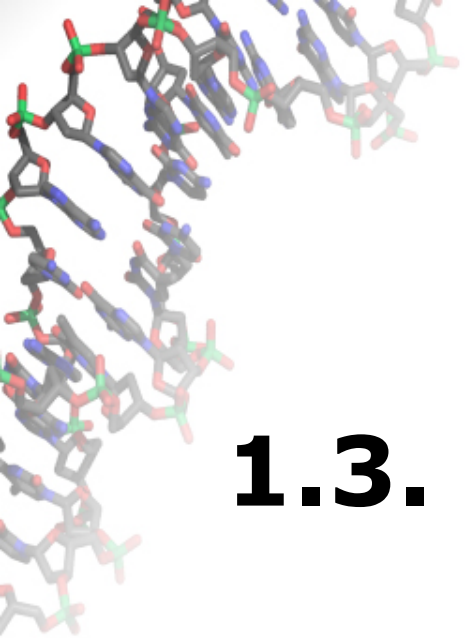
- Проход от стока к источнику:
  из W есть путь в V =>
  => W обрабатывается позже, чем V.
- Рекуррентное уравнение (минимальный путь)

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)
}





1.3. Алгебраическая основа динамического программирования

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

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

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

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

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

умножение относительно сложения

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

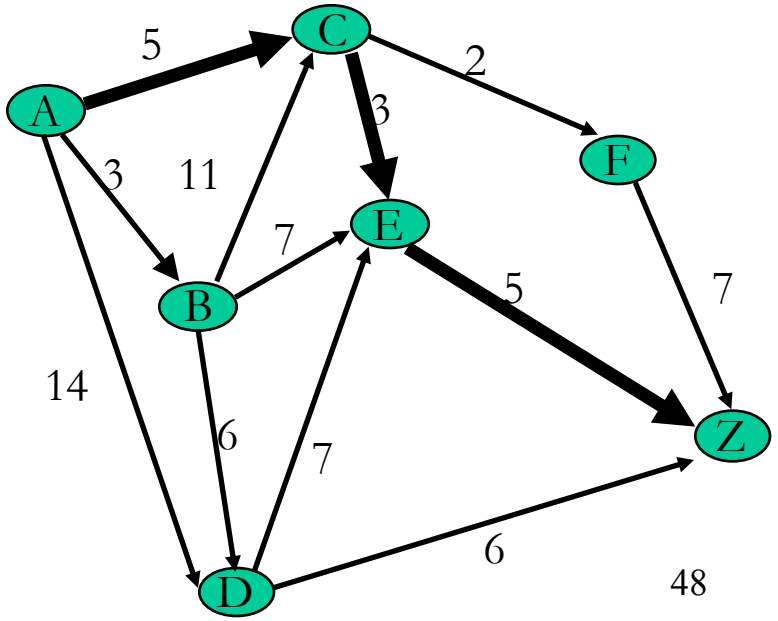
- Sum(B) =
- = M(BCEZ) + M(BCFZ) + M(BDZ) + M(BDEZ) + M(BEZ) =

= 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	***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

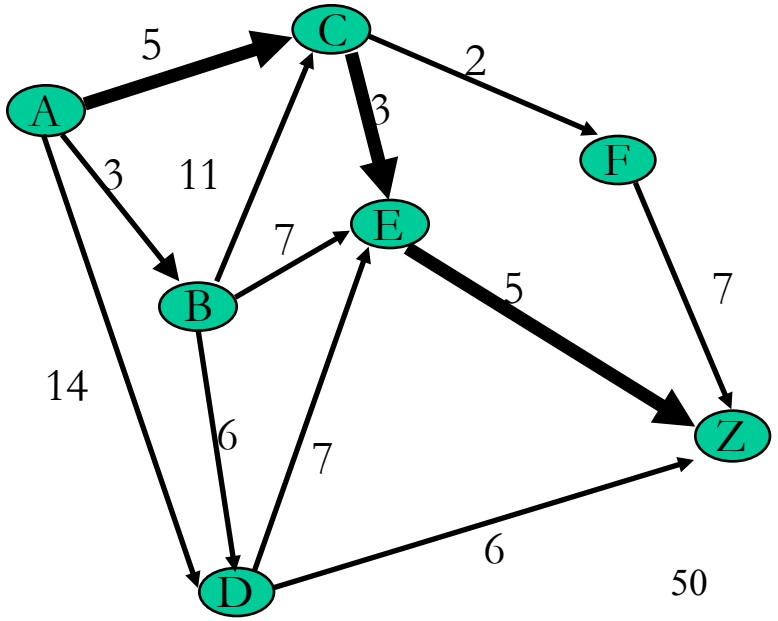
- Sum(B) =
- = M(BCEZ) + M(BCFZ) + M(BDZ) + M(BDEZ) + M(BEZ) =

= 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

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

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

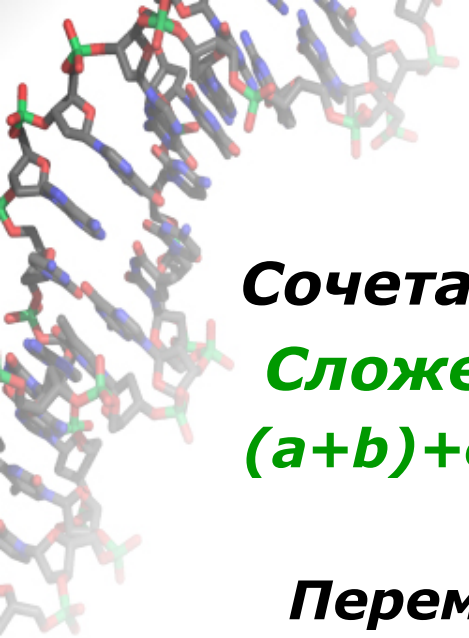
- **Обратные элементы (3-й класс☺) :**

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

- **РАСПРЕДЕЛИТЕЛЬНЫЙ ЗАКОН (ДИСТРИБУТИВНОСТЬ)**

умножение относительно сложения

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



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

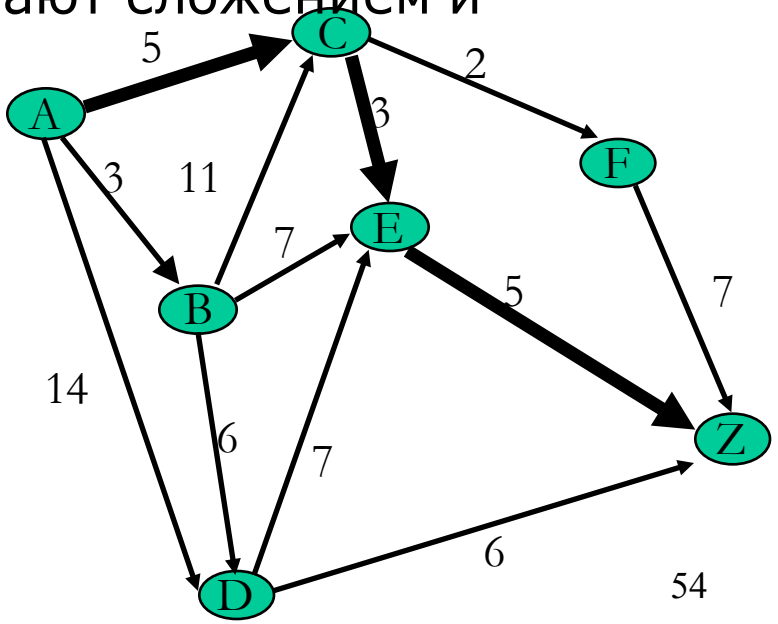
$(a+b)*c = a*c + b*c$ $a*(b+c) = a*b+a*c$

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 левый нейтральный элемент относительно операции *****;
- Операции и обычно называют сложением и умножением.

+ - «целевая» операция

*** - «соединительная» операция**



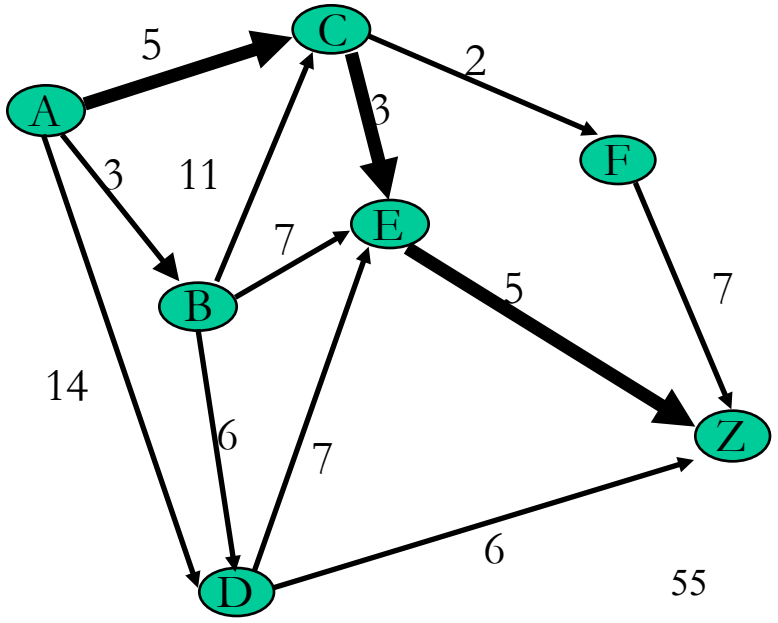
Примеры полуколец.

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

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

+ - «целевая» операция

*** - «соединительная» операция**

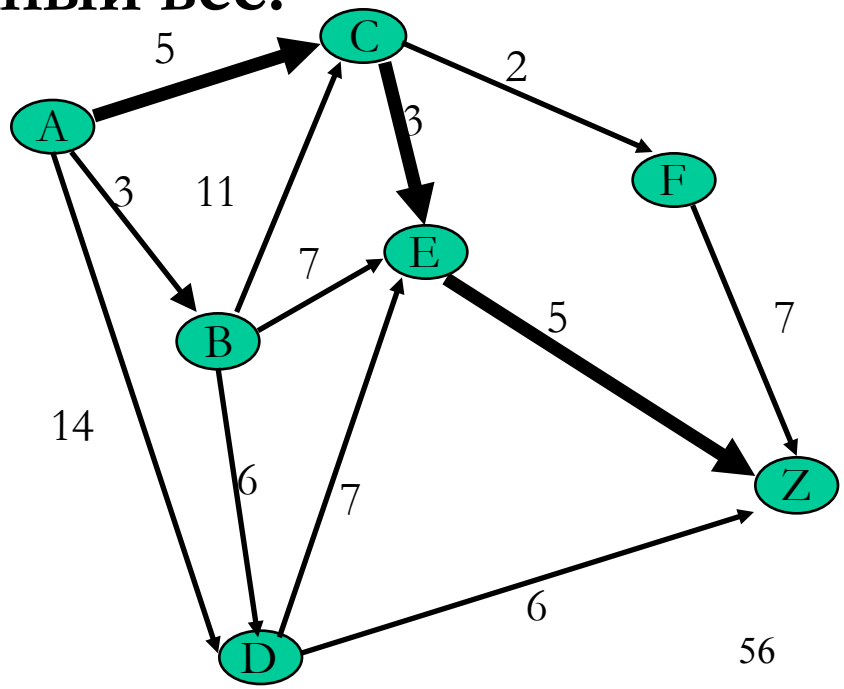


ДАНО: **Ориентированный ациклический граф с весами на ребрах**

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

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**

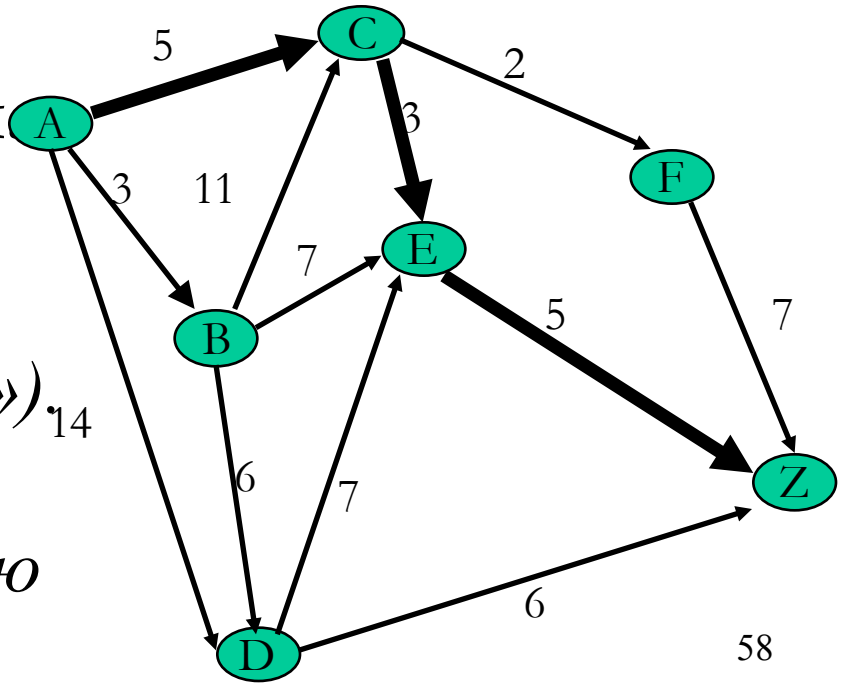
G =< *V, E, W; A, Z*>;

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»).*

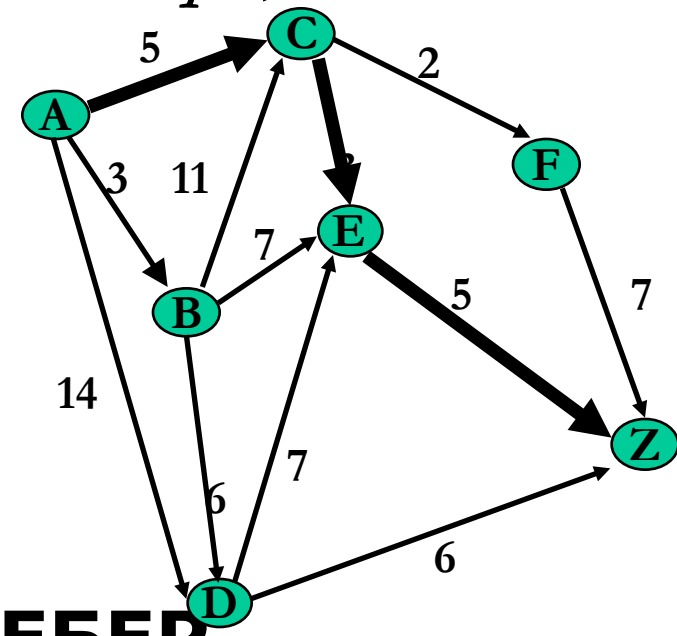
Операция + («сложение») *определяет целевую функцию («целевая операция»).*



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

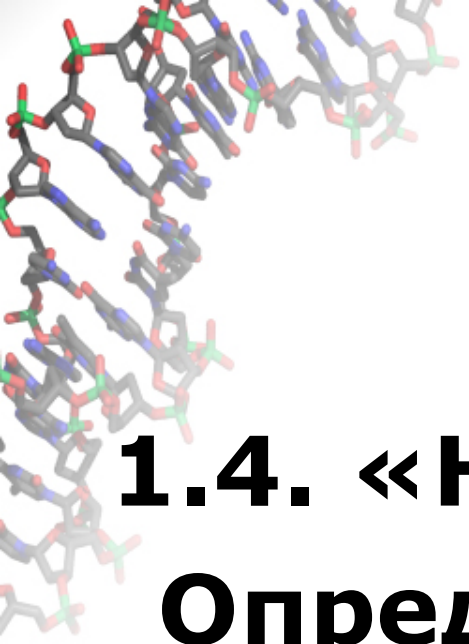
$G =< V, E, W; A, Z >;$

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



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

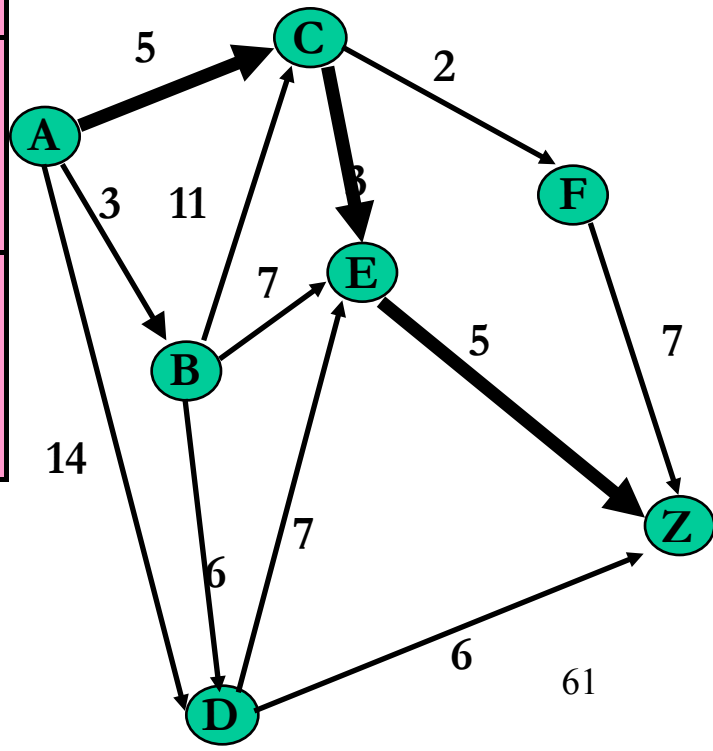
ВРЕМЯ РАБОТЫ ~ к-во РЕБЕР
ПАМЯТЬ ~ к-во ВЕРШИН



1.4. «Накопление результата». Определение рангов вершин.

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

Верш.	Ранг	Куда
Z	0	xxx
D	?	Z, E
E	1	Z
F	1	Z

Ранг вершины T - это длина (количество ребер) в самом длинном пути из T в сток Z.

Обозначение: Rank(T)

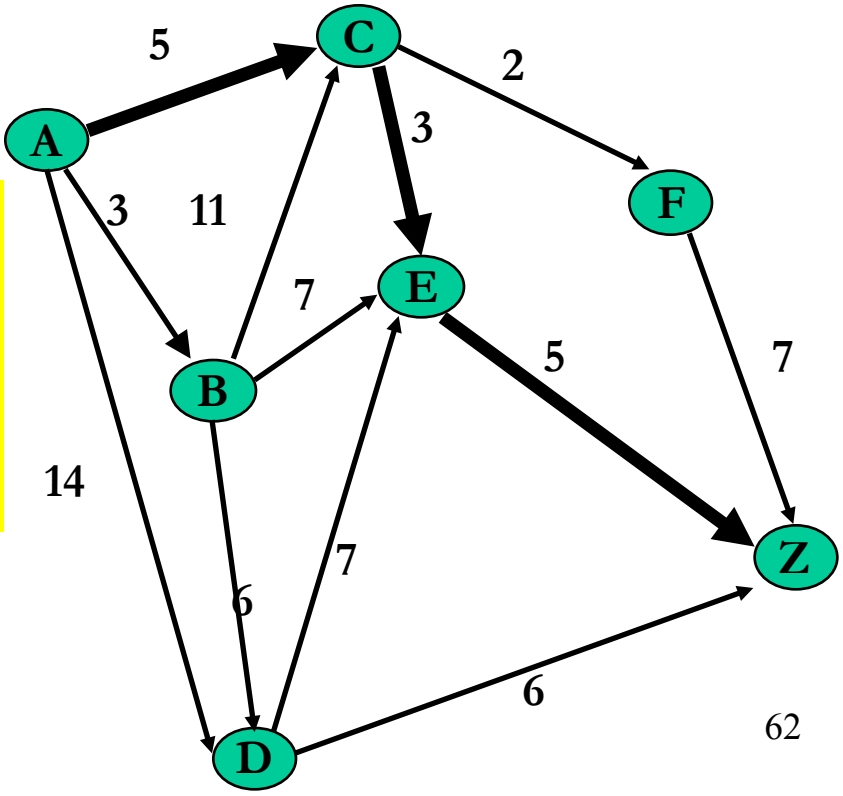
Для любого T: *Rank (T)* =

$=1+Max\ \{Rank(N_1),...,Rank(N_t)\}$

где N_1, ..., N_t - наследники вершины T

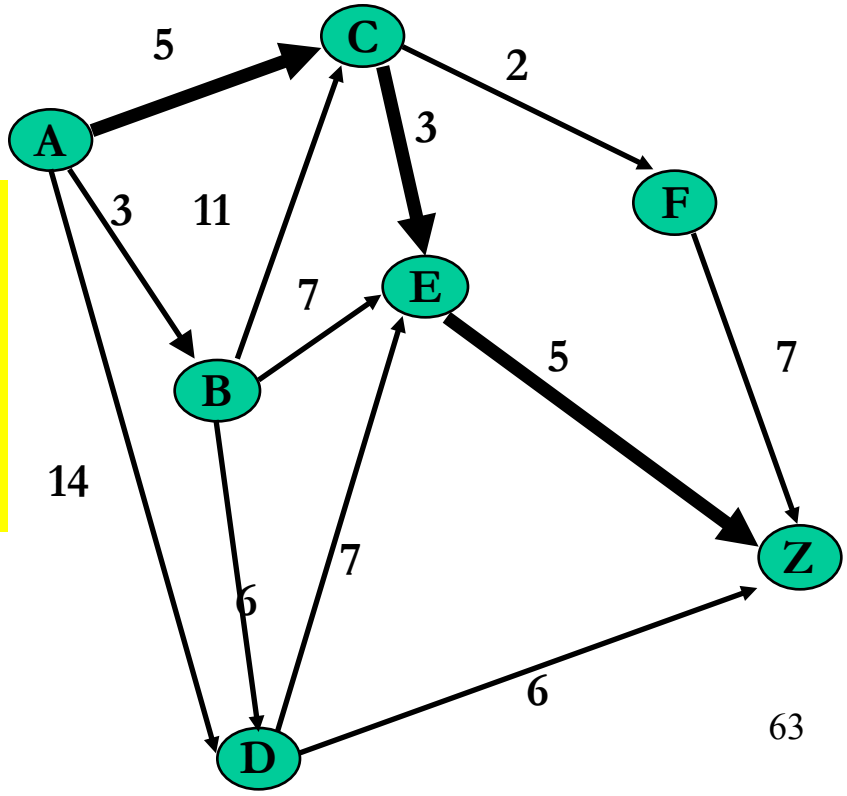
Rank (D) =

1+ max {(Rank(E), Rank(Z)}



Ранг вершины T - это количество ребер в самом длинном пути из T в сток Z. Обозначение: Rank(T)

Упражнение 1.1. Найти ранги всех вершин данного графа G.



Для любого T: *Rank (T)* =
=1+Max {Rank(N_1),...,Rank(N_t) }
где N_1, ..., N_t - наследники вершины T

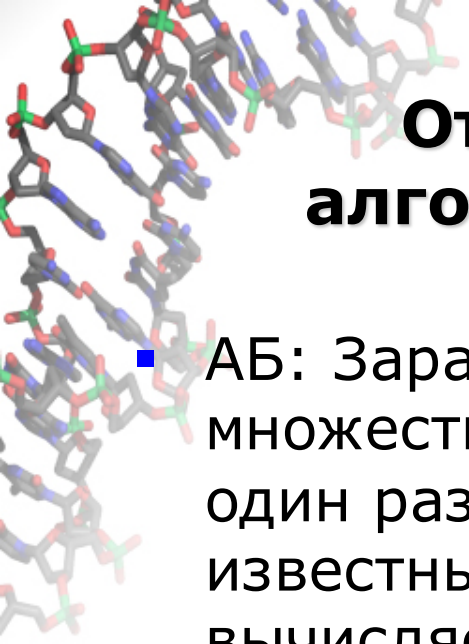
Rank (D) =
1+ max {(Rank(E), Rank(Z)}

Решение упражнения 1.1 Данные

```
typedef VertexPlus = record begin
   int InDegree;    int OutDegree;
   EdgeList InEdges;
   int Rank=0;      // ранг
   int Worked=0;   // кол-во обработанных наследников
end
typedef GraphPlus = record begin
   int NVertex;  VertexPlus array [1.. NVertex]  RealVertex
end

GraphPlus G;

int list Result = ☺;  //     Результат: вершины в
                                  // топологическом порядке.

// очередь вершин, ждущих обработки предшественников
int queue Wait = {Num(Sink)} // начинаем со стока
int K, KP; Vertex V, VP;
```

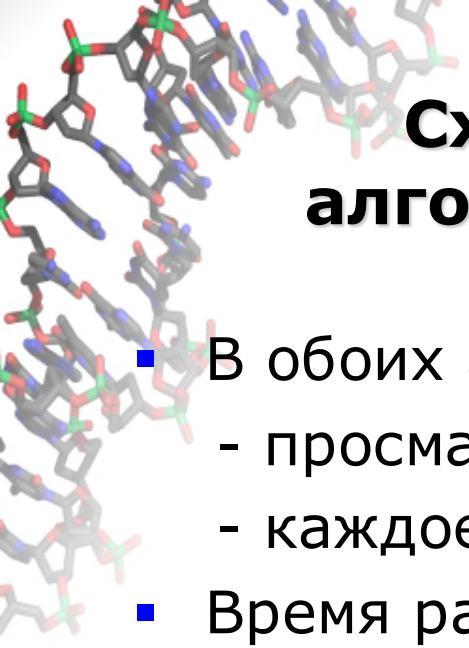
Решение упражнения 1.1. Алгоритм

```
while Wait ≠☺do begin
    K:= first(Wait);          // K – номер очередной вершины
    V:= G.RealVertex[K]; // V – сама вершина
    Result := Result+{K}; // включаем в список-результат
    for all E from V.InEdges do begin
      KP := E.Start; // VP – очередной предшественник V
      VP:=RealVertex[KP];// KP – его номер
      if (VP.Rank < V.Rank+1)
            then VP.Rank := V.Rank+1;
      endif;
   // количество обработанных наследников VP
      VP.Worked := VP.Worked+1;
   // все исходящие ребра VP обработаны
      if (VP.Worked = VP.OutDegree)
         then Wait := Wait+VP // добавить в очередь
      endif
    end
  // убрать из очереди обработанную вершину
    Wait := Wait – V;
end
```



Отличие алгоритма Беллмана от алгоритма с накоплением результата

- АБ: Заранее известен топологический порядок на множестве вершин. Каждая вершина просматривается один раз, при этом результаты для наследников уже известны. Результат для данной вершины вычисляется при обработке вершины.
- НР: Порядок обработки вершин определяется динамически. Результат для вершины известен ДО начала ее обработки. При обработке вершины перевычисляются временные результаты для ее предшественников (идет накопление результата).

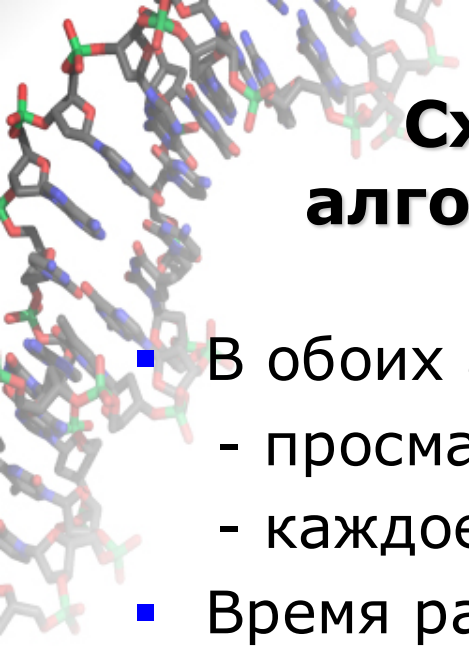


Сходство алгоритма Беллмана и алгоритма с накоплением результата

- В обоих алгоритмах:
  - просматриваются все ребра;
  - каждое ребро просматривается один раз.
- Время работы для обоих алгоритмов

  T ~ количество ребер
- Расход памяти для обоих алгоритмов

  M ~ количество вершин



Сходство алгоритма Беллмана и алгоритма с накоплением результата

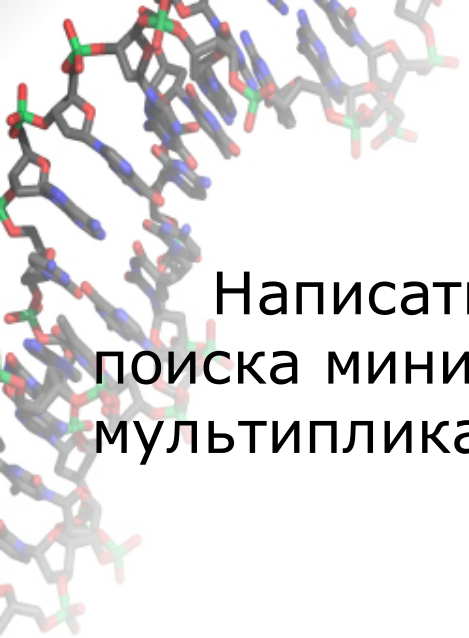
- В обоих алгоритмах:
  - просматриваются все ребра;
  - каждое ребро просматривается один раз.
- Время работы для обоих алгоритмов

  T ~ количество ребер
- Расход памяти для обоих алгоритмов

  M ~ количество вершин

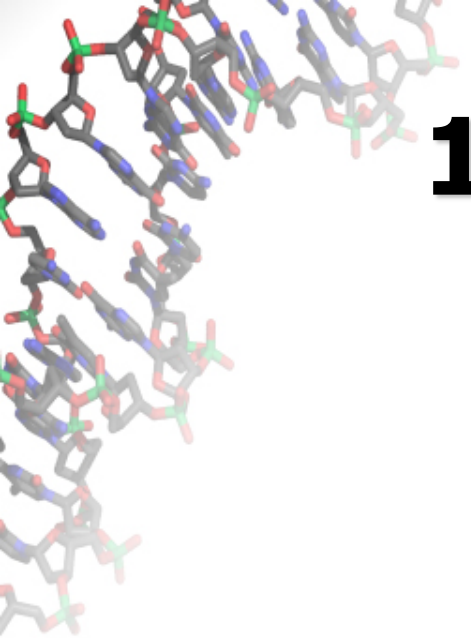
ОБА АЛГОРИТМА ПРИМЕНИМЫ ДЛЯ ПРОИЗВОЛЬНЫХ ПОЛУКОЛЕЦ

В отличие от алгоритма Дейкстры ☹



Упражнение 1.2.

Написать алгоритмы с накоплением результата для поиска минимального пути (задача Беллмана) и суммы мультипликативных весов (задача Больцмана).

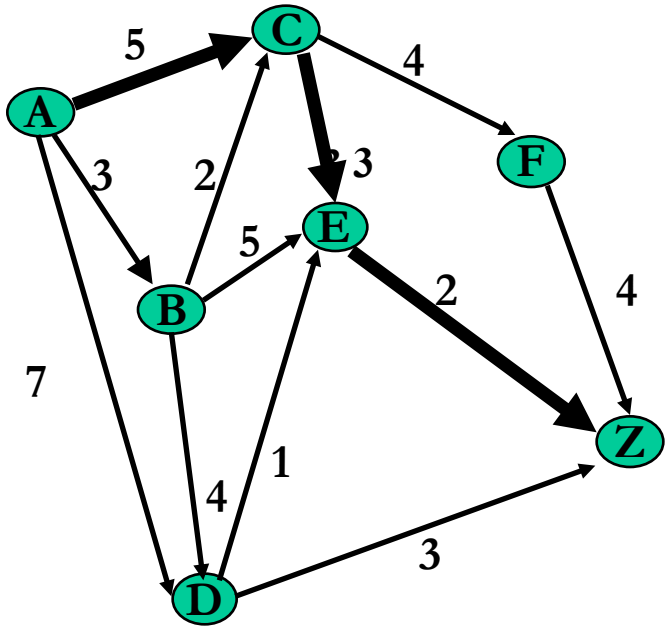


1.5. Завершающие штрихи

1.5A. Все минимальные пути

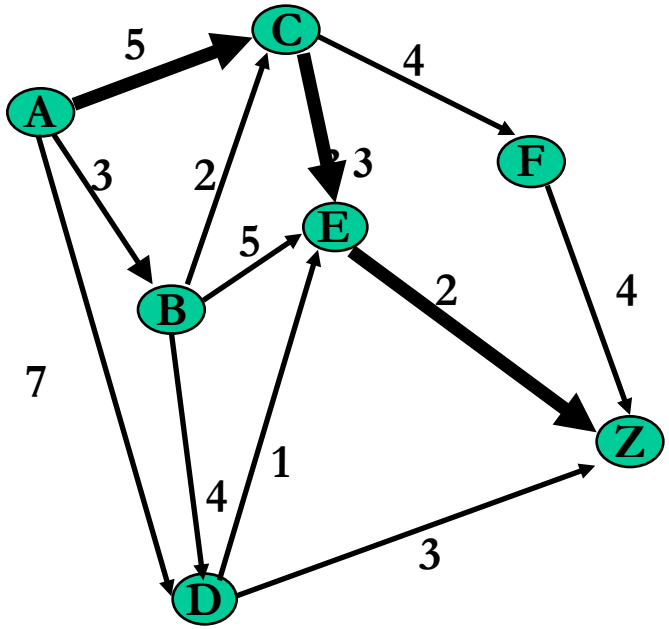
Ранг	Вершина	Исх. ребра	Вес ребра	Вес след. верш.	Лучший вес для ребра	Лучш. вес для верш.	Куда идти
0	Z	xxx	0	0	0	0	xxx
1	F	Z	4	0	4	4	
1	E	Z	2	0	2	2	
2	D	E	1	2	3	3	
		Z	3	0	3		
2	C	E	3	2	5	5	
		F	4	4	8		
3	B	C	2	5	7	7	
		D	4	3	7		
		E	5	2	7		
4	A	B	3	7	10	10	
		C	5	5	10		
		D	7	3	10		

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



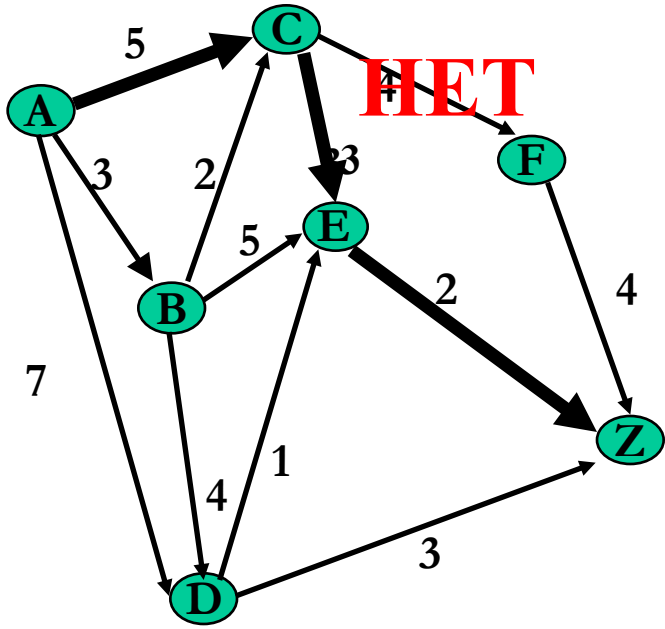
Ранг	Вершина	Исх. ребра	Вес ребра	Вес след. верш.	Лучший вес для ребра	Лучш. вес для верш.	Куда идти
0	Z	xxx	0	0	0	0	xxx
1	F	Z	4	0	4	4	Z
1	E	Z	2	0	2	2	Z
2	D	E	1	2	3	3	E, Z
		Z	3	0	3		
2	C	E	3	2	5	5	E
		F	4	4	8		
3	B	C	2	5	7	7	C, D, E
		D	4	3	7		
		E	5	2	7		
4	A	B	3	7	10	10	B, C, D
		C	5	5	10		
		D	7	3	10		

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



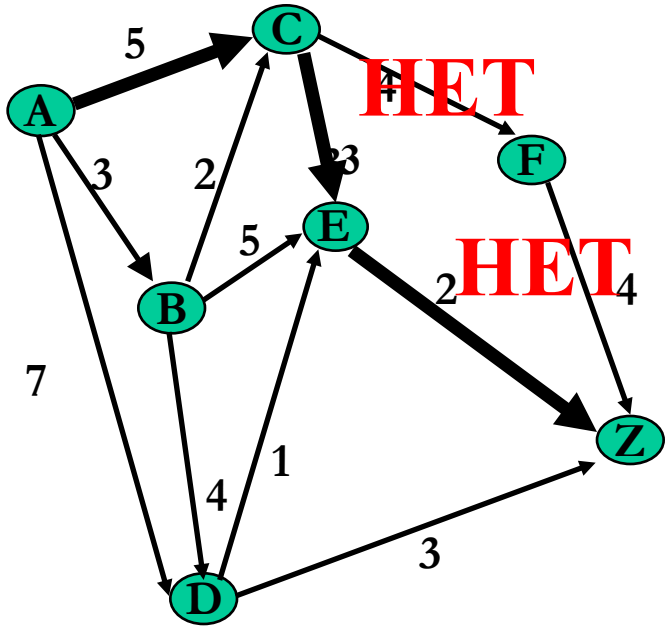
Ранг	Вершина	Исх. ребра	Вес ребра	Вес след. верш.	Лучший вес для ребра	Лучш. вес для верш.	Куда идти
0	Z	xxx	0	0	0	0	xxx
1	F	Z	4	0	4	4	Z
1	E	Z	2	0	2	2	Z
2	D	E	1	2	3	3	E, Z
		Z	3	0	3		
2	C	E	3	2	5	5	E
		F	4	4	8		
3	B	C	2	5	7	7	C, D, E
		D	4	3	7		
		E	5	2	7		
4	A	B	3	7	10	10	B, C, D
		C	5	5	10		
		D	7	3	10		

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



Ранг	Вершина	Исх. ребра	Вес ребра	Вес след. верш.	Лучший вес для ребра	Лучш. вес для верш.	Куда идти
0	Z	xxx	0	0	0	0	xxx
1	F	Z	4	0	4	4	Z
1	E	Z	2	0	2	2	Z
2	D	E	1	2	3	3	E, Z
		Z	3	0	3		
2	C	E	3	2	5	5	E
		F	4	4	8		
3	B	C	2	5	7	7	C, D, E
		D	4	3	7		
		E	5	2	7		
4	A	B	3	7	10	10	B, C, D
		C	5	5	10		
		D	7	3	10		

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

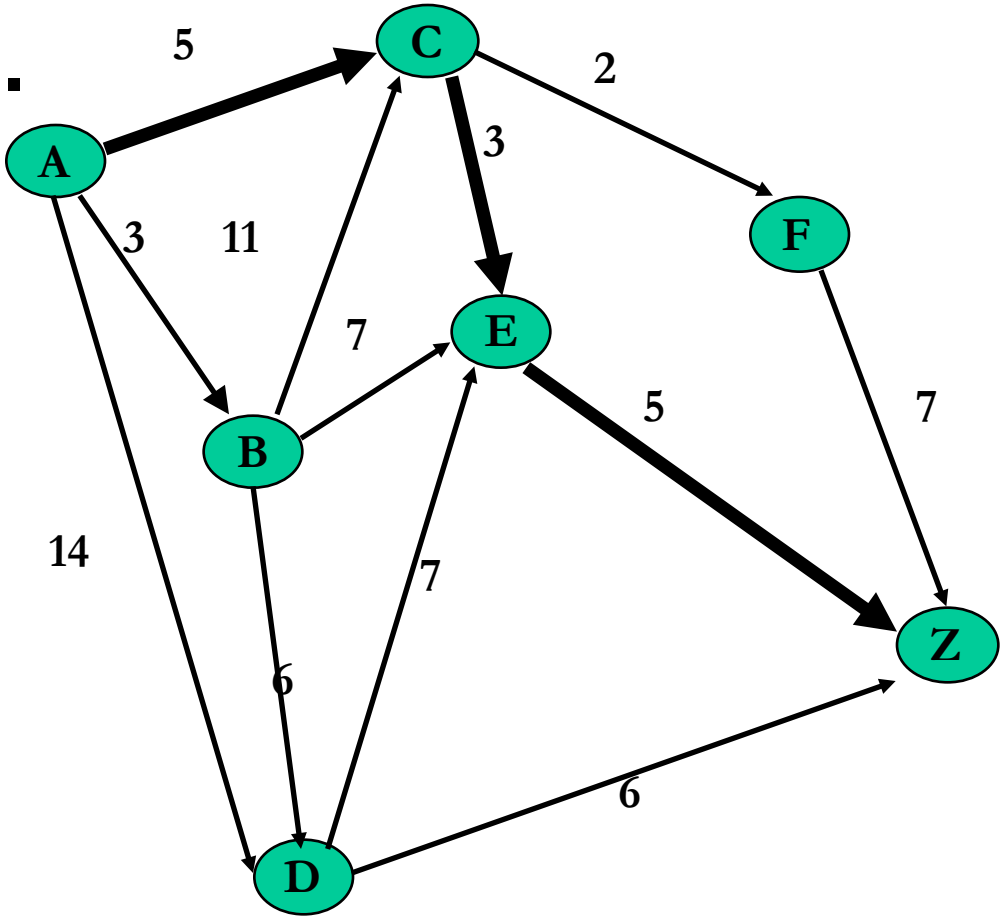


1.5Б. Общее количество путей

Упражнение 1.3.

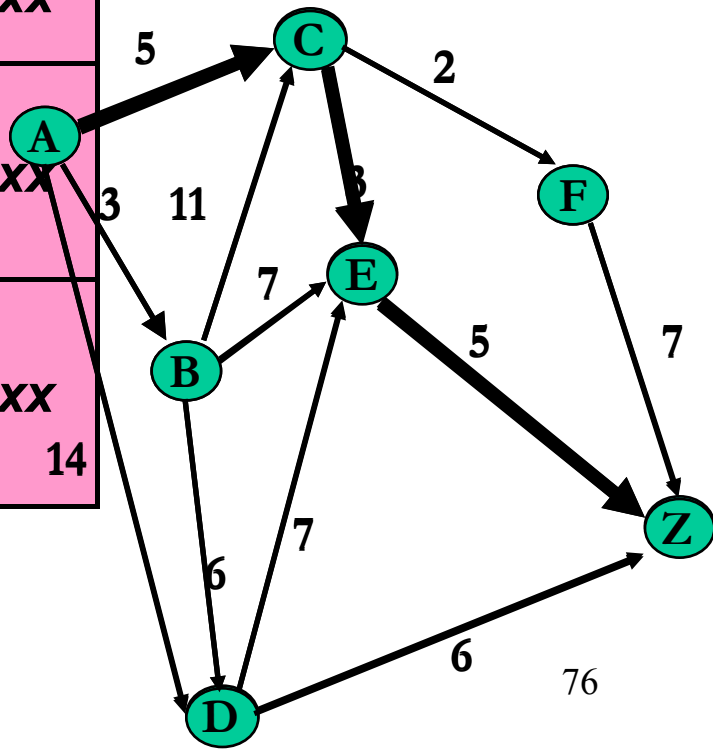
Найти:

Общее количество путей из A в Z



Ранг	Вершина	Исх. ребра	Вес ребра	Вес след. верш.	Сумм. вес для путей через ребро	Сумм. вес для верш	Куда идти
0	Z	xxx	xxx	xxx	xxx	1	xx
1	F	Z	1	1	1	1	xx
1	E	Z	1	1	1	1	xx
2	D	E	1	1	1	2	xx
		Z	1	1	1		
2	C	E	1	1	1	2	xxx
		F	1	1	1		
3	B	C	1	2	2	5	xxx
		D	1	2	2		
		E	1	1	1		
4	A	B	1	5	5	9	xxx
		C	1	2	2		
		D	1	2	2		14

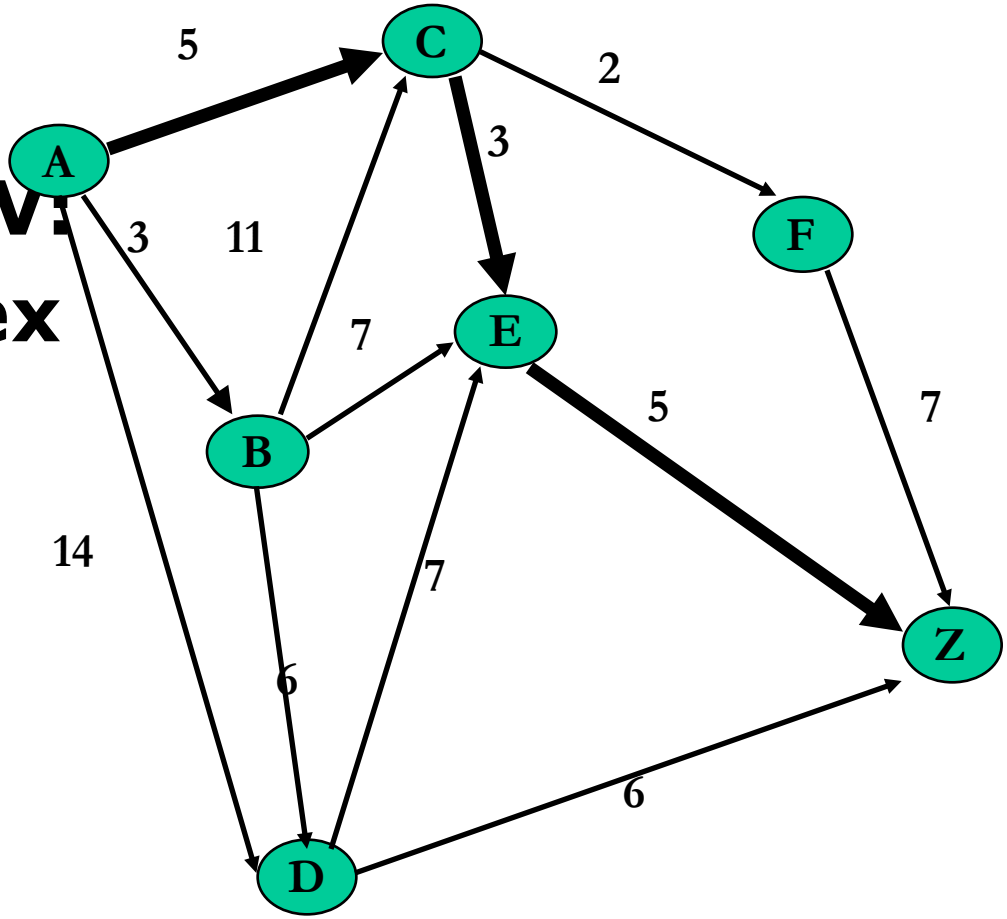
BestW(B) =
= summa{
W(BC) * BestW(C),
W(BD) * BestW(D),
W(BE) * BestW(E),
}



Best Weight: 13
Best Path: ACEZ

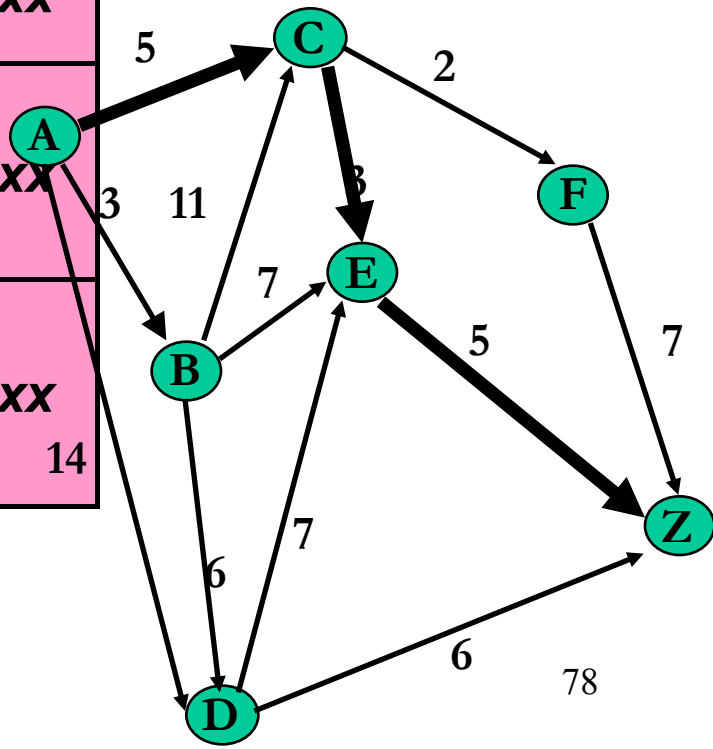
1.5B. Сумма Больцмана для каждой вершины.

Найти для каждой вершины V сумму м-весов всех полных путей, проходящих через V



Ранг	Вершина	Исх. ребра	Вес ребра	Вес след. верш.	Сумм. вес для путей через ребро	Сумм. вес для верш	Куда идти
0	Z	xxx	xxx	xxx	xxx	1	
1	F	Z	7	1	7	7	
1	E	Z	5	1	5	5	
2	D	E	7	5	35	41	
		Z	6	1	6		
2	C	E	3	5	15	29	xxx
		F	2	7	14		
3	B	C	11	29	319	600	xxx
		D	6	41	246		
		E	7	5	35		
4	A	B	3	600	1800	2519	xxx
		C	5	29	145		
		D	14	41	574		

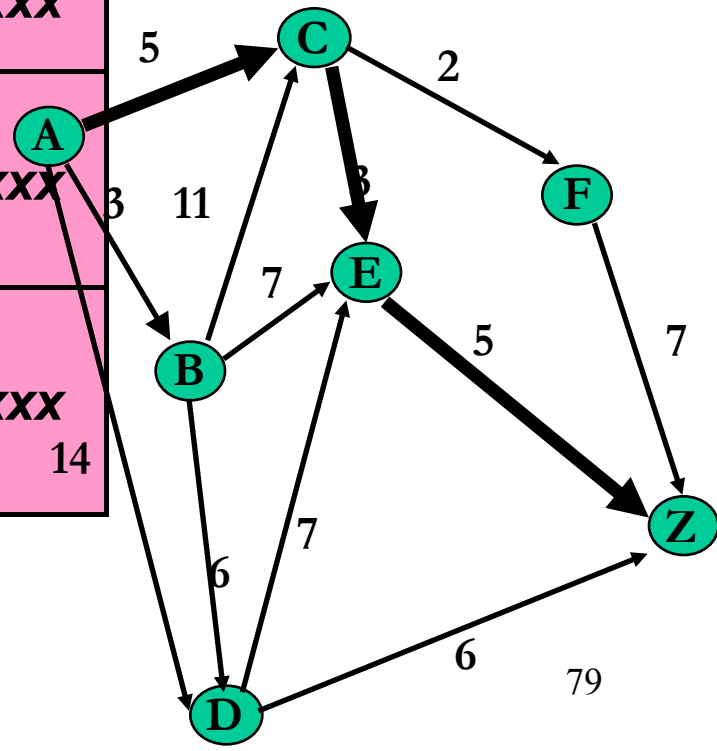
BestFwW(B) =
= summa{
W(BC) * BestFwW(C),
W(BD) * BestFwW(D),
W(BE) * BestFwW(E),
}



Sum: 2519

Анти Ранг	Вер ши на	Вход. ребра	Вес ребра	Вес пред. верш.	Сумм вес для вх. путей через ребро	Сумм. вес для верш	Куда идти
0	A	xxx	xxx	xxx	xxx	1	xx
1	B	A	3	1	3	3	xx
2	C	A	5	1	5	38	xx
		B	11	3	33		xx
2	D	A	14	1	14	32	
		B	6	3	18		xxx
3	E	B	7	3	21	359	
		C	3	38	114		xxx
		D	7	32	224		
3	F	C	2	38	76	76	
4	Z	D	6	32	192	2519	xxx
		E	5	359	1795		
		F	7	76	532		

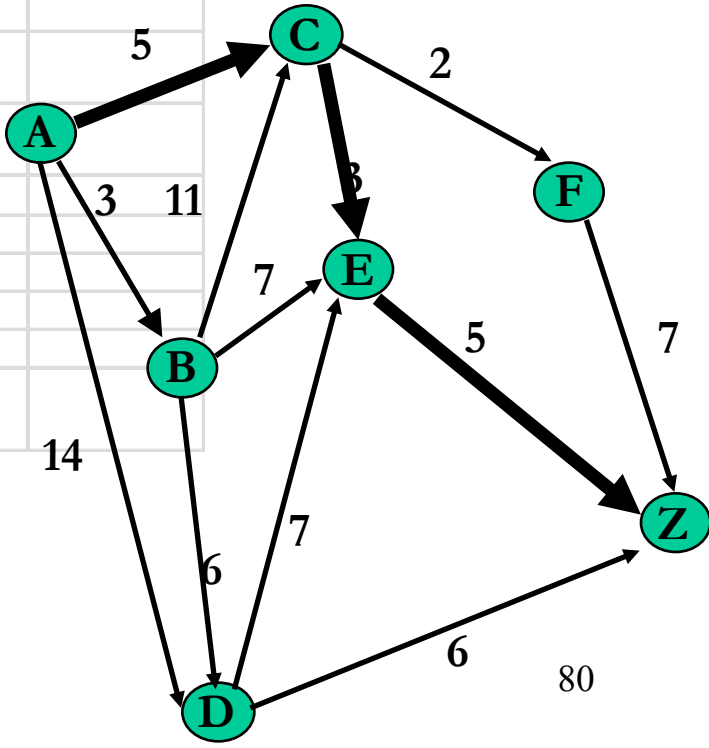
BestPrW(E) =
= summa{
W(BE) * BestPrW(B),
W(CE) * BestPrW(C),
W(DE) * BestPrW(D),
}



Sum: 2519

Вершина	Сумм. вес для вых. путей верш	Сумм. вес для вх. путей верш
A	2519	1
B	600	3
C	29	38
D	41	32
E	5	359
F	7	76
Z	1	2519

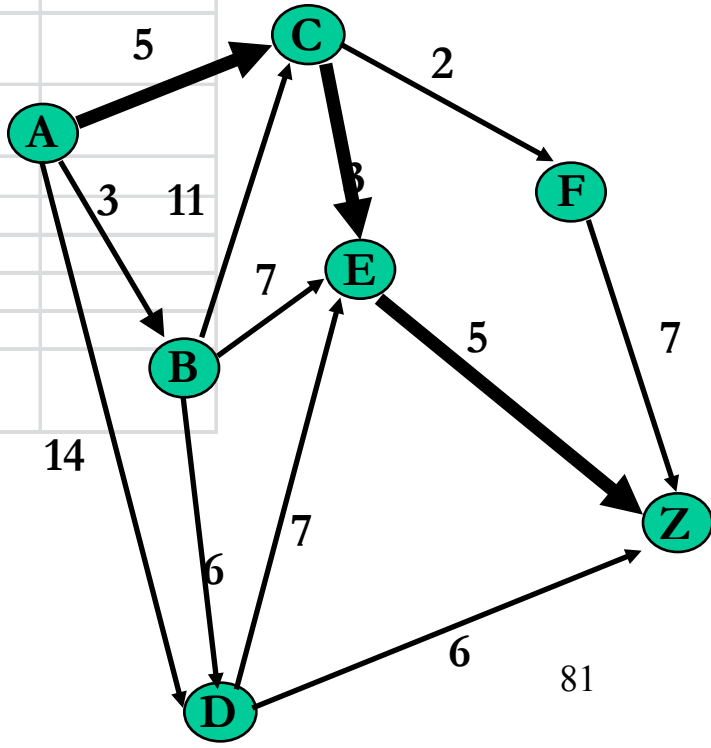
BestPrW(E) =
= summa{
W(BE) * BestPrW(B),
W(CE) * BestPrW(C),
W(DE) * BestPrW(D),
}



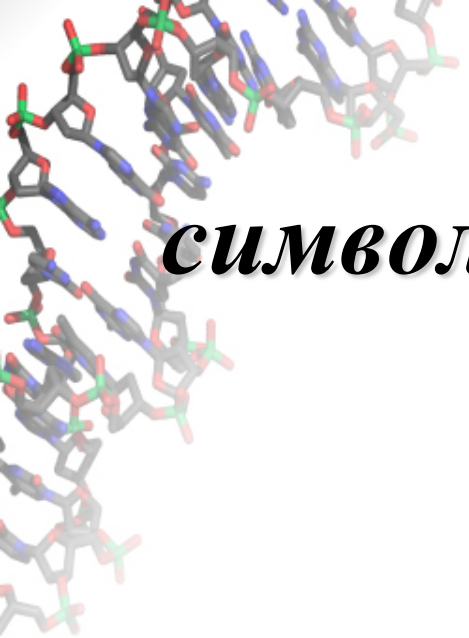
Sum : 2519

Вершина	Сумм. вес для вых. путей верш	Сумм. вес для вх. путей верш	Спец. сумма Больцмана
A	2519	1	2519
B	600	3	1800
C	29	38	1102
D	41	32	1312
E	5	359	1795
F	7	76	532
Z	1	2519	2519

BestPrW(E) =
= summa{
W(BE) * BestPrW(B),
W(CE) * BestPrW(C),
W(DE) * BestPrW(D),
}



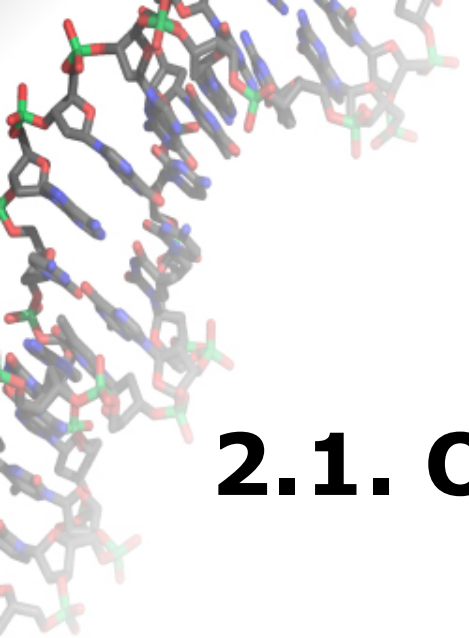
Best Weight: 2519



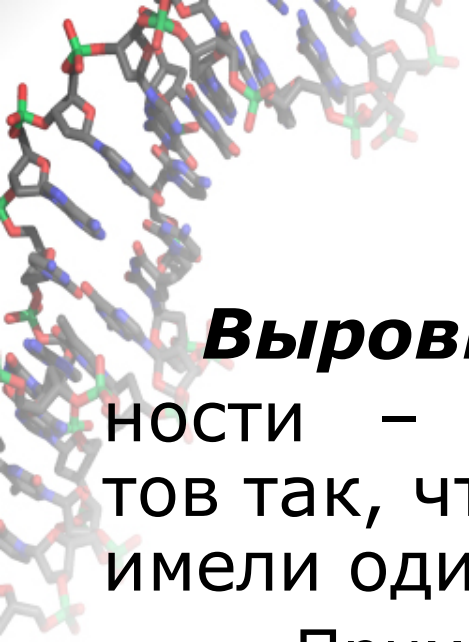
Анализ символьных последовательностей

по мотивам биоинформатики

Тема 2
Глобальное выравнивание



2.1. Определение выравнивания.



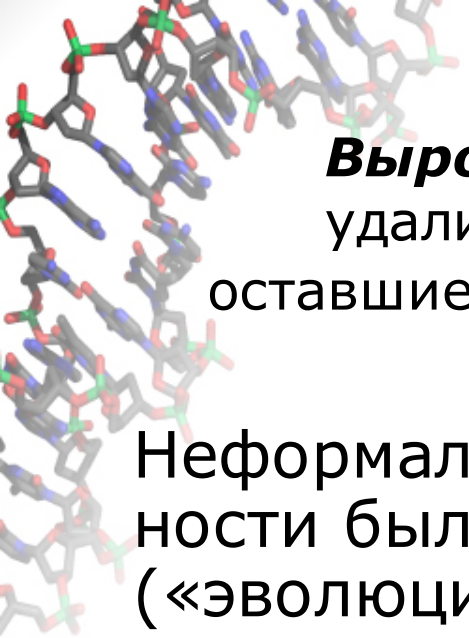
Определение.

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

Пример.

S1 = “КРОКОДИЛИЩЕ”; S2 = «ВОЛКОДАВУШКА»

КРОКОДИЛИЩЕ-	**КРО-КОДИЛИЩЕ-**
ВОЛКОДАВУШКА	**В-ОЛКОДАВУШКА**

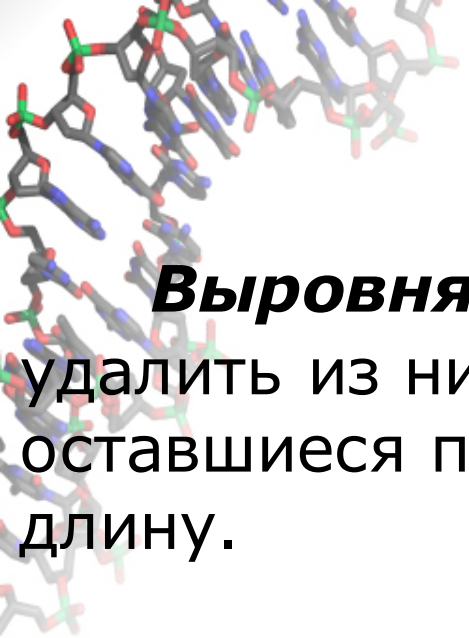


Выровнять две символьные последовательности – удалить из них несколько фрагментов так, чтобы оставшиеся последовательности имели одинаковую длину.

Неформально: предполагается, что последовательности были получены «редактированием» («эволюцией») из общего предка.

Требуется: установить соответствующие друг другу участки.

Предположение: были только замены, удаления и вставки. Не было перестановок фрагментов, дупликаций и т.п.



Пример 2

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

S1 = “ПОДБЕРЕЗОВИК”

S2 = «ПРЕДОСИНОВИЧКИ»

Варианты выравниваний

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

```
--ПОДБЕРЕЗОВИК        ПОДБЕРЕЗОВИК--
ПРЕДОСИНОВИЧКИ        ПРЕДОСИНОВИЧКИ

ПО-ДБЕРЕЗОВИК--       ПО-ДБЕРЕЗОВИК--
ПРЕДОСИН-ОВИЧКИ       ПРЕД-ОСИНОВИЧКИ

ПО-ДБЕРЕЗОВИ-К-
ПРЕД-ОСИНОВИЧКИ
```

Варианты выравниваний

Какой вариант выбрать?

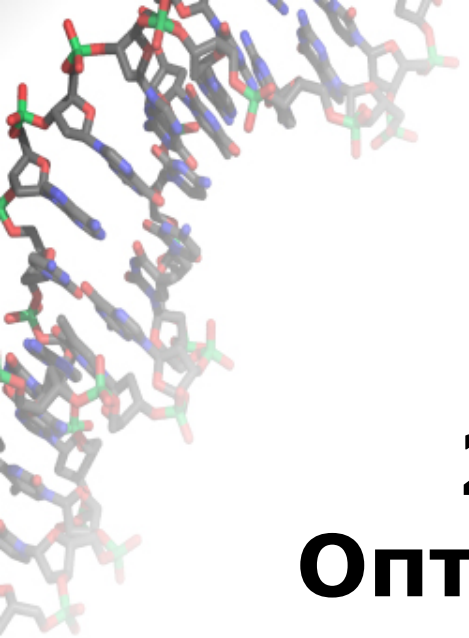
```
--ПОДБЕРЕЗОВИК
ПРЕДОСИНОВИЧКИ
```

```
ПОДБЕРЕЗОВИК--
ПРЕДОСИНОВИЧКИ
```

```
ПО-ДБЕРЕЗОВИК--
ПРЕДОСИН-ОВИЧКИ
```

```
ПО-ДБЕРЕЗОВИК--
ПРЕД-ОСИНОВИЧКИ
```

```
ПО-ДБЕРЕЗОВИ-К-
ПРЕД-ОСИНОВИЧКИ
```



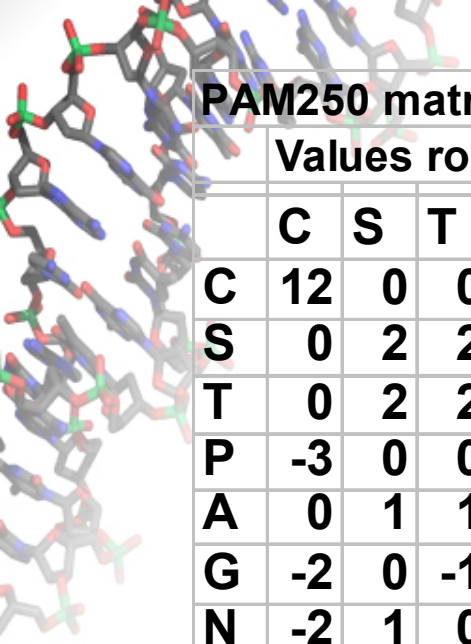
2.2. Вес выравнивания. Оптимальное выравнивание.

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
```

Штрафы за удаление:
фрагмента = 0;
символа = -1

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



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

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

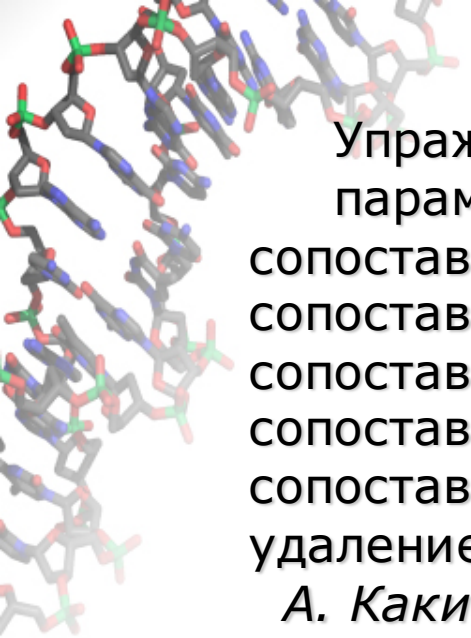
-7 -7

Штрафы за удаление:

фрагмента = 7;

символа = -1

$$Score = \Sigma\, m(i,j) - GapPen = 21 - 3 - 14 = 4$$



Упражнение 2.1. Вес выравнивания задается следующими параметрами:

сопоставление одинаковых гласных букв: *10;*
сопоставление одинаковых согласных букв: *7;*
сопоставление различных гласных букв: -3*;*
сопоставление различных согласных букв: -5*;*
сопоставление гласной и согласной букв: -10*;*
удаление буквы: -d (d >0).

А. Какие из указанных ниже выравниваний являются оптимальными при некоторых значениях параметра d? Укажите эти значения.

Б.* Как нужно изменить определение веса выравнивания, чтобы выравнивание (Д) могло стать оптимальным?

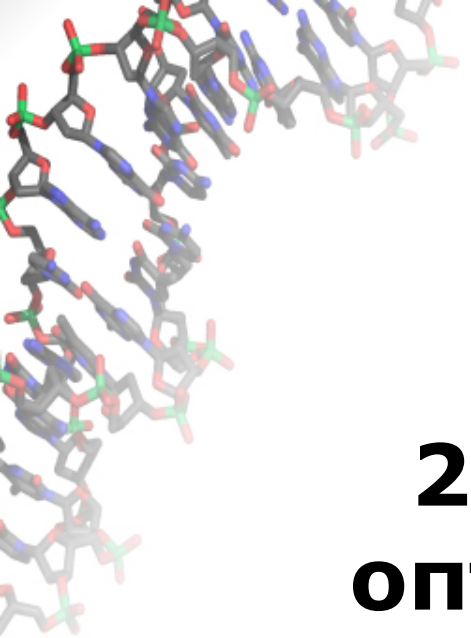
А.--ПОДБЕРЕЗОВИК
ПРЕДОСИНОВИЧКИ

Б.ПОДБЕРЕЗОВИК--
ПРЕДОСИНОВИЧКИ

В.ПО-ДБЕРЕЗОВИК--
ПРЕДОСИН-ОВИЧКИ

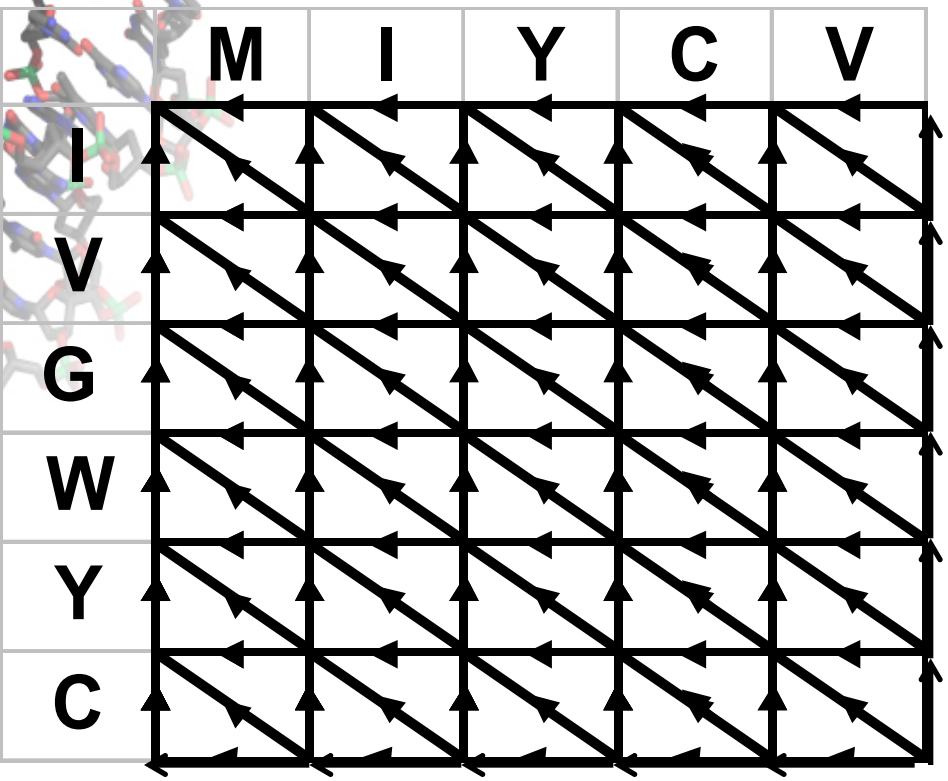
Г.ПО-ДБЕРЕЗОВИК--
ПРЕД-ОСИНОВИЧКИ

Д. ПО-ДБЕРЕЗОВИ-К-
ПРЕД-ОСИНОВИЧКИ



2.3. Алгоритм построения оптимального глобального выравнивания.

Граф



Веса ребер

a) диагонали

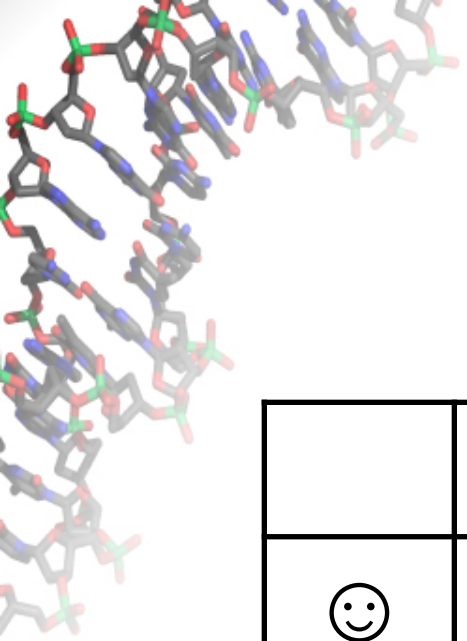
	M	I	Y	C	V
I	2	4	-1	-1	3
V	2	3	-1	0	3
G	-4	-3	-4	-2	-3
W	-1	-2	-2	-1	-3
Y	0	-1	-1	0	-1
C	-1	-1	-1	12	0

b) вертикали

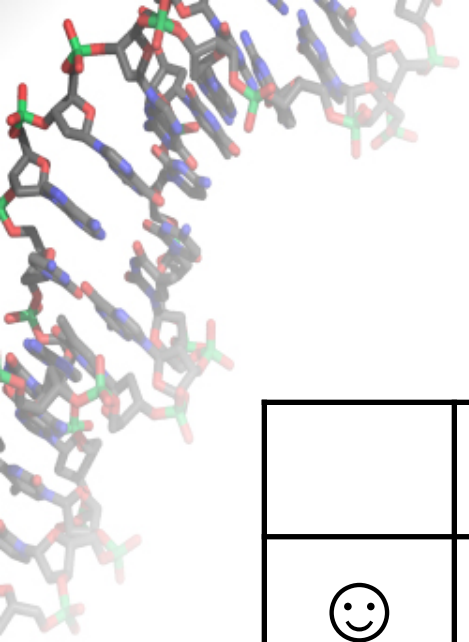
c) горизонтали

Порядок обхода вершин очевиден.

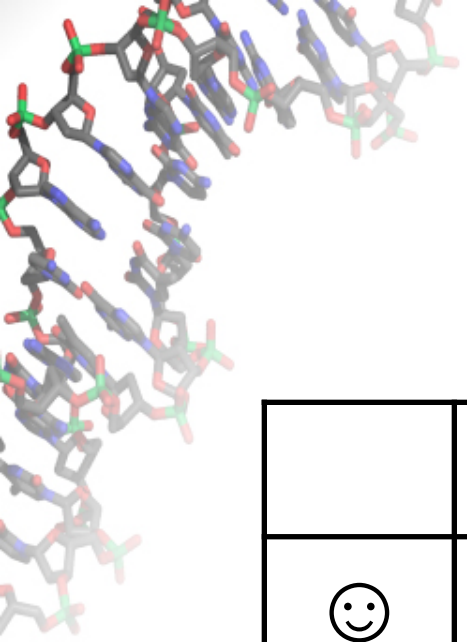
Вычислять ранги не нужно!



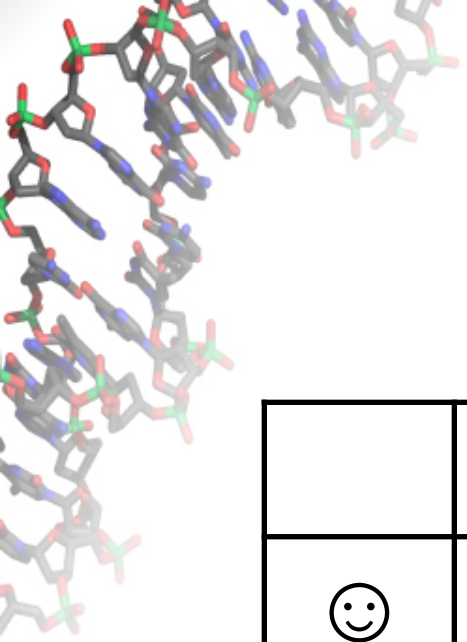
	☺	M	I	Y	C	V
☺	0	-1	-2	-3	-4	-5
I	-1	2	4	-1	-1	3
V	-2	2	3	-1	0	3
G	-3	-4	-3	-4	-2	-3
W	-4	-1	-2	-2	-1	-3
Y	-5	0	-1	8	0	-1
C	-6	-1	-1	-1	12	0



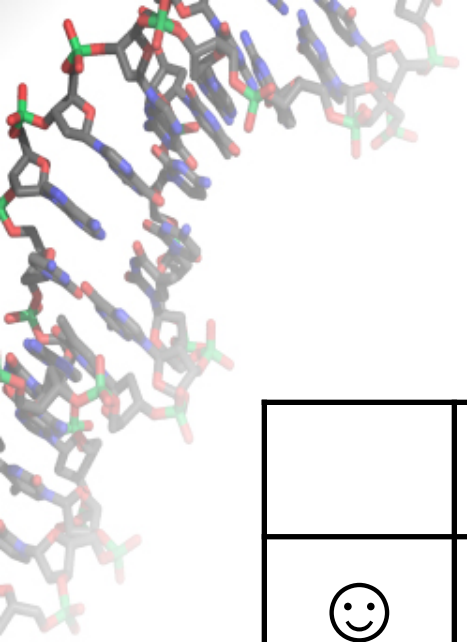
	☺	M	I	Y	C	V
☺	0	-1	-2	-3	-4	-5
I	-1	2	4-1=3	-1	-1	3
V	-2	2	3	-1	0	3
G	-3	-4	-3	-4	-2	-3
W	-4	-1	-2	-2	-1	-3
Y	-5	0	-1	8	0	-1
C	-6	-1	-1	-1	12	0



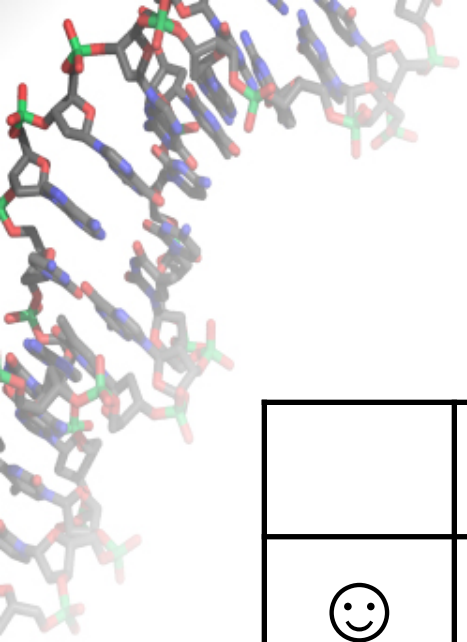
	☺	M	I	Y	C	V
☺	0	-1	-2	-3	-4	-5
I	-1	2	3	3-1=2	-1	3
V	-2	2	3	-1	0	3
G	-3	-4	-3	-4	-2	-3
W	-4	-1	-2	-2	-1	-3
Y	-5	0	-1	8	0	-1
C	-6	-1	-1	-1	12	0



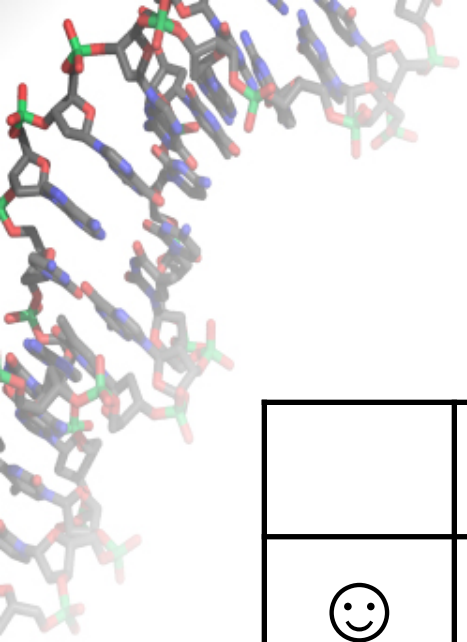
	☺	M	I	Y	C	V
☺	0	-1	-2	-3	-4	-5
I	-1	2	3	2	2-1=1	3
V	-2	2	3	-1	0	3
G	-3	-4	-3	-4	-2	-3
W	-4	-1	-2	-2	-1	-3
Y	-5	0	-1	8	0	-1
C	-6	-1	-1	-1	12	0



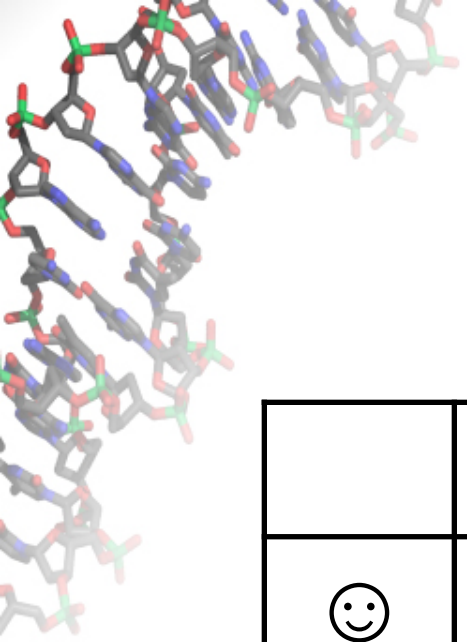
	☺	M	I	Y	C	V
☺	0	-1	-2	-3	-4	-5
I	-1	2	3	2	1	1-1=0
V	-2	2	3	-1	0	3
G	-3	-4	-3	-4	-2	-3
W	-4	-1	-2	-2	-1	-3
Y	-5	0	-1	8	0	-1
C	-6	-1	-1	-1	12	0



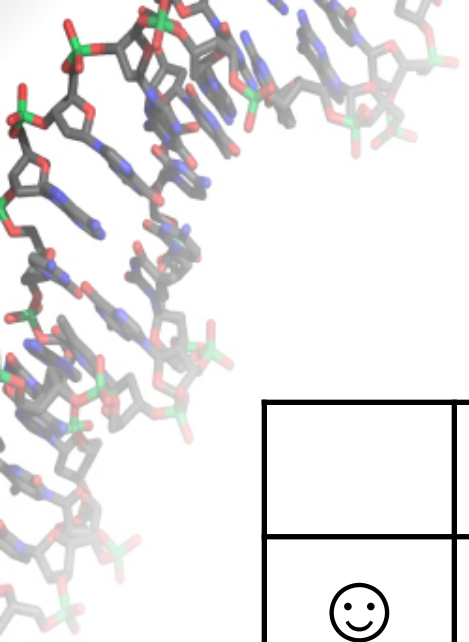
	☺	M	I	Y	C	V
☺	0	-1	-2	-3	-4	-5
I	-1	2	3	2	1	0
V	-2	1	5	4	3	4
G	-3	-4	-3	-4	-2	-3
W	-4	-1	-2	-2	-1	-3
Y	-5	0	-1	8	0	-1
C	-6	-1	-1	-1	12	0



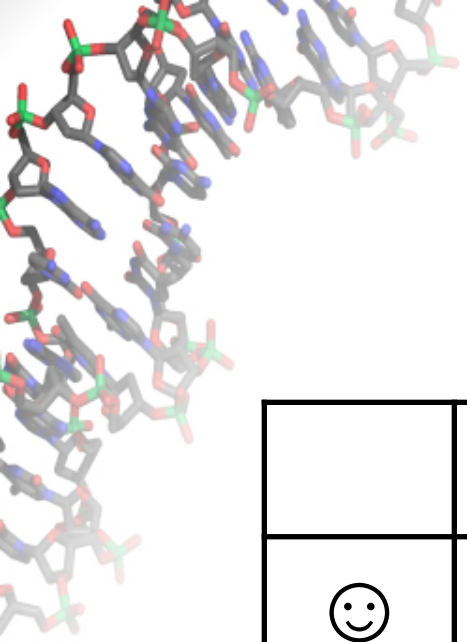
	☺	M	I	Y	C	V
☺	0	-1	-2	-3	-4	-5
I	-1	2	3	2	1	0
V	-2	1	5	4	3	4
G	-3	0	4	3	2	3
W	-4	-1	-2	-2	-1	-3
Y	-5	0	-1	8	0	-1
C	-6	-1	-1	-1	12	0



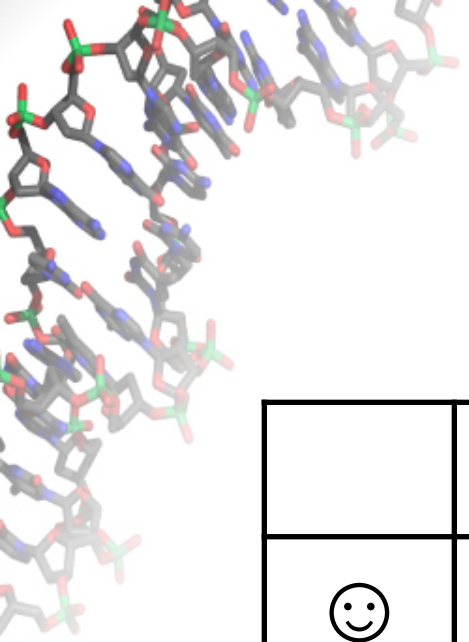
	☺	M	I	Y	C	V
☺	0	-1	-2	-3	-4	-5
I	-1	2	3	2	1	0
V	-2	1	5	4	3	4
G	-3	0	4	3	2	3
W	-4	-1	3	2	1	2
Y	-5	0	-1	8	0	-1
C	-6	-1	-1	-1	12	0



	☺	M	I	Y	C	V
☺	0	-1	-2	-3	-4	-5
I	-1	2	3	2	1	0
V	-2	1	5	4	3	4
G	-3	0	4	3	2	3
W	-4	-1	3	2	1	2
Y	-5	-2	2	11	10	9
C	-6	-1	-1	-1	12	0



	☺	M	I	Y	C	V
☺	0	-1	-2	-3	-4	-5
I	-1	2	3	2	1	0
V	-2	1	5	4	3	4
G	-3	0	4	3	2	3
W	-4	-1	3	2	1	2
Y	-5	-2	2	11	10	9
C	-6	-3	1	10	23	22



MI--YCV
IVGWYC-

	☺	M	I	Y	C	V
☺	0	-1	-2	-3	-4	-5
I	-1	2	3	2	1	0
V	-2	1	5	4	3	4
G	-3	0	4	3	2	3
W	-4	-1	3	2	1	2
Y	-5	-2	2	11	10	9
C	-6	-3	1	10	23	22

Построение оптимального глобального выравнивания
1. Постановка задачи

ВХОД: Алфавит A;

Последовательности: $v = v_1 \ldots v_n$; $w = w_1 \ldots w_m$ из A*

Матрица весов замен M: AxA -> **R**;

Штраф за удаление символа k

ВЫХОД: Выравнивание ***v*** и ***w*** (какое-то) имеющее наибольший возможный вес;

Вес этого выравнивания

ГРАФ:

Вершины {(i, j) | i=0,…, n; j = 0, …, m}

Ребра из (i, j)-> (1) (i-1, j); (2) (i, (j-1)); (3) (i-1, j-1)

(все координаты – неотрицательные)

Веса на ребрах, выходящих из вершины (i, j):

(i-1, j-1): M[v[i], w[j]]; (i, j-1): -k; (i-1, j): -k

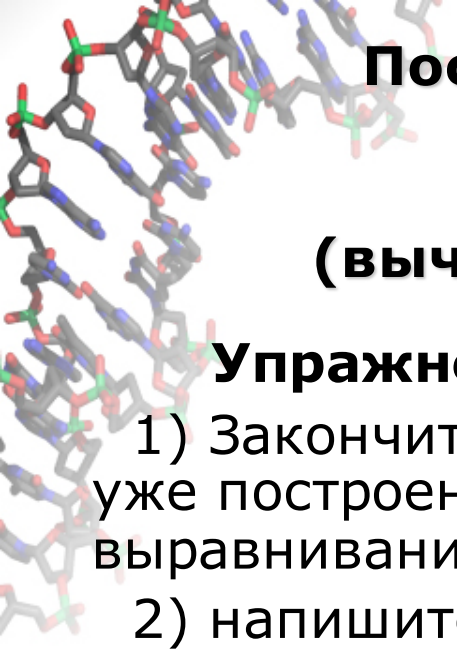
Построение оптимального глобального выравнивания
2. Данные

- **int** m, n;
  char array v[1..m], w[1..n];
- **float array** S[0..m, 0..n]; // веса оптимальных
  // выравниваний начал слов v, w
  int array D[0..m, 0..n] // направление лучшего пути
// 1 ≤ D[i, j] ≤ 7; в двоичной записи D[i, j]:
// 1-й бит соответствует удалению буквы v[i];
// 2-й бит соответствует удалению буквы w[j];
// 3-й бит соответствует сопоставлению v[i] и w[j];
- **int** NumberOfMatches; //кол-во сопоставлений в
  // выравнивании-результате
  // Позиции сопоставлений в v и w
  int array MatchPositionsV[1.. NumberOfMatches];
  int array MatchPositionsW[1.. NumberOfMatches];

Построение оптимального глобального выравнивания

3. Алгоритм: прямой проход (вычисление весов и локальных напрвлений)

```
int i, j; float xv, xw, xd; int p;
for j = 0 to n do begin
   S[0, j] := -k•j; D[0, j] := 2;
for i = 1 to m do begin
  S[0, i] := -k•i; D[i, 0] := 1;
  for j = 0 to n do begin
     xv := S[i-1, j]-k; xw:= S[i, j-1]-k;
     xd:=M[v[i], w[j]] + S[i-1, j-1];
     S[i, j] := max(xv, xw, xd);
     D[I, j] := argmax(xv, xw, xd);
  end
end
```



Построение оптимального глобального выравнивания
4. Алгоритм: обратный проход (вычисление оптимального выравниваия)

Упражнение 1.4.

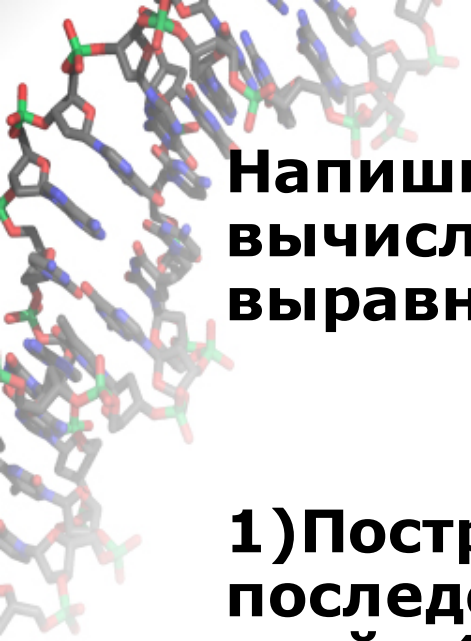
1) Закончите алгоритм - допишите под-алгоритм, который по уже построенным матрицам S и D строит какое-то оптимальное выравнивание исходных последовательностей.

2) напишите реализацию функции **argmax(xv, xw, xd);**

Упражнение 1.5.

1) скопируйте полный алгоритм в текстовый файл и дополните его комментариями;

2) реализуйте алгоритм в виде программы на каком-нибудь языке программирования;
допишите подпрограммы чтения входных данных и вывода результатов в текстовый файл; отладьте программу.



Упражнение 1.6.

Напишите алгоритм, который по матрице D вычисляет количество оптимальных выравниваний данных последовательностей.

Упражнение 1.7.

1)Постройте вручную матрицы S и D для последовательностей IVGWYC и MIYCV (см. слайд 43, там же указаны веса).

2) Нарисуйте какое-нибудь оптимальное выравнивание.

3) Сколько оптимальных выравниваний существует?

4) Измените веса так, чтобы для последовательностей было ровно одно оптимальное выравнивание.

К занятию 2

Задача о дугах

ДАНО: набор из *N* дуг, каждая из которых соединяет две целочисленные точки числовой оси *(x, y)*, где $1 \le x < y \le K$. Все дуги расположены над числовой осью.

***Правильная система* дуг – это такая система дуг, в которой никакие дуги не пересекаются.**

НАДО: найти в данном наборе правильную подсистему, содержащую максимально возможное количество дуг

[Рисунок на доске]