

Алгоритм поиска диссонансов временного ряда

Оглавление

- Определения
- Входные данные и результат работы
- Алгоритм
 - Подбор эвристики
 - z-нормализация подпоследовательностей C_i временного ряда
 - Аппроксимация с помощью кусочной агрегации (РАА)
 - Кодирование с помощью lookup table (аппроксимация с помощью символьной агрегации)
 - Подсчет частот, нахождение мин. значения
 - Построение цепочек с индексами начала подпоследовательностей, имеющих одинаковое SAX представление
 - Поиск диссонансов: на основе эвристики с ранним выходом из итерации цикла
- Модульная структура
- Файловая структура

Определения

Временной ряд (time series) T представляет собой хронологически упорядоченную последовательность вещественных значений t_1, t_2, \dots, 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. $T_{im} = t_i, t_{i+1}, \dots, t_{i+m-1}$, 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 $|p - q| \geq n$.

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 подпоследовательности временного ряда, т.е. $\forall C \in T$, несамопересекающихся с D M_D и с C $M_C : \min(\text{Dist}(D, M_D)) > \min(\text{Dist}(C, M_C))$.

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. $|p - p_i| \geq n$.

Подходы к нахождению диссонансов:

- составить матрицу расстояний для всех подпоследовательностей (для self-match будут фиктивные расстояния). Затем найти максимум из минимумов расстояний.
- 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.

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),
 $1 \leq w \leq n, n \bmod w = 0$
- Lookup table (LT) – для точек разделения в SAX
- A – мощность алфавита для SAX представления подпоследовательностей

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

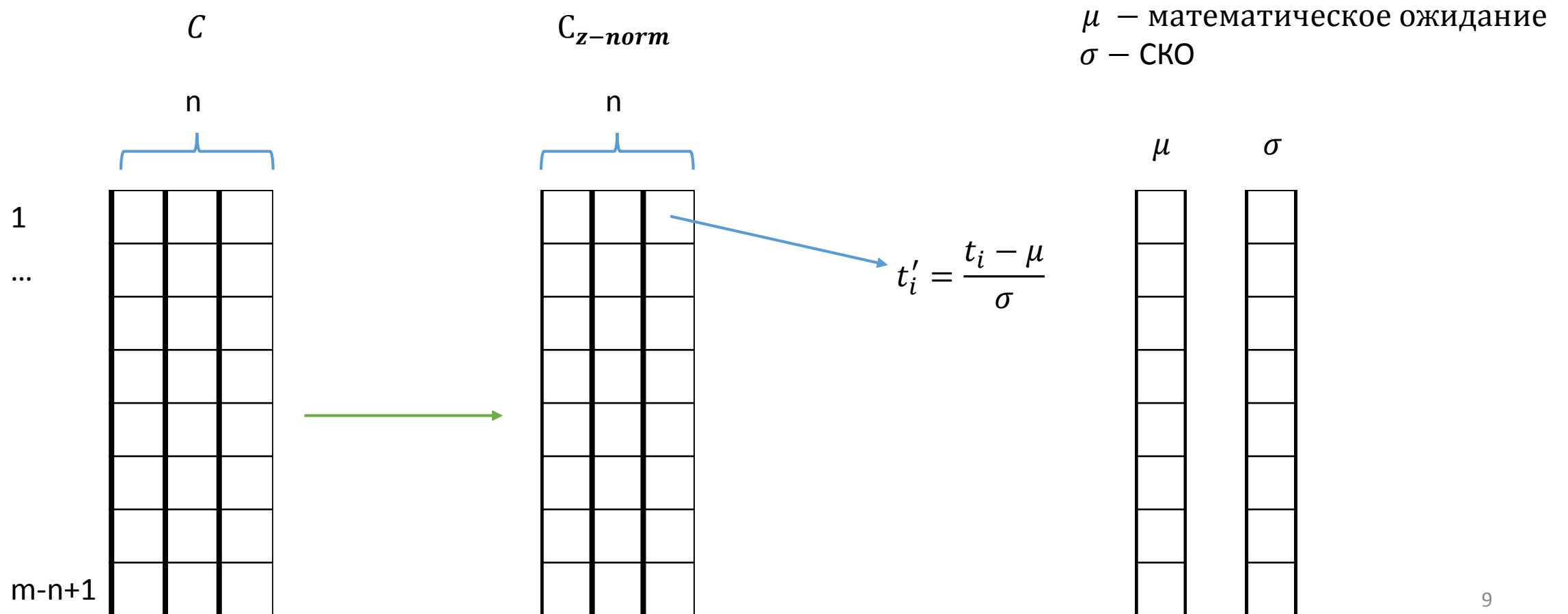
1. Подготовка (выбор эвристики) – подбор порядка подачи подпоследовательностей, при котором возможно быстро отбрасывать неподходящие подпоследовательности.
2. Поиск диссонансов – перебор упорядоченных подпоследовательностей временного ряда с поиском наибольшего расстояния до ближайшего соседа

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 (РАА-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.

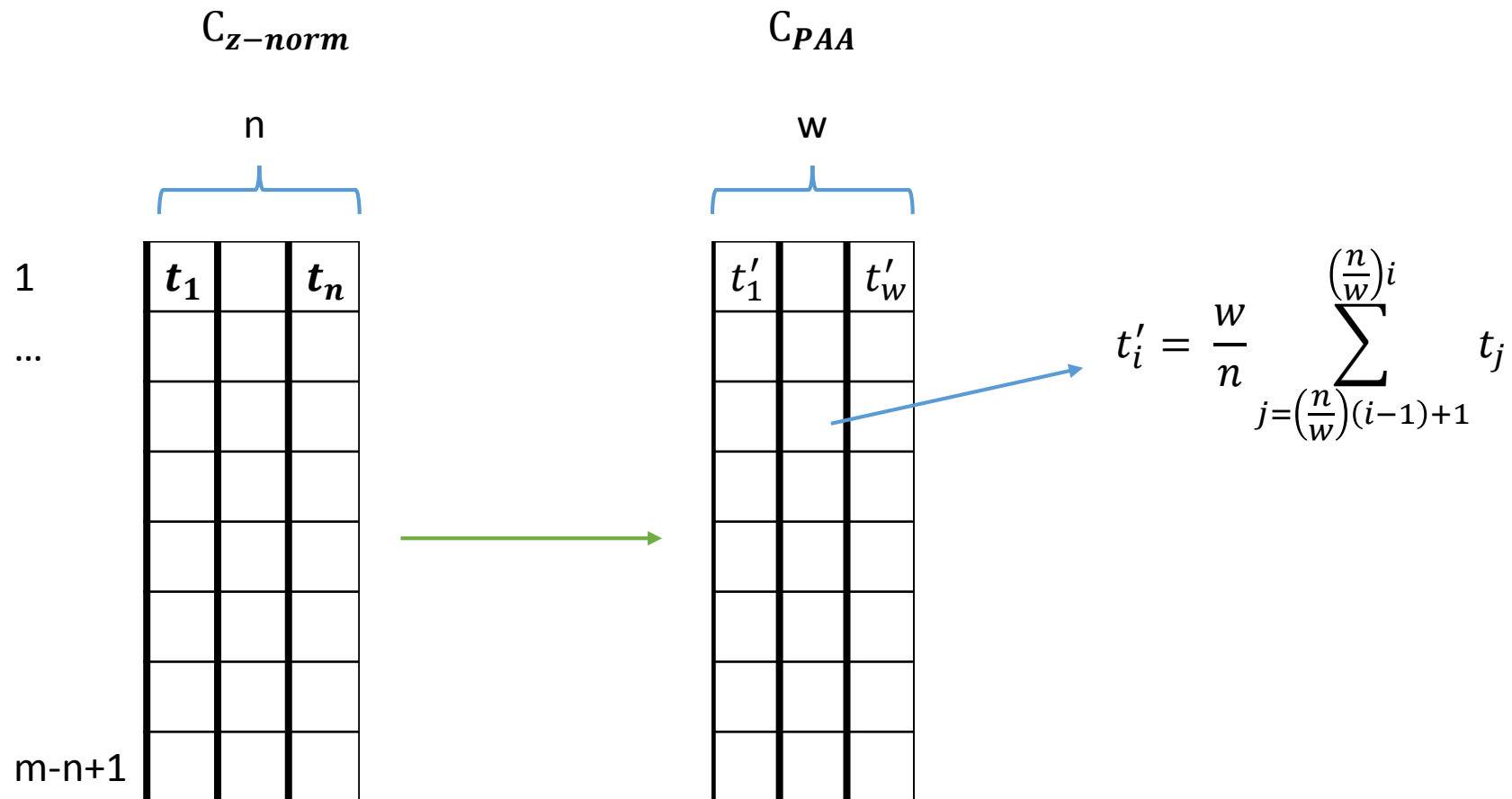
1. Подбор эвристики

1.1 z-нормализация подпоследовательностей C_i временного ряда



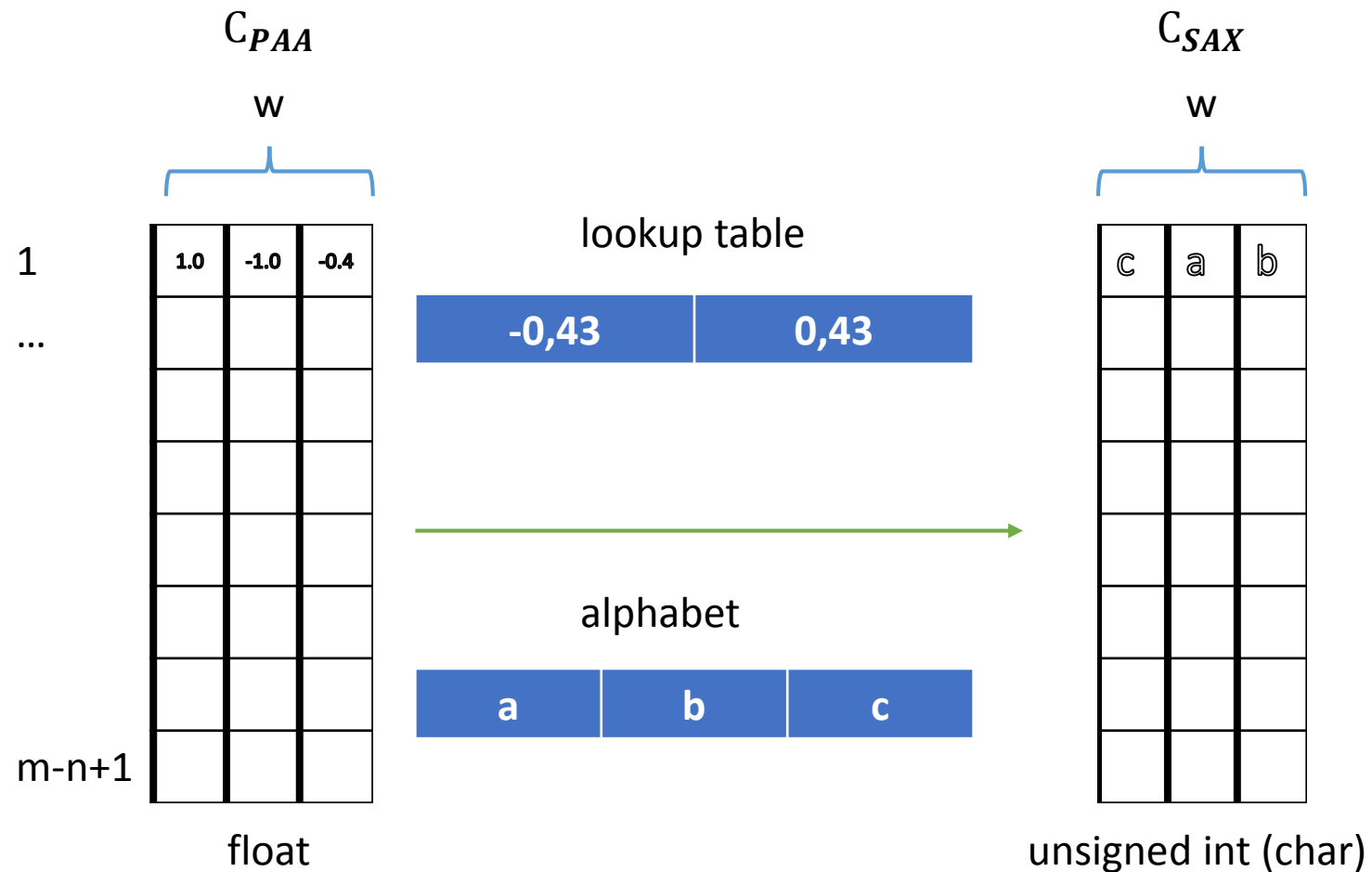
1. Подбор эвристики

1.2 Аппроксимация с помощью кусочной агрегации (РАА)



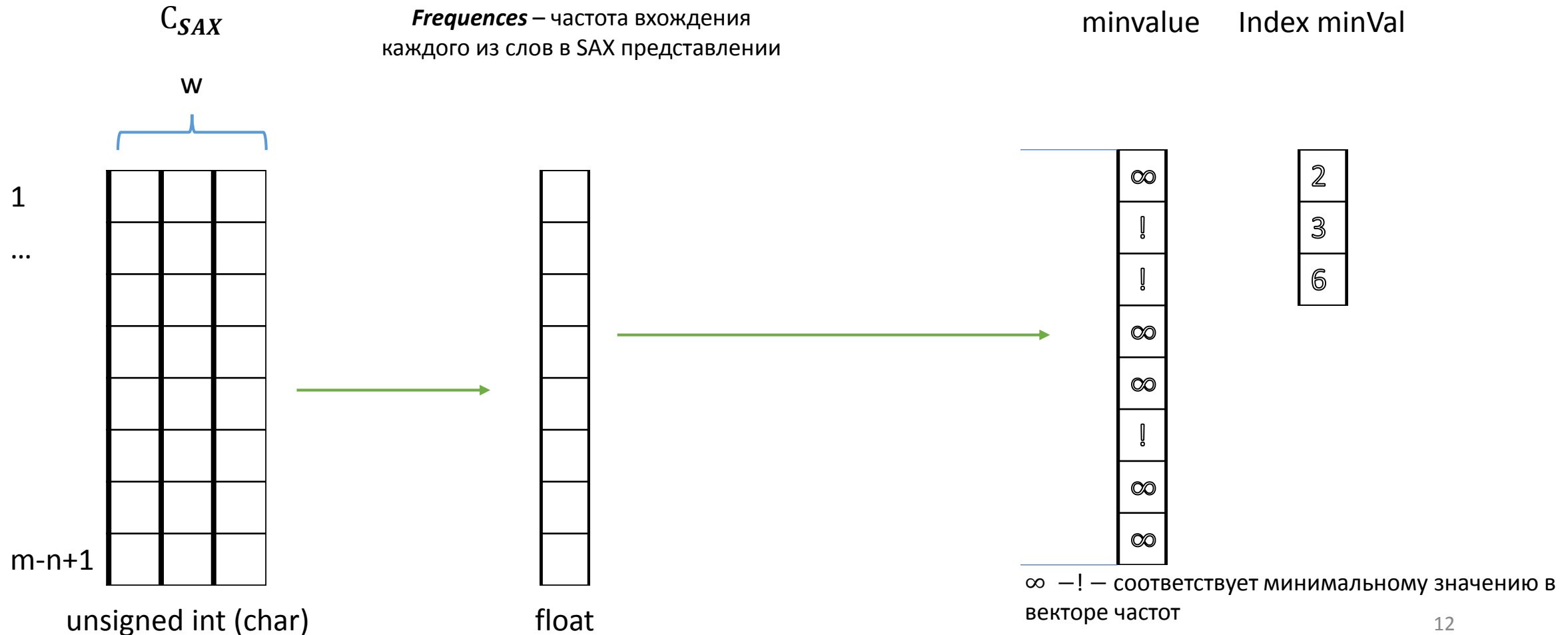
1. Подбор эвристики

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



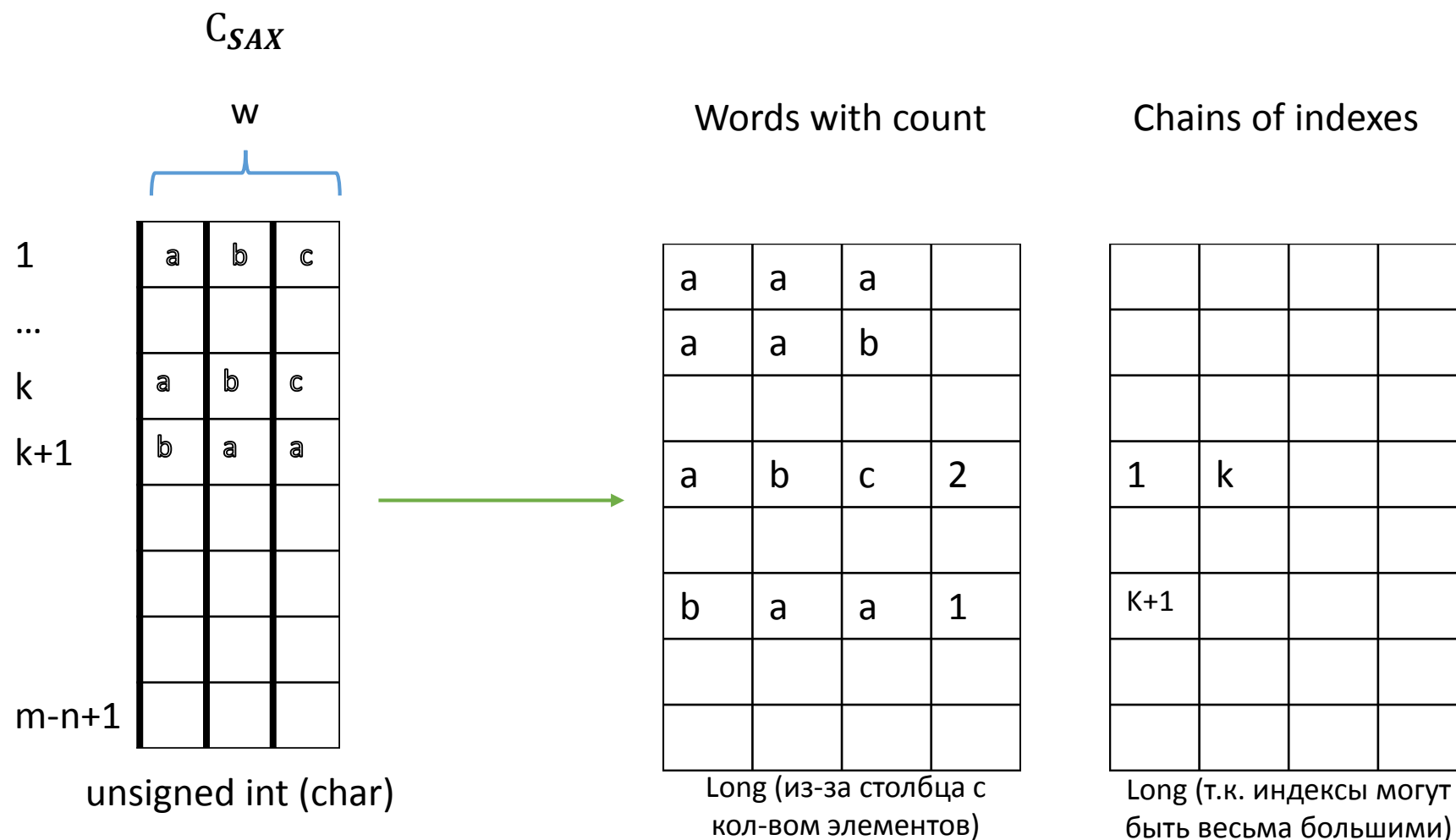
1. Подбор эвристики

1.4 Подсчет частот, нахождение мин. значения



1. Подбор эвристики

1.5 Построение цепочек с индексами начала подпоследовательностей, имеющих одинаковое SAX представление



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

```
For(int p = 0; p < m-n+1; p++) {  
    min =  $\infty$ ;  
    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;  
    }  
}
```

Последовательная версия алгоритма

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

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

```
For(int p in IndexMinVal) {
  min = ∞;
  For(int j in ChainsOfIndexes[sax(Cp)]){
    if(is_self_match(Cp, Cj) {
      continue;
    }
    distj = Euclid_dist(Cp, Cj)
    if (distj < best_so_far_dist) { -> Cp не подходит
      break;
    }
    if (distj < min) {
      min = distj;
    }
  }
  #pragma omp parallel for scheduled(guided)
  For(int j = 0; j < m-n+1 && not in ChainsOfIndexes[sax(Cp)]; j++){
    if(is_self_match(Cp, Cj) {
      continue;
    }
    distj = Euclid_dist(Cp, Cj)
    if (distj < best_so_far_dist) { -> Cp не подходит
      break;
    }
    if (distj < min) {
      min = distj;
    }
  }
  If(min > best_so_far_dist) {
    best_so_far_dist = min;
    best_so_far_pos = p;
  }
}
```

Не распараллеливаем, т.к.
предположительно мало
итераций

Ранний выход

Ранний выход

Для получения наилучшего (наиболее близкого к диссонансу) значения best_so_far_dist выполняется внешний цикл по всем подпоследовательностям с наиболее редко встречающимися SAX представлениями И два внутренних цикла: по предположительным ближайшим соседям подпоследовательностей из внешнего цикла и второй по всем оставшимся подпоследовательностям.

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

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

```
#pragma omp parallel for scheduled(guided)
```

```
For(int p = 1; p not in IndexMinVal && p < m - n + 1 ; p++) {
```

```
    min =  $\infty$ ;
```

```
    For(int j in ChainsOfIndexes[sax( $C_p$ )]){
```

```
        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 =  $dist_j$ ;
```

```
        }
```

```
    }
```

```
    For(int j = 0; j < m-n+1 && not in ChainsOfIndexes[sax( $C_p$ )]; 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;
```

```
    }
```

```
}
```

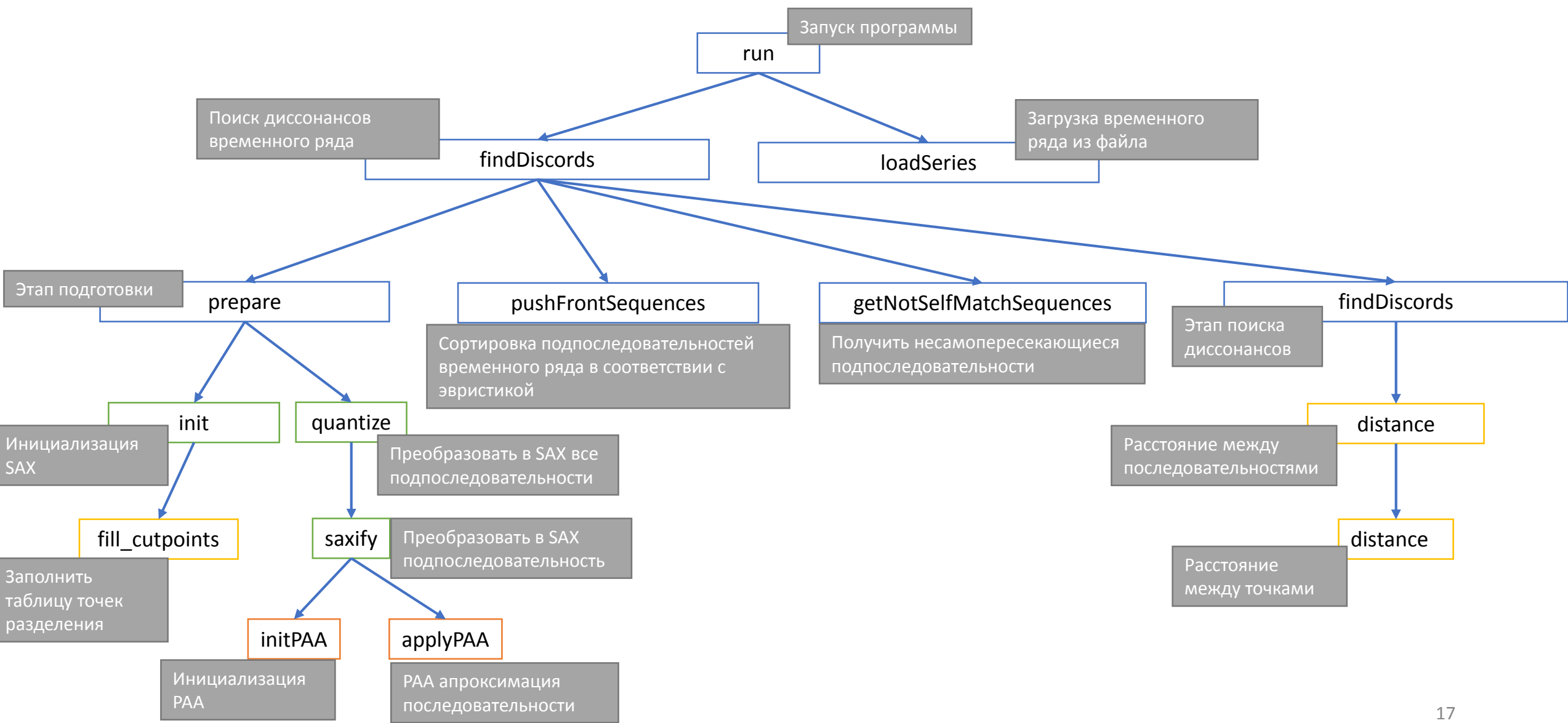
Ранний выход

Ранний выход

Далее внешний цикл по всем подпоследовательностям, которые не имеют наиболее редко встречающееся SAX представление. И два внутренних цикла: по предположительным ближайшим соседям подпоследовательностей из внешнего цикла и второй по всем оставшимся подпоследовательностям.

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

Модульная структура



Модульная структура

discords-finder

```
void run();
```

Запуск программы

```
long findDiscords(  
    time_series_t series,  
    long m,  
    long n  
);
```

Поиск диссонансов временного ряда

```
time_series_t loadSeries(long m);
```

Загрузка временного ряда из файла

```
long* prepare(  
    time_series_t* subsequences,  
    long n  
);
```

Этап подготовки

SAX

```
void pushFrontSequences(  
    long* sequences  
);
```

Сортировка подпоследовательностей временного ряда в соответствии с эвристикой

```
time_series_t* getNotSelfMatchSequences(  
    time_series_t series,  
    long m,  
    time_series_t subsequence,  
    long n  
);
```

Получить несамопересекающиеся подпоследовательности

```
long findDiscords(  
    time_series_t series,  
    long m,  
    long n  
);
```

Этап поиска диссонансов

Utils

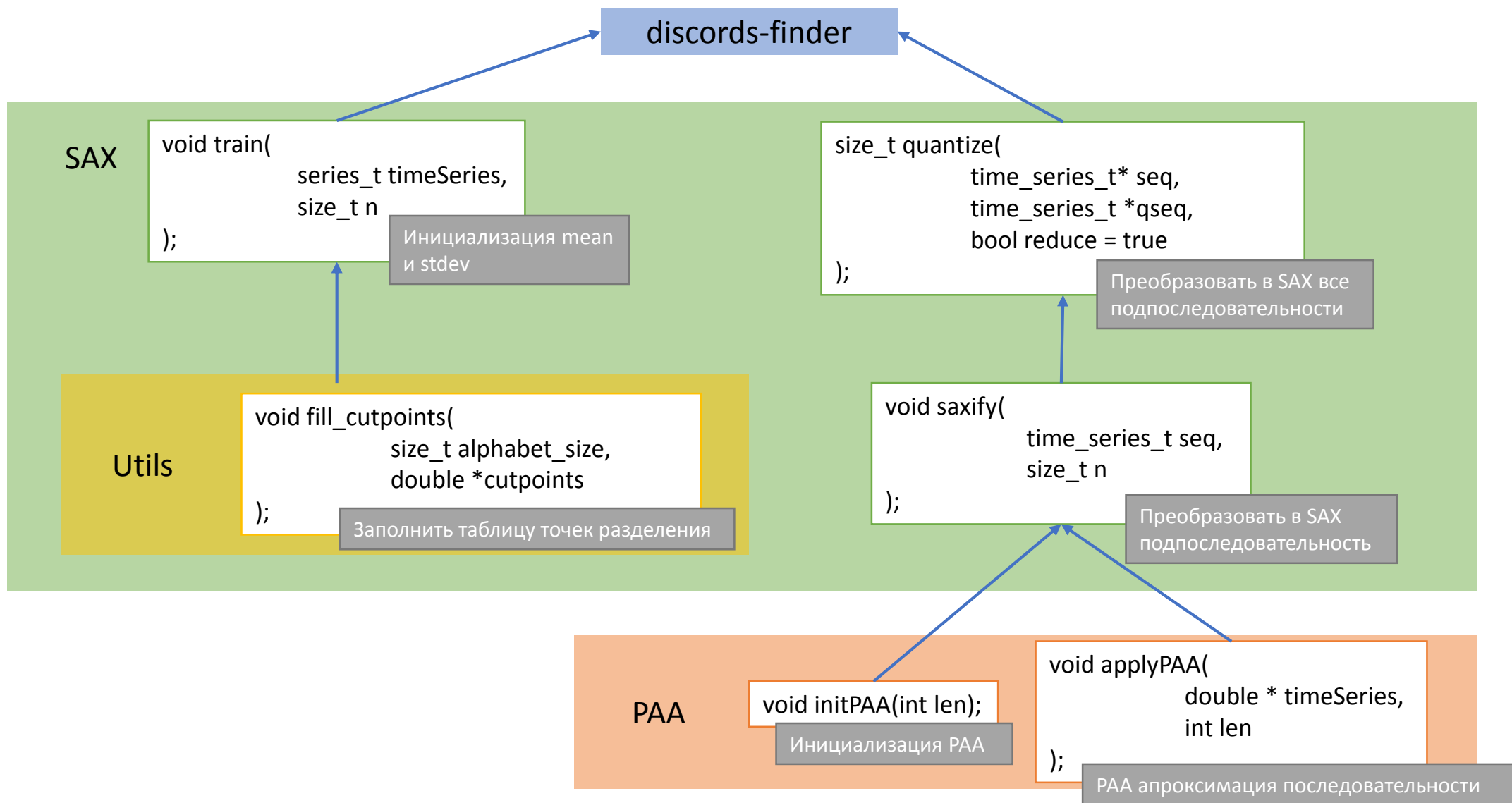
```
double distance(double series1[], double series2[], long length);
```

Расстояние между последовательностями

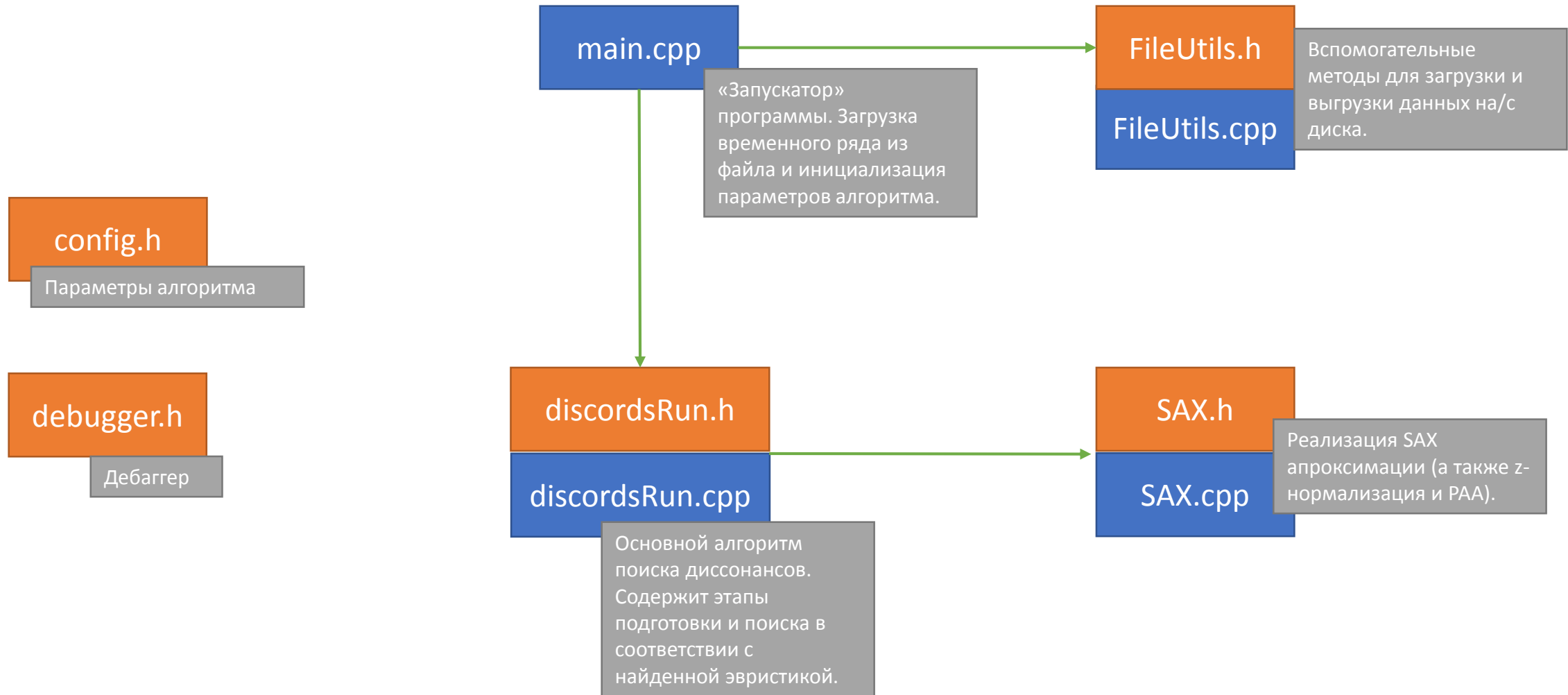
```
double distance(double p1, double p2);
```

Расстояние между точками

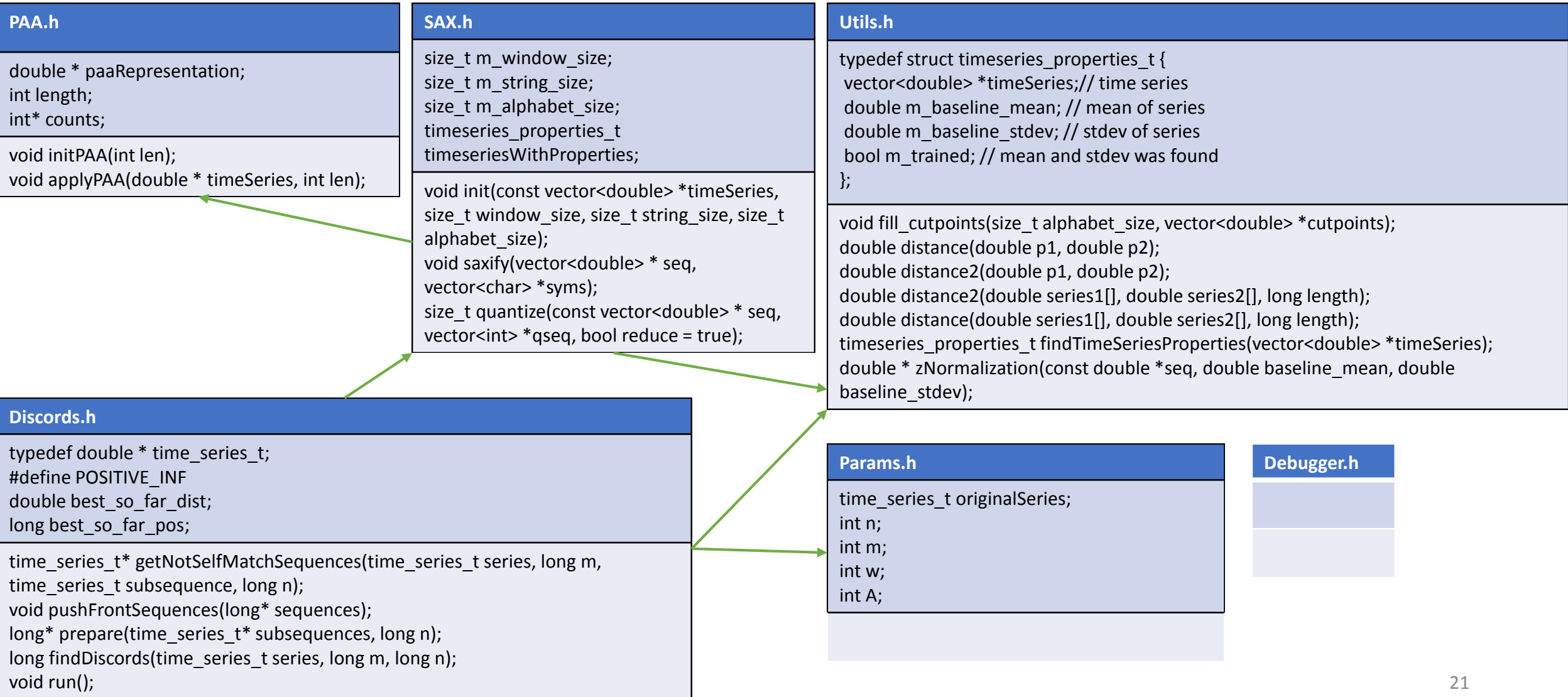
Модульная структура



Файловая структура

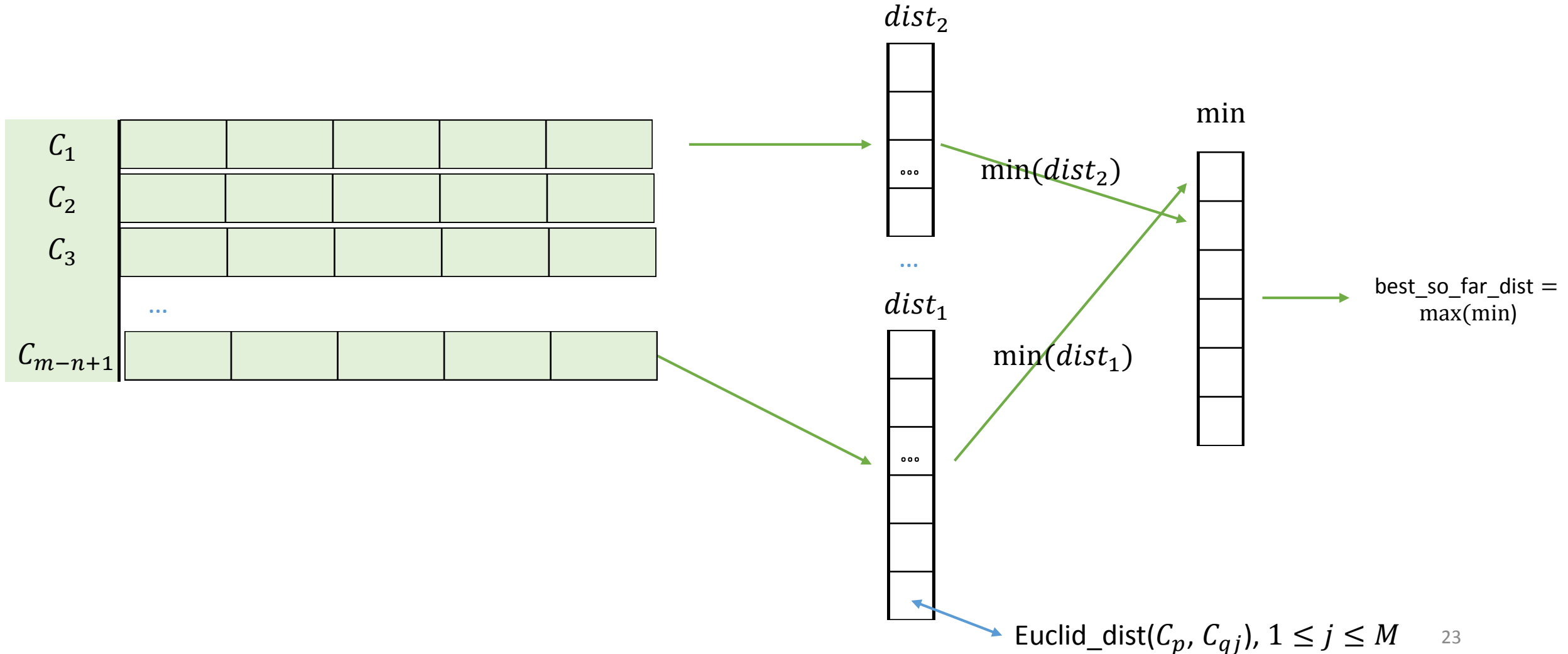


Файловая структура



Доп. Слайды (архив)

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



Упорядочивание множества подпоследовательностей

Index minVal

2
3
k

2	t_2	t_3	t_4		t_{n+1}
3	t_3	t_4	t_5		t_{n+2}
k	t_k	t_{k+1}	t_{k+2}		t_{n+k-1}
1	t_1	t_2	t_3		t_n
4	t_4	t_5	t_6		t_{n+3}
	...				
m-n+1	t_{m-n+1}	t_{m-n+2}	t_{m-n+3}		t_m

m-n+1

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

Формируем матрицы $(n \times (1 + \text{nsm_count}_p))$ для каждой подпоследовательности C_p , $1 \leq p \leq m - n + 1$, где nsm_count_p - количество несамопересекающихся с C_p подпоследовательностей.

