Motif Finding.
Greedy

ЖАДНЫЕ АЛГОРИТМЫ

- **Жадные алгоритмы выбирают «самую привлекательную» альтернативу на каждой итерации.**

  Пример: жадный алгоритм в шахматах может попытаться захватить самую ценную фигуру противника на каждом ходу.

- Жадные алгоритмы обычно не могут найти точного решения проблемы.

- Жадные алгоритмы часто бывают быстрыми эвристиками, которые используются для быстрого поиска приближенного решения.

ВСПОМОГАТЕЛЬНЫЕ МАТРИЦЫ

Motifs		T	C	G	G	G	G	g	T	T	T	t	t
		c	C	G	G	t	G	A	c	T	T	a	C
		a	C	G	G	G	G	A	T	T	T	t	C
		T	t	G	G	G	G	A	c	T	T	t	t
		a	a	G	G	G	G	A	c	T	T	C	C
		T	t	G	G	G	G	A	c	T	T	C	C
		T	C	G	G	G	G	A	T	T	c	a	t
		T	C	G	G	G	G	A	T	T	c	C	t
		T	a	G	G	G	G	A	a	c	T	a	C
		T	C	G	G	G	t	A	T	a	a	C	C

Score 3 + 4 + 0 + 0 + 1 + 1 + 1 + 5 + 2 + 3 + 6 + 4 = 30

Count	A:	2	2	0	0	0	0	9	1	1	1	3	0
	C:	1	6	0	0	0	0	0	4	1	2	4	6
	G:	0	0	10	10	9	9	1	0	0	0	0	0
	T:	7	2	0	0	1	1	0	5	8	7	3	4

Profile	A:	.2	.2	0	0	0	0	.9	.1	.1	.1	.3	0
	C:	.1	.6	0	0	0	0	0	.4	.1	.2	.4	.6
	G:	0	0	1	1	.9	.9	.1	0	0	0	0	0
	T:	.7	.2	0	0	.1	.1	0	.5	.8	.7	.3	.4

ЖАДНЫЙ ПОДХОД К ПОИСКУ МОТИВОВ

- Будем рассматривать матрицу ***Profile*** для заданного набора *k*-меров ***Motifs***.

Profile												
	A:	.2	.2	0	0	0	0	.9	.1	.1	.1	.3	0
	C:	.1	.6	0	0	0	0	0	.4	.1	.2	.4	.6
	G:	0	0	1	1	.9	.9	.1	0	0	0	0	0
	T:	.7	.2	0	0	.1	.1	0	.5	.8	.7	.3	.4

- Используя ***Profile*** определим вероятность *Pr*(String | *Profile*) для заданной String:

String	A	C	G	G	G	G	A	T	T	A	C	C	
Probability	.2	· .6	· 1	· 1	· .9	· .9	· .9	· .5	· .8	· .1	· .4	· .6	= 0.000839808

- k-мер имеет тенденцию иметь более высокую вероятность, когда он больше похож на консенсусную строку профиля.

  Например, для того же профиля и его консенсусной строки TCGGGGATTTCC:

Pr(TCGGGGATTTCC | *Profile*) = 0.7 · 0.6 · 1.0 · 1.0 · 0.9 · 0.9 · 0.9 · 0.5 · 0.8 · 0.7 · 0.4 · 0.6 = 0.0205753

PROFILE-MOST PROBABLE K-MER

Для заданной матрицы $\boldsymbol{Profile}$ можно оценить вероятность появления каждого k-мера в строке $\boldsymbol{Text}$ и найти наиболее вероятный k-мер в $\boldsymbol{Text}$, т.е. k-мер, который, скорее всего, был сгенерирован $\boldsymbol{Profile}$ среди все k-меров в $\boldsymbol{Text}$.

***Profile*-most Probable *k*-mer Problem:** *Find a Profile-most probable k-mer in a string.*

Input: A string *Text*, an integer *k*, and a 4 × *k* matrix *Profile*.

Output: A *Profile*-most probable *k*-mer in *Text*.

CODE CHALLENGE: Solve the *Profile*-most Probable *k*-mer Problem.

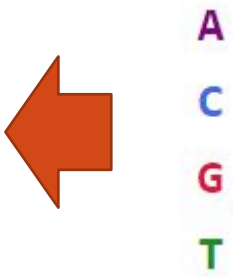
Debug Datasets

Sample Input:

```
CCCCTATAGTTCTTGGTGCAGCGTGCACCCTCGTCTGGTTCGGATACGGGCCTGCCAGGA
5
0.583 0.25 0.417 0.25 0.167
0.083 0.25 0.417 0.333 0.25
0 0.25 0.167 0 0.333
0.333 0.25 0 0.417 0.25
```

Sample Output:

```
ATACG
```



ЖАДНЫЙ ПОДХОД К ПОИСКУ МОТИВОВ

***GreedyMotifSearch*(*Dna, k, t*)**:

- Пробует каждый из *k*-меров в Dna_1 в качестве первого мотива.
- Для данного выбора *k*-мера $Motif_1$ в Dna_1 он создает матрицу профиля *Profile* для этого одиночного *k*-мера и устанавливает $Motif_2$ равным самому вероятному *k*-меру в Dna_2, посчитанному при помощи *Profile*.
- Затем он выполняет итерацию, обновляя *Profile* как матрицу профиля, образованную из $Motif_1$ и $Motif_2$, и устанавливает $Motif_3$ равным самому вероятному *k*-меру в Dna_3, посчитанному при помощи *Profile*.
- В общем случае, после обнаружения (*i-1*) *k*-меров *Motifs* в первых (*i-1*) строках *Dna*, *GreedyMotifSearch* конструирует *Profile*(*Motifs*) и выбирает наиболее вероятный *k*-мер из Dna_i на основе этой профильной матрицы.
- После получения *k*-мера из каждой строки для получения коллекции *Motifs*, *GreedyMotifSearch* проверяет, превосходит ли *Motifs* текущую коллекцию мотивов с лучшей оценкой, а затем перемещает $Motif_1$ на один символ в Dna_1, начиная снова весь процесс генерации *Motifs*.

ЖАДНЫЙ ПОДХОД К ПОИСКУ МОТИВОВ

```
GreedyMotifSearch(Dna, k, t)
    BestMotifs ← motif matrix formed by first k-mers in each string
                 from Dna
    for each k-mer Motif in the first string from Dna
        Motif_1 ← Motif
        for i = 2 to t
            form Profile from motifs Motif_1, …, Motif_i - 1
            Motif_i ← Profile-most probable k-mer in the i-th string
                      in Dna
        Motifs ← (Motif_1, …, Motif_t)
        if Score(Motifs) < Score(BestMotifs)
            BestMotifs ← Motifs
    return BestMotifs
```

GREEDY MOTIF SEARCH

CODE CHALLENGE: Implement ***GreedyMotifSearch***.

Input: Integers *k* and *t*, followed by a collection of strings *Dna*.

Output: A collection of strings *BestMotifs* resulting from applying ***GreedyMotifSearch(Dna,k,t)***. If at any step you find more than one *Profile*-most probable *k*-mer in a given string, use the one occurring first.

Sample Input:

```
5 5
TGGAGTCACTGGACGCGTAACGTTGATTTATCCCG
TCAGTAGGTGGCAACACGTAAAAAAGACCTAGGCG
ACATTCTAAATGCACGCTGTGATTTCGGGGACGCG
CCTTCATGGAGGGTGGGAGAATTGGGACTCCCACT
GCCGGGTCACACCCGGAAGGCGGCGCAGAGCTTGG
```

Sample Output:

```
GGACG
TCAGT
GCACG
GGAGG
GCCGG
```

НЕДОСТАТКИ ЖАДНОГО ПОДХОДА

Определение:
***k*-мер является (*k*, *d*)-мотивом для набора строк *DNA* и целого числа *d*, если он появляется в каждой строке *DNA* с не более чем *d* мутациями.**

GreedyMotifSearch может на первый взгляд показаться достаточно хорошим алгоритмом, однако, на самом деле это не так. Можно проверить, найдет ли *GreedyMotifSearch* (4,1)-мотив ACGT, имплантированный в строки *Dna*, показанные ниже.

```
ttACCTtaac
gATGTctgtc
acgGCGTtag
ccctaACGAg
cgtcagAGGT
```

Предположим, что алгоритм уже правильно выбрал имплантированный 4-мер ACCT из первой последовательности и построил соответствующий *Profile*:

```
A: 1 0 0 0
C: 0 1 1 0
G: 0 0 0 0
T: 0 0 0 1
```

НЕДОСТАТКИ ЖАДНОГО ПОДХОДА

```
A: 1 0 0 0
C: 0 1 1 0
G: 0 0 0 0
T: 0 0 0 1
```

Теперь алгоритм готов к поиску наиболее вероятного 4-мера в силу *Profile* во второй последовательности.

Проблема состоит в том, что в матрице *Profile* так много нулей, что вероятность каждого 4-мера, кроме ACCT, равна нулю. Таким образом, если ACCT не присутствует в каждой строке в *Dna*, мало шансов, что *GreedyMotifSearch* найдет имплантированный мотив.

Основная проблема – нули в матрице профиля

МОДИФИКАЦИЯ ЖАДНОГО ПОДХОДА

Предпосылки:

В любом наблюдаемом наборе данных существует вероятность, особенно для событий с низкой вероятностью или небольшими наборами данных, что событие с ненулевой вероятностью не происходит. Поэтому его наблюдаемая частота равна нулю. Однако установление эмпирической вероятности события равным нулю представляет собой неточное упрощение, которое может вызвать проблемы. Искусственно регулируя вероятность редких событий, эти проблемы можно смягчить.

МОДИФИКАЦИЯ ЖАДНОГО ПОДХОДА

Рассмотрим следующий *Profile:*

Profile	A:	.2	.2	0	0	0	0	.9	.1	.1	.1	.3	0
	C:	.1	.6	0	0	0	0	0	.4	.1	.2	.4	.6
	G:	0	0	1	1	.9	.9	.1	0	0	0	0	0
	T:	.7	.2	0	0	.1	.1	0	.5	.8	.7	.3	.4
String		T	C	G	T	G	G	A	T	T	T	C	C
Probability		.7	· .6	· 1	· 0	· .9	· .9	· .9	· .5	· .8	· .7	· .4	· .6 = 0

Четвертый символ **TCGTGGATTTCC** заставляет $Pr($**TCGTGGATTTCC** $|$ *Profile*$)$ равняться нулю.

В результате всей строке присваивается нулевая вероятность, хотя **TCGTGGATTTCC** отличается от консенсусной строки только в одной позиции.

В этом отношении **TCGTGGATTTCC** имеет такую же низкую вероятность, что и **AAATCTTGGAA**, которая сильно отличается от консенсусной строки.

LAPLACE'S RULE OF SUCCESSION

Добавим 1 к каждому элементу *Count*(*Motifs*).

Motifs				
	T	A	A	C
	G	T	C	T
	A	C	T	A
	A	G	G	T

Было

Count				
	A:	2	1	1	1
	C:	0	1	1	1
	G:	1	1	1	0
	T:	1	1	1	2

Profile				
	A:	2/4	1/4	1/4	1/4
	C:	0	1/4	1/4	1/4
	G:	1/4	1/4	1/4	0
	T:	1/4	1/4	1/4	2/4

Стало

Count				
	A:	2+1	1+1	1+1	1+1
	C:	0+1	1+1	1+1	1+1
	G:	1+1	1+1	1+1	0+1
	T:	1+1	1+1	1+1	2+1

Profile				
	A:	3/8	2/8	2/8	2/8
	C:	1/8	2/8	2/8	2/8
	G:	2/8	2/8	2/8	1/8
	T:	2/8	2/8	2/8	3/8

LAPLACE'S RULE OF SUCCESSION

```
GreedyMotifSearch(Dna, k, t)
    form a set of k-mers BestMotifs by selecting 1st k-mers in each string from Dna
    for each k-mer Motif in the first string from Dna
        Motif_1 ← Motif
        for i = 2 to t
            form Profile from motifs Motif_1, …, Motif_i - 1
            Motif_i ← Profile-most probable k-mer in the i-th string in Dna
        Motifs ← (Motif_1, …, Motif_t)
        if Score(Motifs) < Score(BestMotifs)
            BestMotifs ← Motifs
    output BestMotifs
```

Было

```
GreedyMotifSearch(Dna, k, t)
    form a set of k-mers BestMotifs by selecting 1st k-mers in each string from Dna
    for each k-mer Motif in the first string from Dna
        Motif_1 ← Motif
        for i = 2 to t
            apply Laplace's Rule of Succession to form Profile from motifs
                Motif_1, …, Motif_i - 1
            Motif_i ← Profile-most probable k-mer in the i-th string in Dna
        Motifs ← (Motif_1, …, Motif_t)
        if Score(Motifs) < Score(BestMotifs)
            BestMotifs ← Motifs
    output BestMotifs
```

Стало

LAPLACE'S RULE OF SUCCESSION

Определение:
k-мер является **(*k*, *d*)-мотивом** для набора строк *DNA* и целого числа *d*, если он появляется в каждой строке *DNA* с не более чем *d* мутациями.

Применим Laplace's rule of succession **для поиска (4,1)-мотива ACGT, имплантированного в следующие строки** *Dna*:

ttACCTtaac
gATGTctgtc
acgGCGTtag
ccctaACGAg
cgtcagAGGT

Предположим, что алгоритм уже правильно выбрал имплантированный 4-мер ACCT из первой последовательности. **Можно построить соответствующие матрицы оценок и профиля с помощью** Laplace's rule of succession:

Count				
	A:	1+1	0+1	0+1	0+1
	C:	0+1	1+1	1+1	0+1
	G:	0+1	0+1	0+1	0+1
	T:	0+1	0+1	0+1	1+1

Profile				
	A:	2/5	1/5	1/5	1/5
	C:	1/5	2/5	2/5	1/5
	G:	1/5	1/5	1/5	1/5
	T:	1/5	1/5	1/5	2/5

LAPLACE'S RULE OF SUCCESSION

Используем эту матрицу профиля для вычисления вероятностей всех 4-меров во **второй** строке из *Dna*:

```
Profile   A: 2/5 1/5 1/5 1/5
          C: 1/5 2/5 2/5 1/5
          G: 1/5 1/5 1/5 1/5
          T: 1/5 1/5 1/5 2/5
```

gATG	ATGT	TGTc	GTct	Tctg	ctgt	tgtc
$1/5^4$	$\mathbf{4/5^4}$	$1/5^4$	$\mathbf{4/5^4}$	$2/5^4$	$2/5^4$	$1/5^4$

В второй последовательности есть два наиболее вероятных 4-мера в силу *Profile* (**ATGT** и **GT**ct); предположим, что выбран имплантированный 4-мер **ATGT**.

Теперь есть следующие матрицы мотивов, оценок и профиля:

```
Motifs    A  C  C  T
          A  T  G  T
```

```
Count     A: 2+1 0+1 0+1 0+1
          C: 0+1 1+1 1+1 0+1
          G: 0+1 0+1 1+1 0+1
          T: 0+1 1+1 0+1 2+1
Profile   A: 3/6 1/6 1/6 1/6
          C: 1/6 2/6 2/6 1/6
          G: 1/6 1/6 2/6 1/6
          T: 1/6 2/6 1/6 3/6
```

LAPLACE'S RULE OF SUCCESSION

Используем эту матрицу профиля для вычисления вероятностей всех 4-меров в **третьей** строке из *D*:

acgG	cgGC	gGCG	GCGT	CGTt	GTta	Ttag
$12/6^4$	$2/6^4$	$2/6^4$	**$12/6^4$**	$3/6^4$	$2/6^4$	$2/6^4$

Profile				
	A:	3/6	1/6	1/6	1/6
	C:	1/6	2/6	2/6	1/6
	G:	1/6	1/6	2/6	1/6
	T:	1/6	2/6	1/6	3/6

Во третьей последовательности есть два наиболее вероятных 4-мера в силу *Profile* (acgG и GCGT) . На этот раз предположим, что acgG выбран вместо имплантированного 4-мера GCGT.

Теперь есть следующие матрицы мотивов, оценок и профиля:

Motifs				
	A	C	C	T
	A	T	G	T
	a	c	g	G

Count				
	A:	3+1	0+1	0+1	0+1
	C:	0+1	2+1	1+1	0+1
	G:	0+1	0+1	2+1	1+1
	T:	0+1	1+1	0+1	2+1

Profile				
	A:	4/7	1/7	1/7	1/7
	C:	1/7	3/7	2/7	1/7
	G:	1/7	1/7	3/7	2/7
	T:	1/7	2/7	1/7	3/7

LAPLACE'S RULE OF SUCCESSION

Используем эту матрицу профиля для вычисления вероятностей всех 4-меров в **четвертой** строке из *Dn*

ccct	ccta	ctaA	taAC	aACG	ACGA	CGAg
$18/7^4$	$3/7^4$	$2/7^4$	$1/7^4$	$16/7^4$	$\mathbf{36/7^4}$	$2/7^4$

```
Profile   A: 4/7 1/7 1/7 1/7
          C: 1/7 3/7 2/7 1/7
          G: 1/7 1/7 3/7 2/7
          T: 1/7 2/7 1/7 3/7
```

Несмотря на то, что был пропущен имплантированный 4-мер в третьей последовательности, теперь найдет имплантированный 4-мер в четвертой строке в *Dna* в качестве наиболее вероятного 4-мера ACGA в силу *Profile*.

Теперь есть следующие матрицы мотивов, оценок и профиля:

```
Motifs   A  C  C  T
         A  T  G  T
         a  c  g  G
         A  C  G  A
```

```
Count    A: 4+1 0+1 0+1 1+1
         C: 0+1 3+1 1+1 0+1
         G: 0+1 0+1 3+1 1+1
         T: 0+1 1+1 0+1 2+1
Profile  A: 5/8 1/8 1/8 2/8
         C: 1/8 4/8 2/8 1/8
         G: 1/8 1/8 4/8 2/8
         T: 1/8 2/8 1/8 3/8
```

LAPLACE'S RULE OF SUCCESSION

Используем эту матрицу профиля для вычисления вероятностей всех 4-меров в **пятой** строке из *Dna*:

cgtc	gtca	tcag	cagA	agAG	gAGG	AGGT
$1/8^4$	$8/8^4$	$8/8^4$	$8/8^4$	$10/8^4$	$8/8^4$	**$60/8^4$**

Profile				
	A:	5/8	1/8	1/8	2/8
	C:	1/8	4/8	2/8	1/8
	G:	1/8	1/8	4/8	2/8
	T:	1/8	2/8	1/8	3/8

Наиболее вероятный 4-мер в силу *Profile* в 5-й строке в *Dna* – AGGT, имплантированный 4-мер. В результате модифицированный алгоритм создал следующую матрицу мотивов, которая подразумевает правильную консенсусную строку ACGT:

Motifs				
	A	C	C	T
	A	T	G	T
	a	c	g	G
	A	C	G	A
	A	G	G	T

GREEDY MOTIF SEARCH WITH PSEUDOCOUNTS

CODE CHALLENGE: Implement ***GreedyMotifSearch*** with pseudocounts.

Input: Integers *k* and *t*, followed by a collection of strings *Dna*.

Output: A collection of strings *BestMotifs* resulting from applying ***GreedyMotifSearch(Dna,k,t)*** with pseudocounts. If at any step you find more than one *Profile*-most probable *k*-mer in a given string, use the one occurring first.

Debug Datasets

Sample Input:

```
5 5
GCCAGATCCAATGACGGCTCCGCCAGAATCTGACA
AAAGGCATAGTCGAATATGCCGACAGGTTTGAGTT
ACTTCGGTCGCGATACCCGCAAGGAGTACCCCGAG
CCTCGTAGGGGCGAACCTCAGAAGGCTACAAGACA
ATTGGTGATATTGAGCGGTGCCTGAGGATCATCCT
```

Sample Output:

```
ACGGC
AAGGC
AAGGA
AAGGC
GAGGA
```