Алгоритм поиска диссонансов временного ряда

Определения

Временной ряд (time series) T представляет собой хронологически упорядоченную последовательность вещественных значений $t_1, t_2, ..., t_N$, ассоциированных с отметками времени, где N – длина последовательности.

Подпоследовательность (subsequence) T_{im} временного ряда T представляет собой непрерывное подмножество T из m элементов, начиная с позиции i, т.е. $T_{im} = t_i, t_{i+1}, ..., t_{i+m-1}$, где $1 \leq i \leq N$ и $i + m \leq N$.

Расстояние (distance) между подпоследовательностями C и M представляет собой функцию, которая в качестве аргументов принимает C и M, и возвращает неотрицательное число R. Для подпоследовательностей функция расстояния является симметричной, т.е. $Dist(C, M) = Dist(M, C)$.

Пусть имеется временной ряд T, подпоследовательность C длины n, начинающаяся с позиции p, и подпоследовательность M длины n, начинающаяся с позиции q. Подпоследовательности C и M называются *несамопересекающимися (non-self match)*, если $\mid p - q \mid \geq n$.

Определения

Подпоследовательность D длины n, начинающаяся с позиции l называется *диссонансом* временного ряда T, если D находится на наибольшем расстоянии от ближайшей несамопересекающейся с D подпоследовательности, чем любые другие подпоследовательсти временного ряда, т.е. $\forall C \in T$, несамопересекающихся с D M_D и с C $M_C : min(Dist(D, M_D)) > min(Dist(C, M_C))$.

Подпоследовательность D длины n, начинающаяся с позиции p называется *K-м диссонансом временного ряда*, если D имеет K-е по размеру расстояние от ближайшей несамопересекающейся подпоследовательности, при этом не имея пересекающихся частей с i-ми диссонансами, начинающимися на позициях p_i, для всех $1 \leq i \leq K$, т.е. $\mid p - p_i \mid \geq n$.

Подходы к нахождению диссонансов:

- 2а – составить матрицу расстояний для всех подпоследовательностей (для self-match будут фиктивные расстояния). Затем найти максимум из минимумов расстояний.
- 2б – на основе последовательного алгоритма HOTSAX Кеога. На основе полученной эвристики выбирается оптимальный порядок перебора подпоследовательностей при поиске расстояния до ближайшего соседа. При этом находить расстояние до ближайшего соседа придется только для нескольких первых подпоследовательностей. Для оставшихся будет срабатывать условие «раннего выхода» из цикла.

Основной алгоритм

Входные данные

- T – временной ряд ($t_i, 1 \leq i \leq m$)
- m – длина ряда
- n – длина подпоследовательности
- C – множество подпоследовательностей

Результат работы

- bsf_loc – позиция начала диссонанса временного ряда
- bsf_dist – расстояние до ближайшего соседа подпоследовательности-диссонанса

Алгоритм

1. Подготовка: составить вектор C из всех подпоследовательностей временного ряда .
2. Вычислить матрицу расстояний D каждой подпоследовательности временного ряда с каждой. Расстояния между подпоследовательностями C_i и C_j вычисляются по формуле:

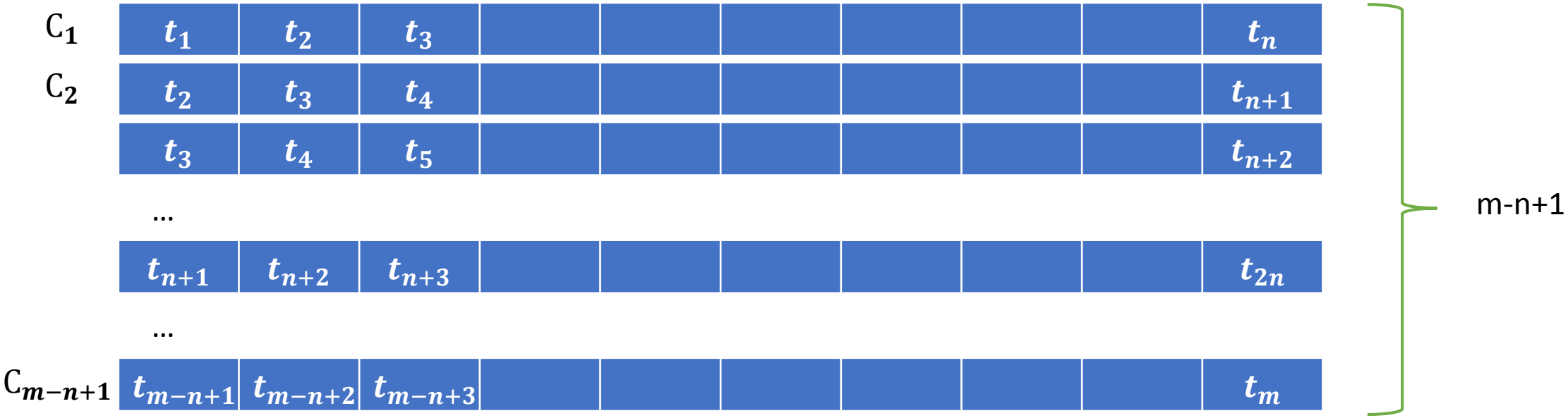
$$D_{i,j} = \sum_{k=1}^{n} (C_i^k - C_j^k)^2$$

3. В каждой i-й строке матрицы D $(1 \leq i \leq m - n + 1)$, соответствующей подпоследовательности C_i заменить расстояния до пересекающихся с C_i подпоследовательностей (self-match) на ∞
4. В каждой i-й строке матрицы D находим минимальный элемент $D_{i,j\ min}$, формируем вектор V_{min} из элементов $D_{i,j\ min}$.
5. Находим максимальный элемент в векторе V_{min} (позиция этого элемента будет соответствовать bsf_pos, а значение – bsf_dist.

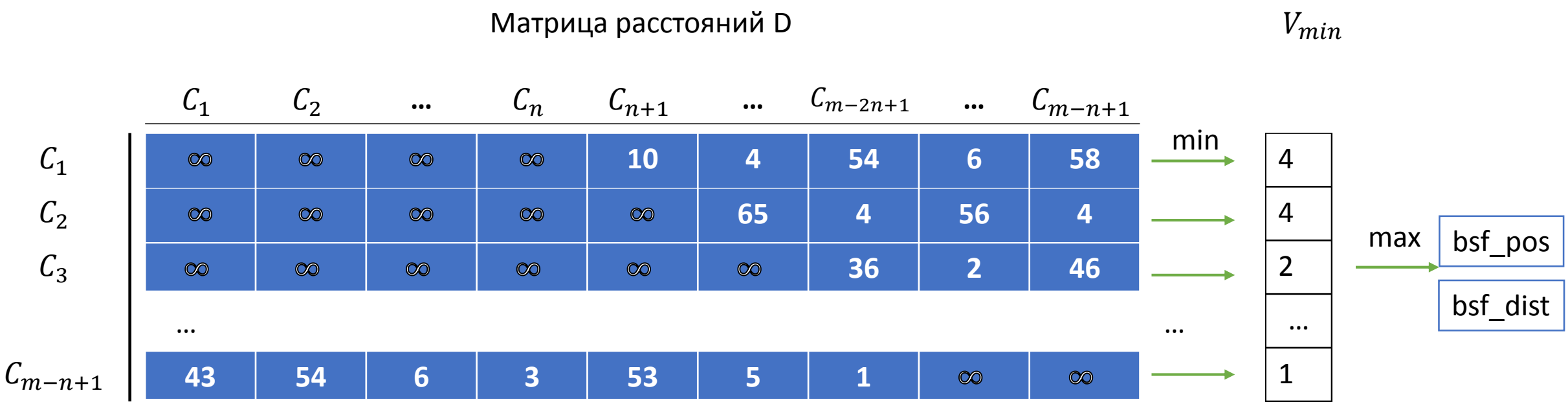
Хранение подпоследовательностей временного ряда

Временной ряд T:

Подпоследовательности C:



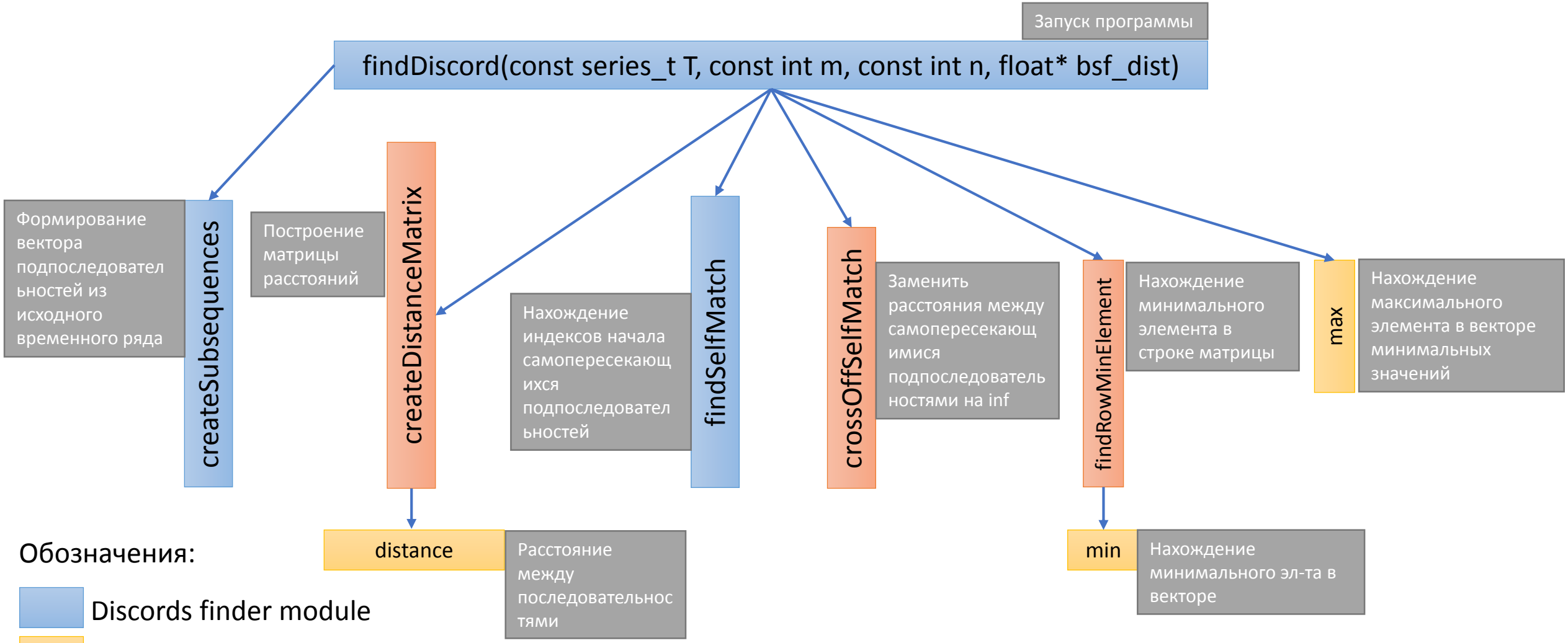
Поиск диссонансов: построение матрицы расстояний и нахождение самого далекого из ближайших соседей для каждой из подпоследовательностей



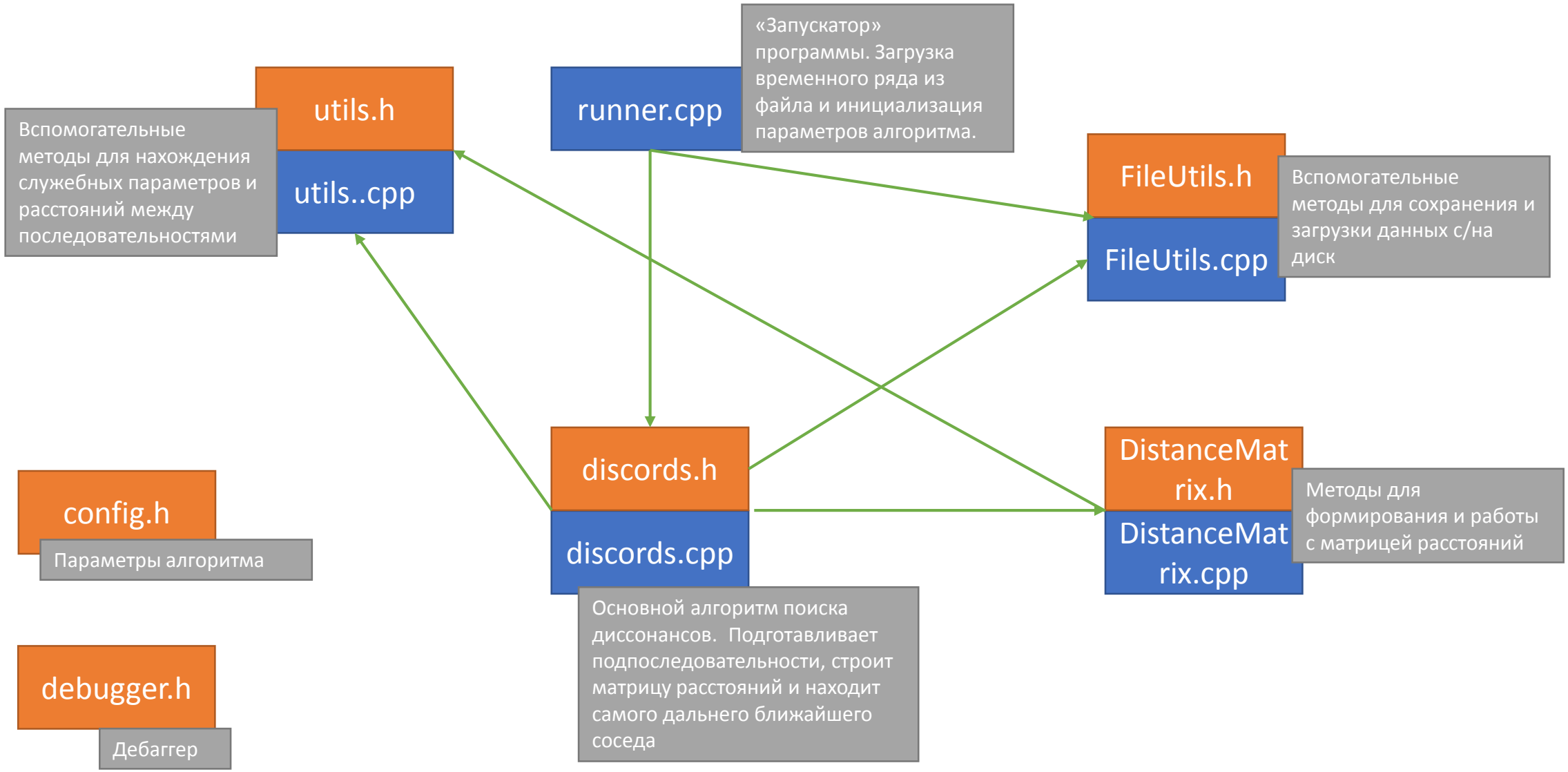
$$D_{C_i,C_j} = \text{Euclid\_dist}(C_i, C_j) = \sum_{k=1}^{n} (C_i^k - C_j^k)^2 \,, 1 \le j \le m \; - n + 1$$

∞ – фиктивное расстояние (между подпоследовательностью и самой собой и между подпоследовательностью и $self - match$ подпоследовательностями)

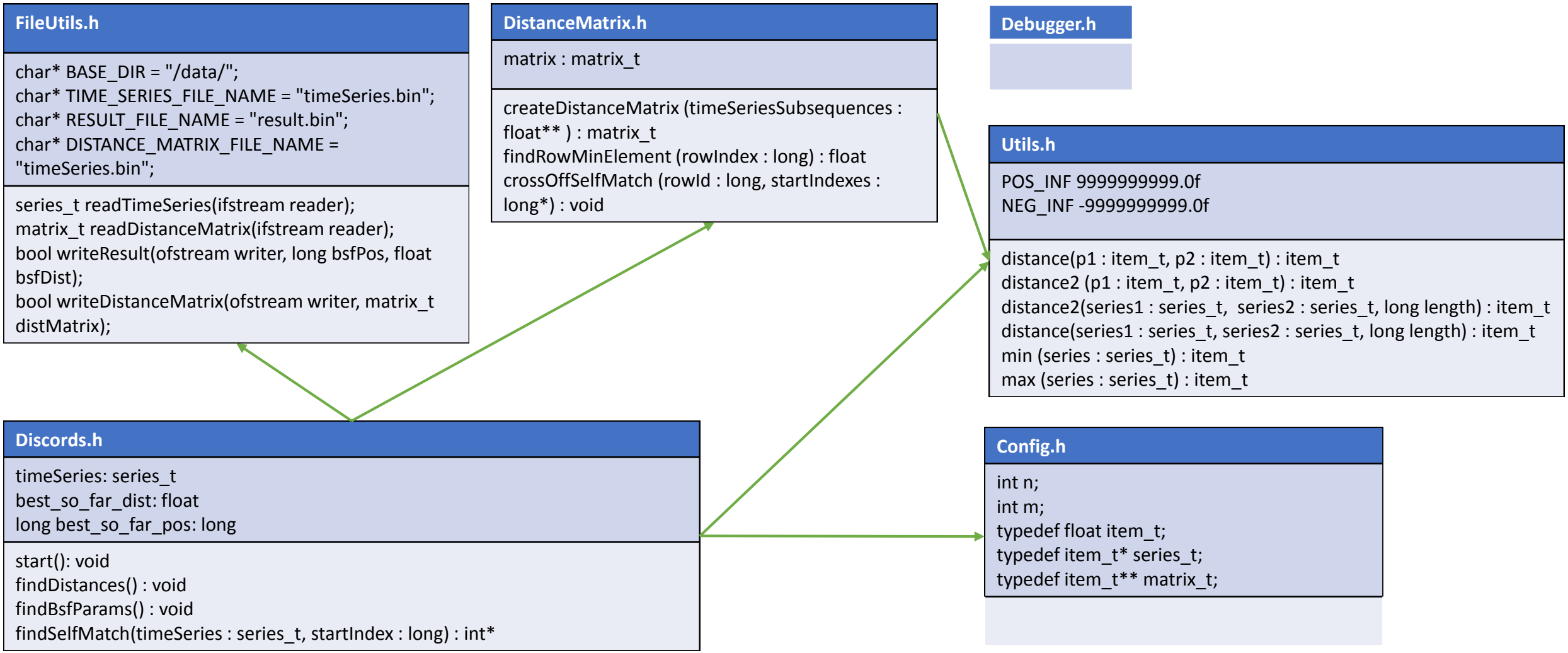
Модульная структура



Файловая структура



Файловая структура



Ограничения алгоритма

Размер временного ряда ~ 50 000 элементов типа float при объеме памяти ускорителя 8gb

Длина ряда	Размер элемента	Длина подпоследовательности	К-во подпоследовательностей	Эл-тов в м-це расстояний	Эл-тов в м-це подпоследовательностей	Размер памяти, Гб
1 000 000	4	128	999 873	999 746 016 129	127 983 744	3 999,50
500 000	4	1024	498 977	248 978 046 529	510 952 448	997,96
45 000	4	1024	43 977	1 933 976 529	45 032 448	7,92
45 000	4	128	44 873	2 013 586 129	5 743 744	8,08
43 200	4	7	43 194	1 865 721 636	302 358	7,46
175 000	4	128	174 873	30 580 566 129	22 383 744	122,41
175 000	4	1024	173 977	30 267 996 529	178 152 448	121,78

Тестовые данные

BruteForce		
DataSet	Кол-во элементов	Время работы
HotSAX		
DataSet	Кол-во элементов	Время работы
Последовательный с матрицей расстояний		
DataSet	Кол-во элементов	Время работы

Old: последовательный алгоритм на основании алгоритма HOTSAX Кеога (Keogh)

Входные данные

- T – временной ряд ($t_i, 1 \leq i \leq m$)
- m – длина ряда
- n – длина подпоследовательности
- C – множество подпоследовательностей
- w – длина слова (в SAX аппроксимации подпоследовательностей), $1 \leq w \leq n, n \; mod \; w = 0$
- Lookup table (LT) – для точек разделения в SAX
- A – мощность алфавита для SAX представления подпоследовательностей

б.) Подбор эвристики для цикла с ранним выходом из итераций. Алгоритм:

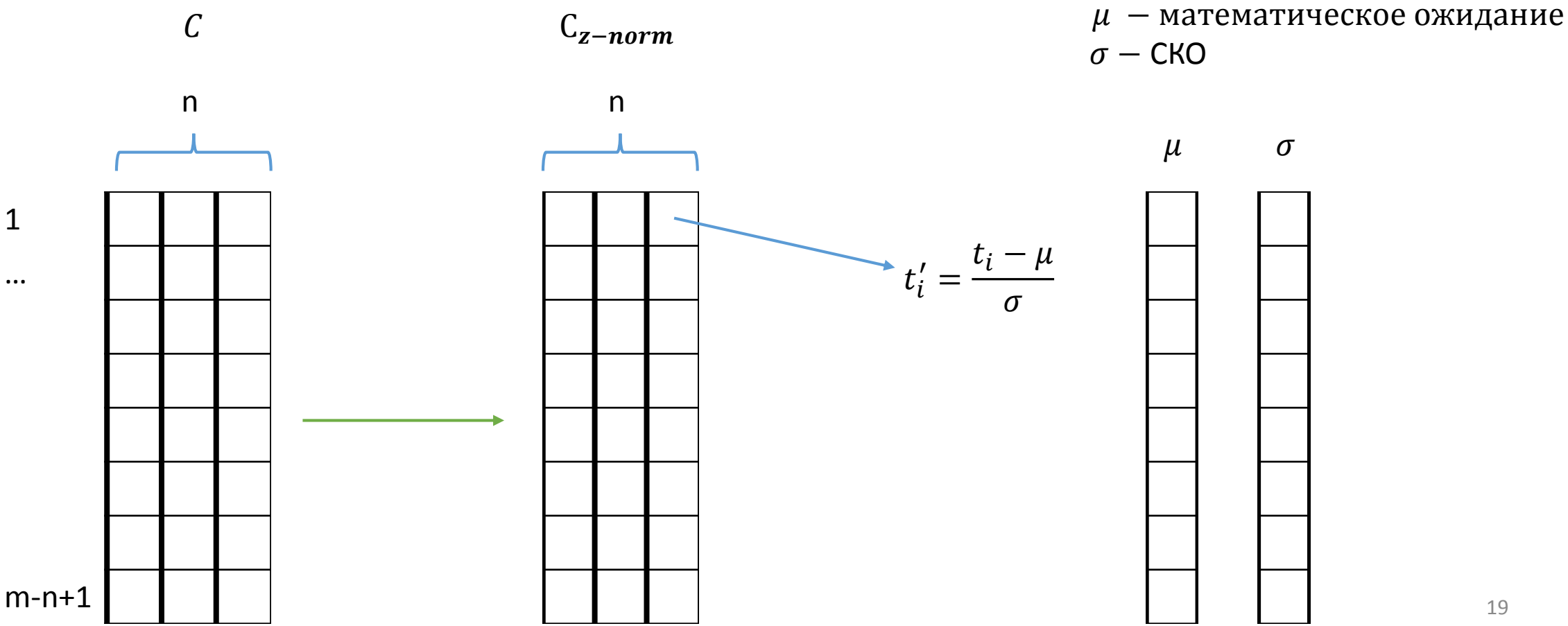
1. Подготовка (выбор эвристики) – подбор порядка подачи подпоследовательностей, при котором возможно быстро отбрасывать неподходящие подпоследовательности.
2. Поиск диссонансов – перебор упорядоченных подпоследовательностей временного ряда с поиском наибольшего расстояния до ближайшего соседа

Алгоритм:

1. Подготовка (выбор эвристики) – подбор порядка подачи подпоследовательностей, при котором возможно быстро отбрасывать неподходящие подпоследовательности.
    1. Z-нормализация подпоследовательностей C_i временного ряда
    2. Кусочная аппроксимация (PAA-представление)
    3. Кодирование с помощью lookup table
    4. Подсчет частот, нахождение мин. значения
2. Поиск диссонансов – перебор упорядоченных подпоследовательностей временного ряда с поиском наибольшего расстояния до ближайшего соседа, с постепенным обновлением best_so_far_dist и «быстрым» выходом из итераций цикла.

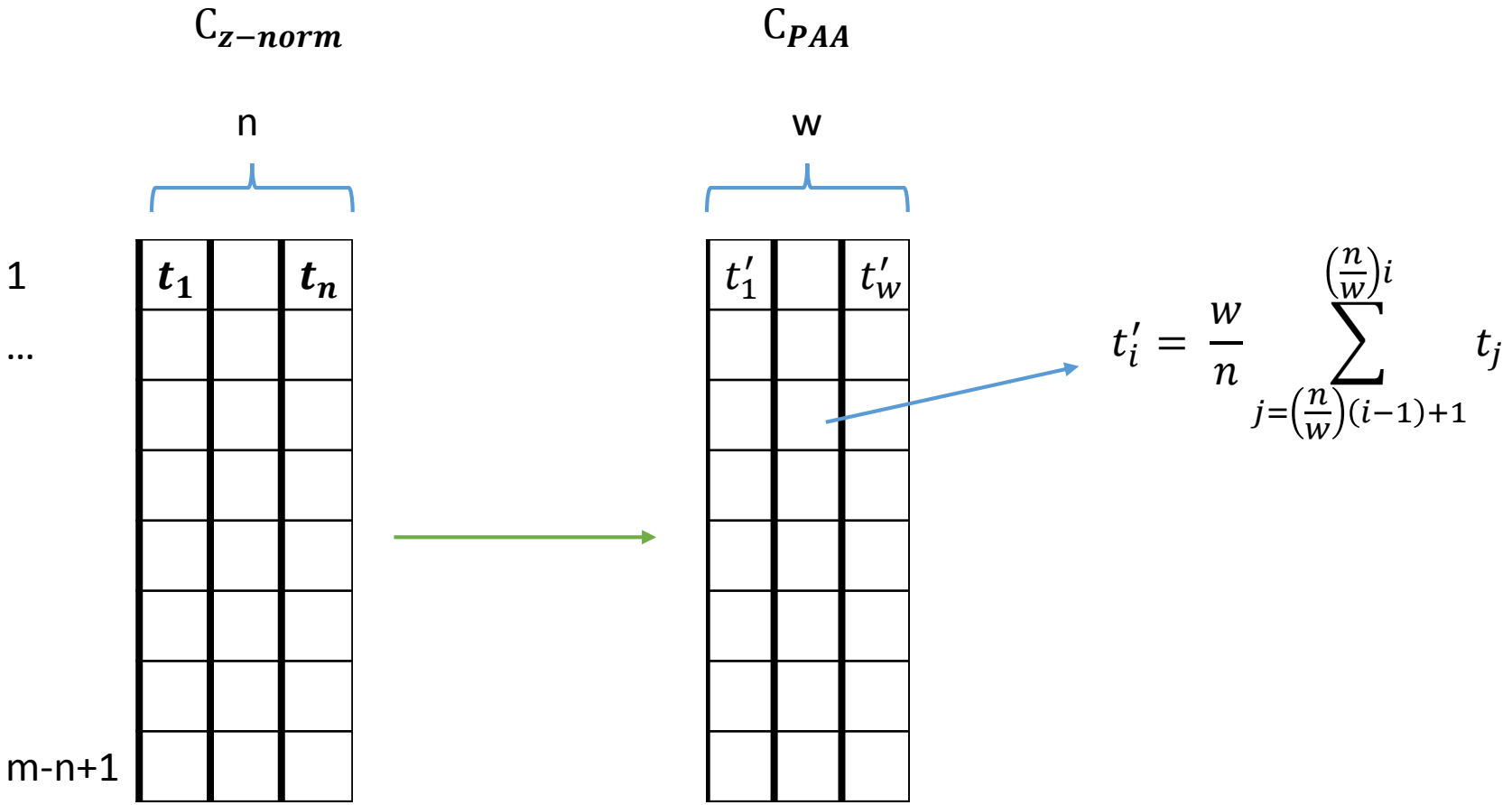
1. Подбор эвристики

1.1 z-нормализация подпоследовательностей C_i временного ряда



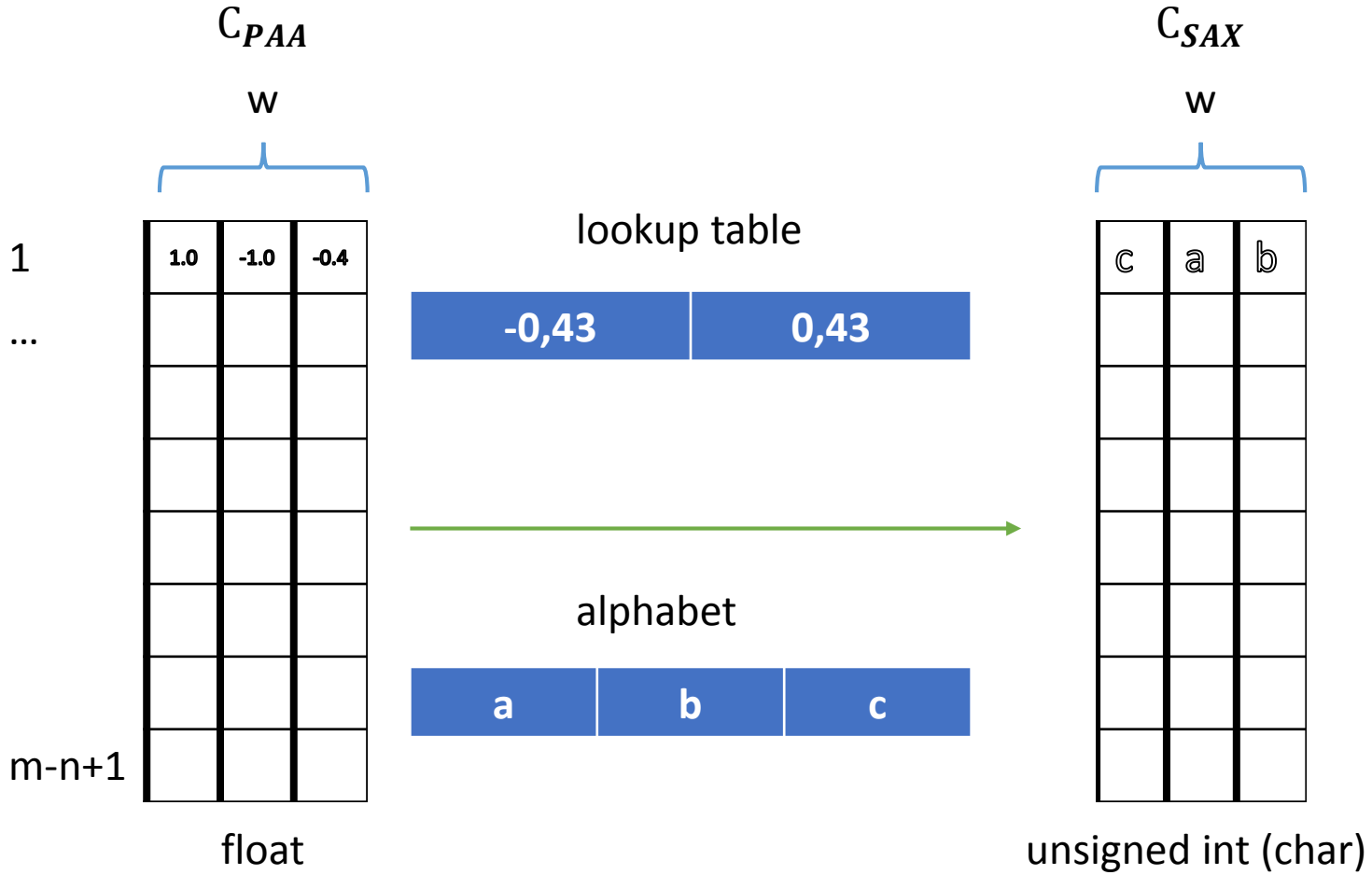
1. Подбор эвристики

1.2 Аппроксимация с помощью кусочной агрегации (PAA)



1. Подбор эвристики

1.3 Кодирование с помощью lookup table (аппроксимация с помощью символьной агрегации)

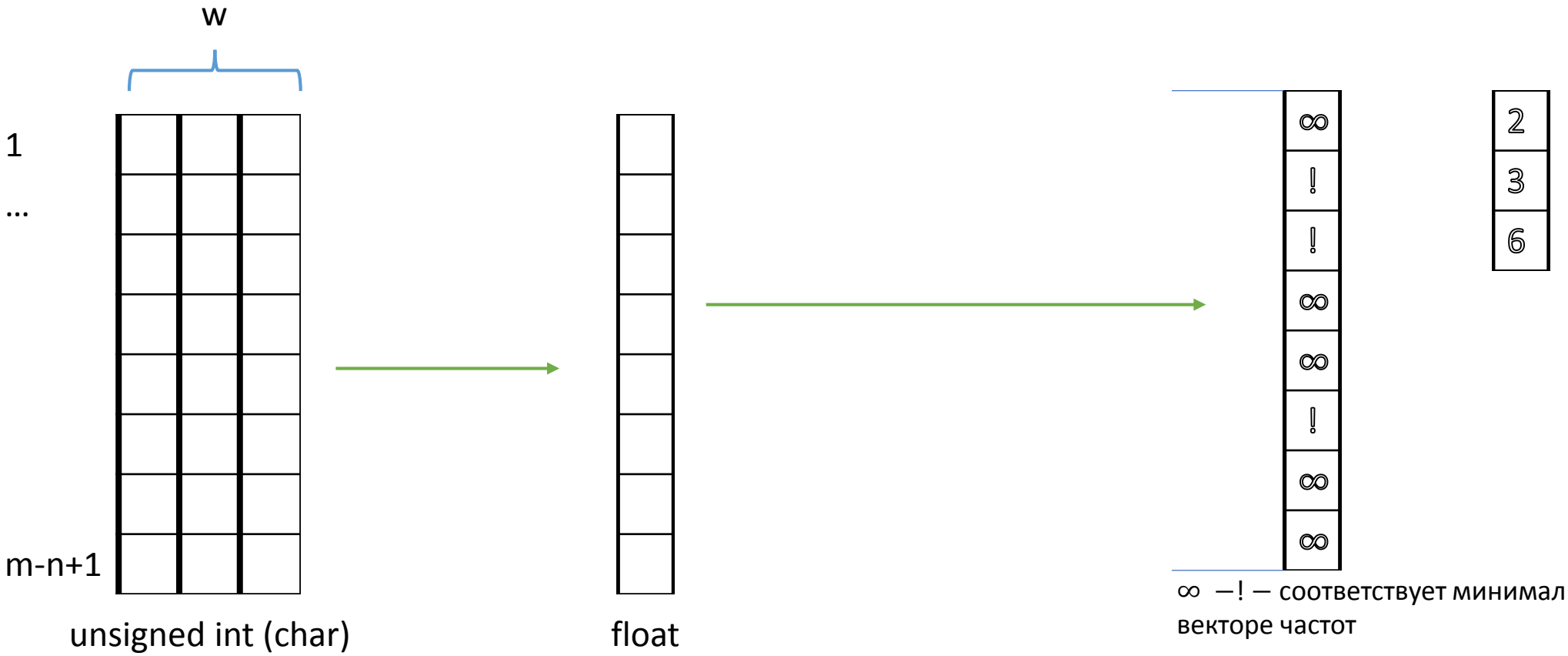


1. Подбор эвристики

1.4 Подсчет частот, нахождение мин. значения

C_{SAX}

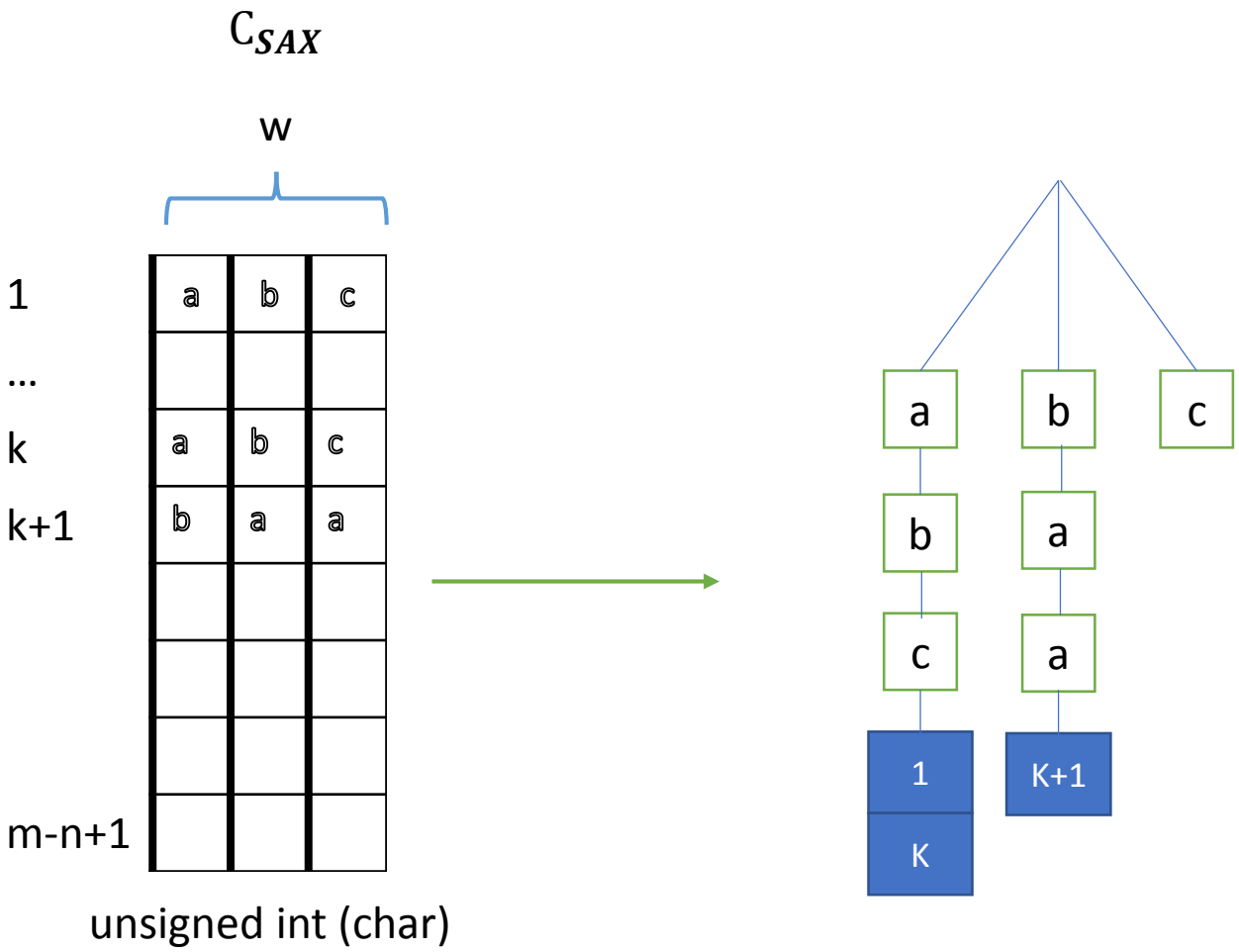
Frequences – частота вхождения каждого из слов в SAX представлении



∞ —! — соответствует минимальному значению в векторе частот

1. Подбор эвристики

1.5 Построение префиксного дерева

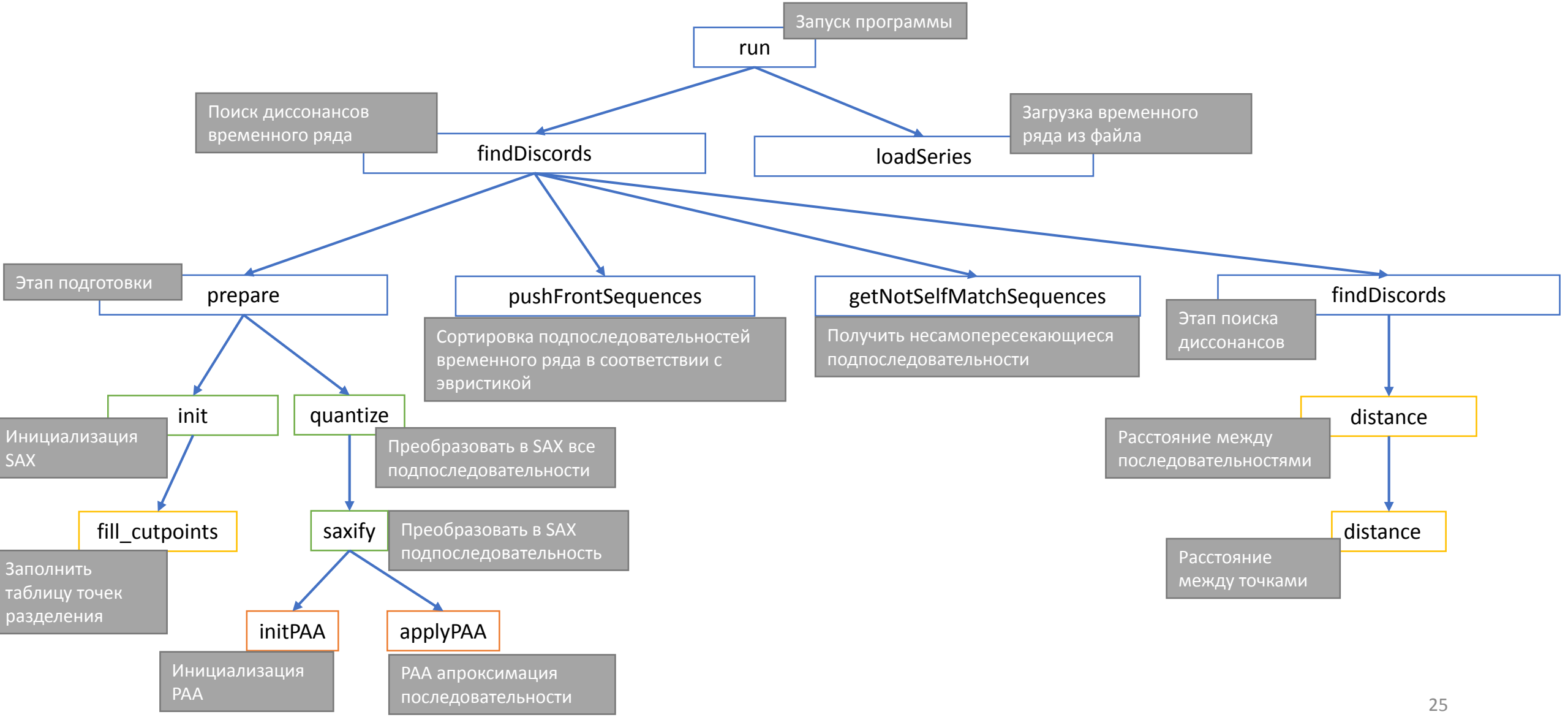


2. Поиск диссонансов: на основе эвристики с ранним выходом из итерации цикла

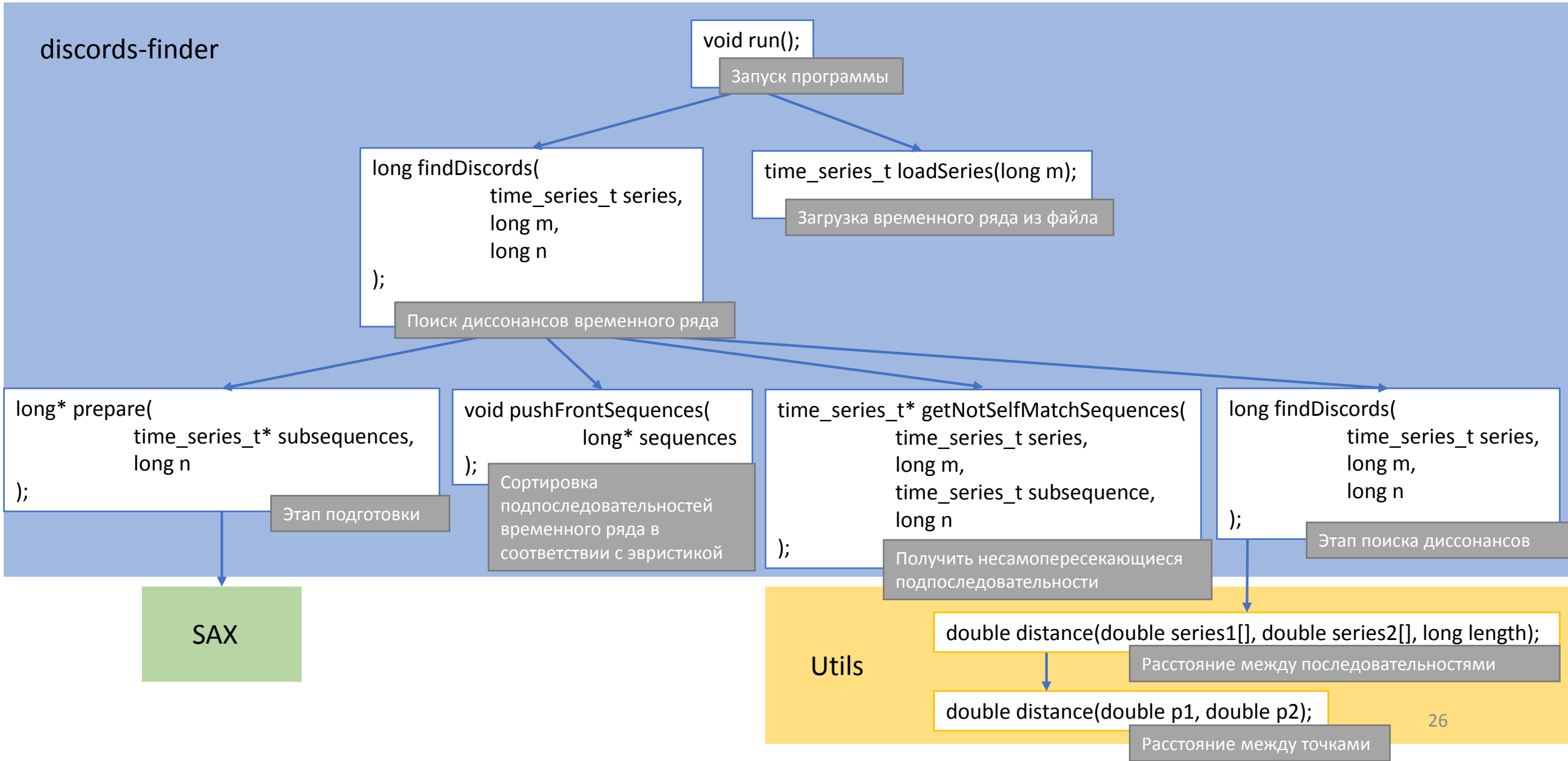
```
For(int p = 1; p< m-n+1; p++) {
        min = ∞;
        For(int j = 0; j < m-n+1; j++) {
                if(is_self_match(C_p, C_j) {
                        continue;
                }
                dist_j = Euclid_dist(C_p, C_j)
                if (dist_j < best_so_far_dist) {
                        -> C_p не подходит
                        break;
                }
                if (dist_j < min) {
                        min = dist_j;
                }
        }
        If(min > best_so_far_dist) {
                best_so_far_dist = min;
                best_so_far_pos = p;
        }
}
```

Output: диссонансом является подпоследовательность начинающаяся с best_so_far_pos позиции в исходном временном ряде.

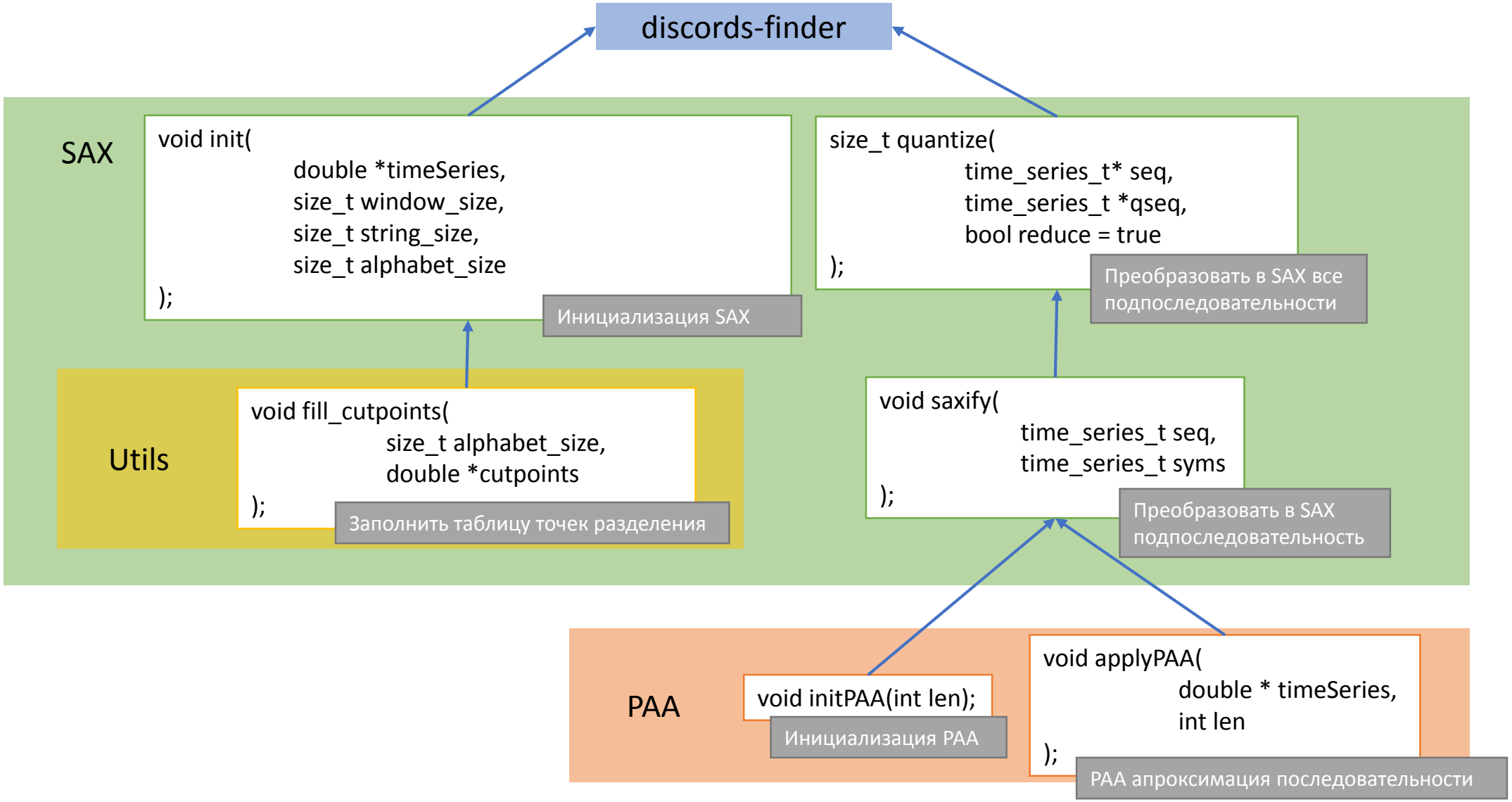
Модульная структура



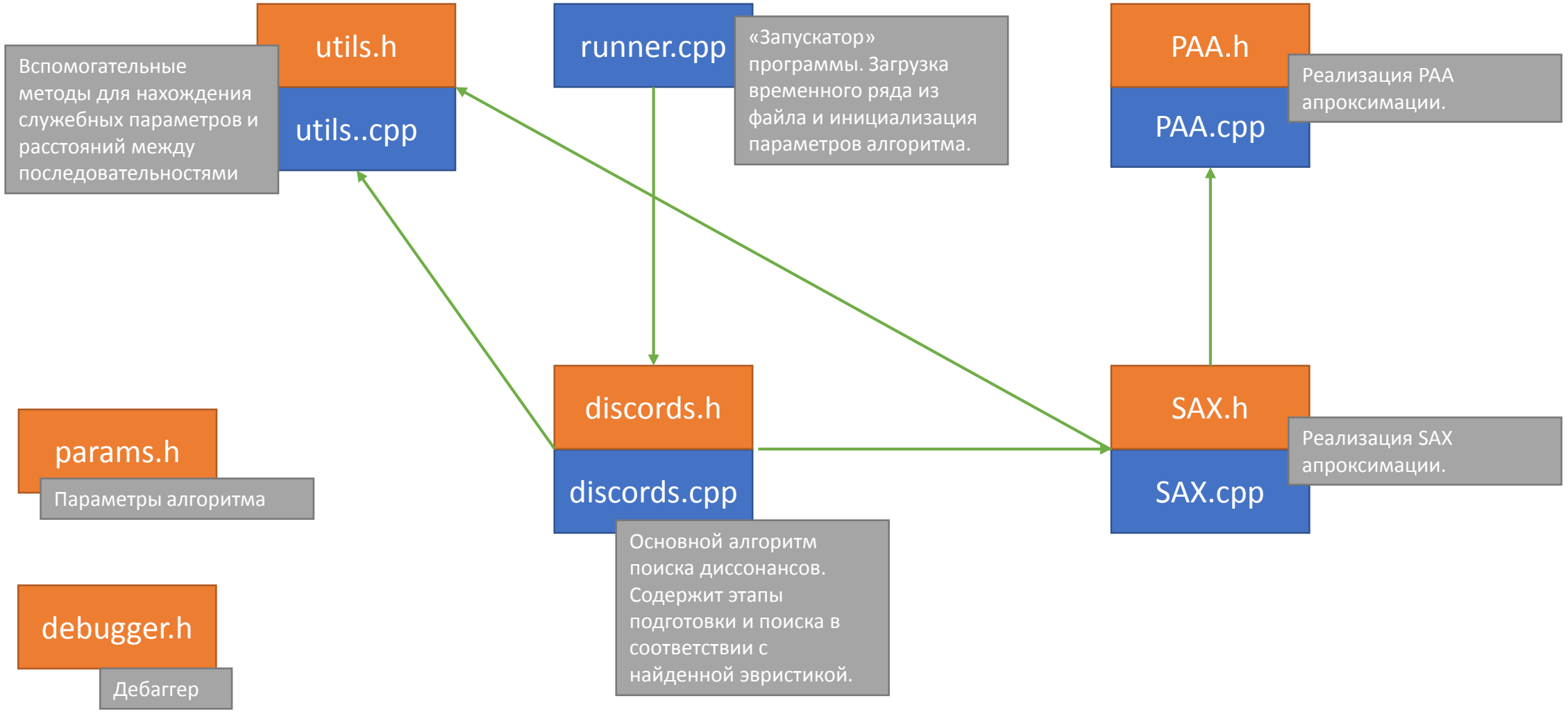
Модульная структура



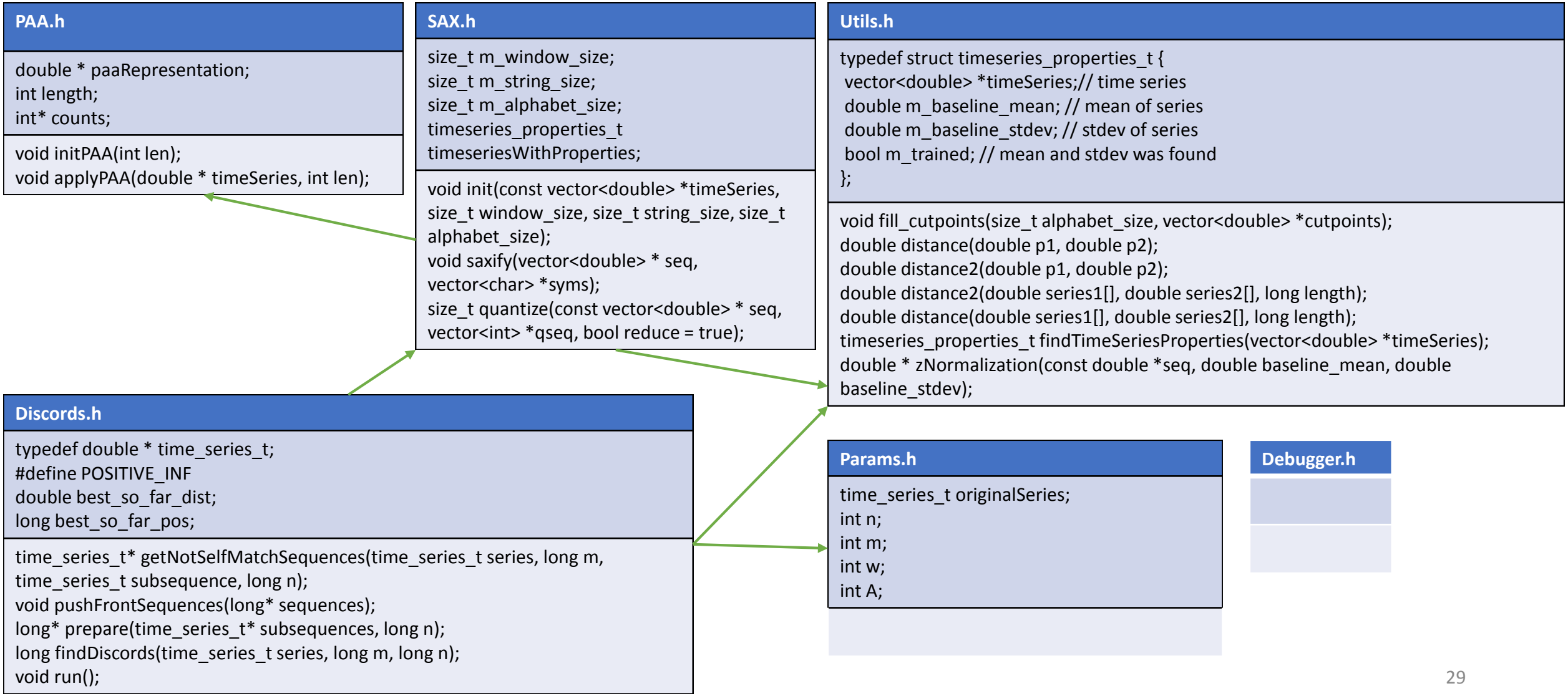
Модульная структура



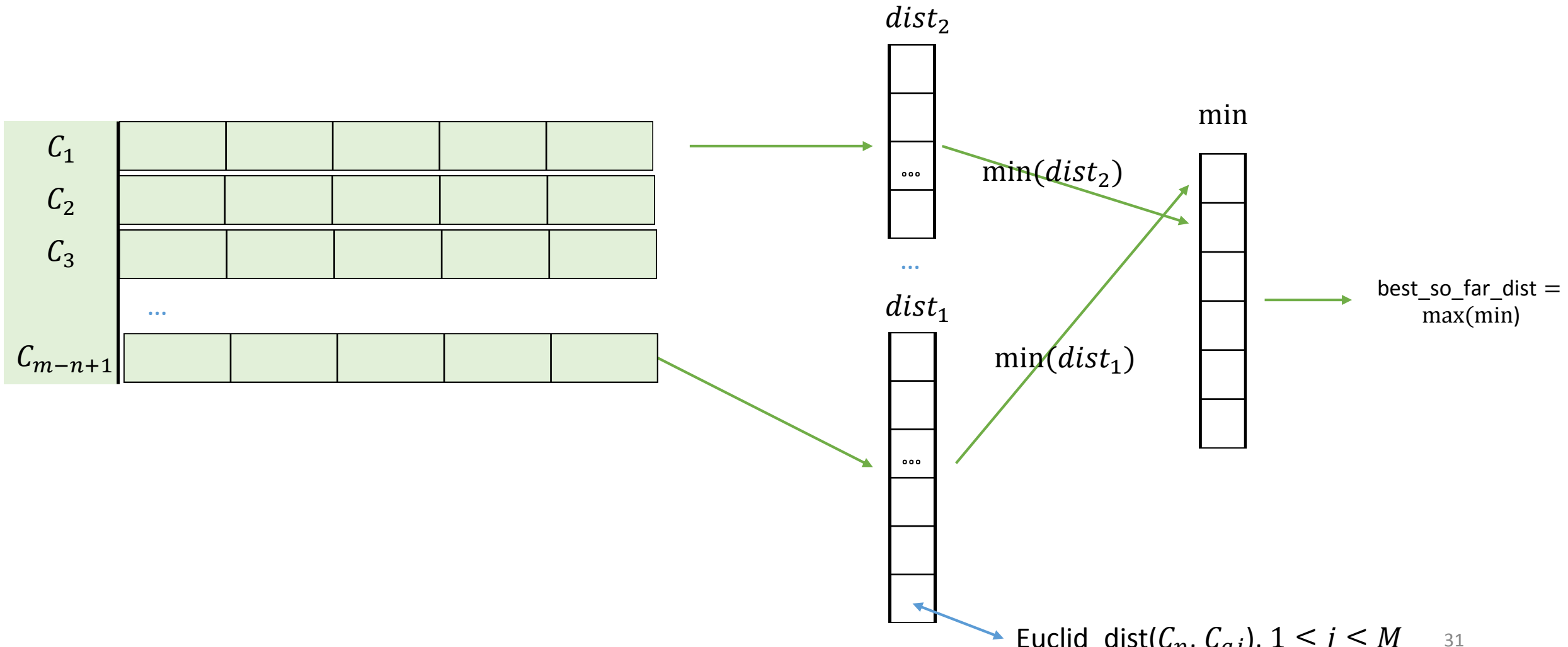
Файловая структура



Файловая структура



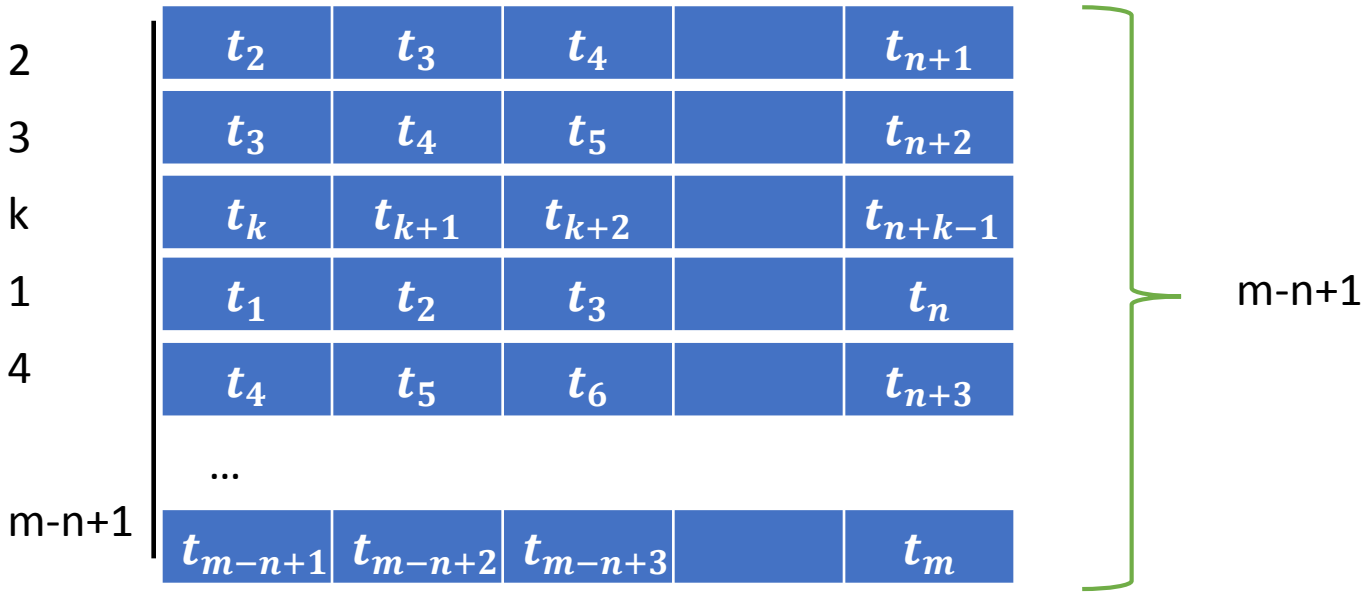
2. Поиск диссонансов: на основе эвристики с ранним выходом из итерации цикла



Упорядочивание множества подпоследовательностей

Index minVal

2
3
k



2. Поиск диссонансов: на основе эвристики с ранним выходом из итерации цикла

Формируем матрицы (n × (1 + nsm_count$_p$) для каждой подпоследовательности C_p, $1 \le p \le m - n + 1$, где nsm_count$_p$ - количество несамопересекающихся с C_p подпоследовательностей.