Институт проблем передачи информации

Российской Академии наук

Лаборатория «**Математических методов и моделей в биоинформатике**»

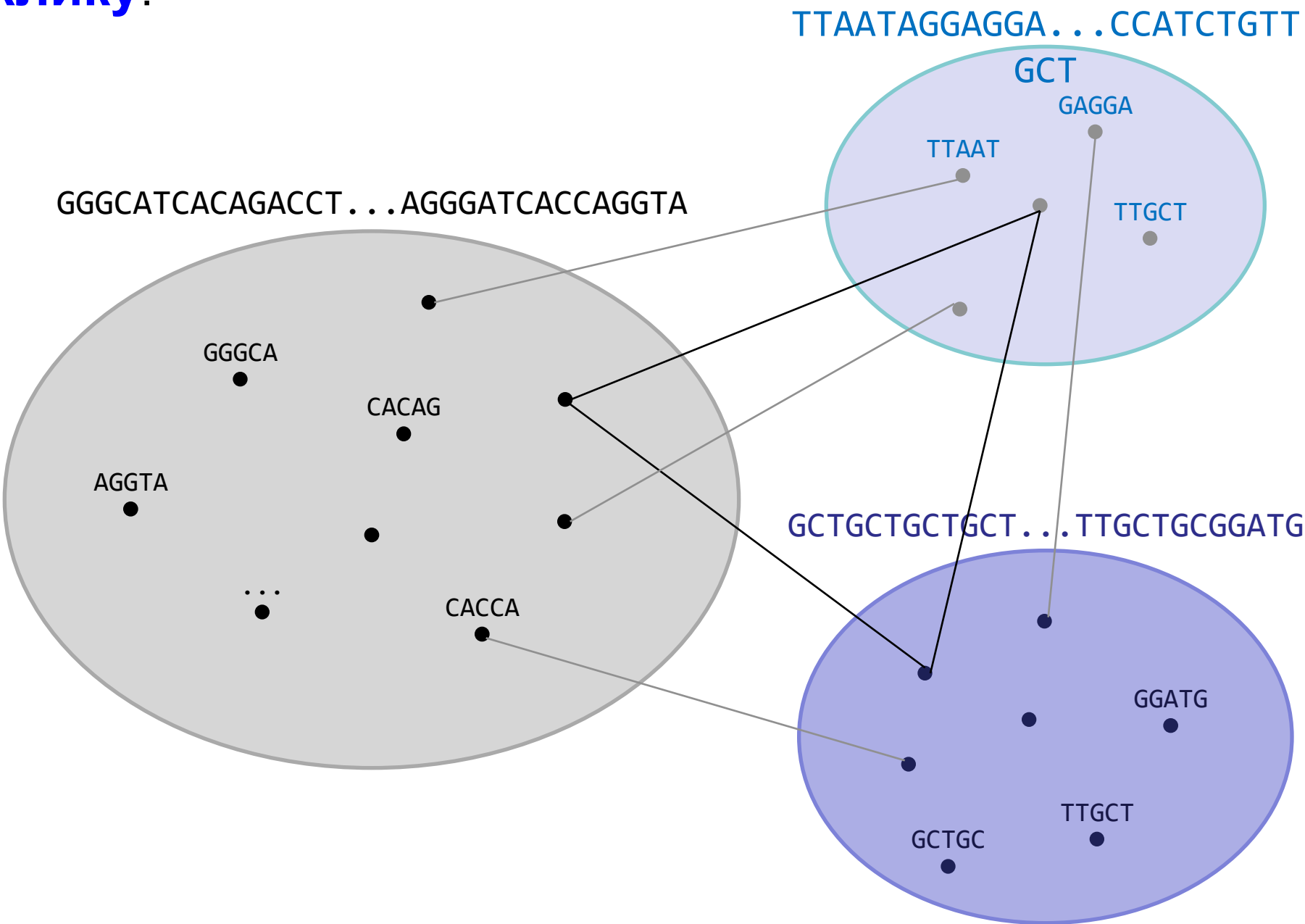
сайт лаборатории: **http://lab6.iitp.ru/**

адрес: **lyubetsk@iitp.ru**

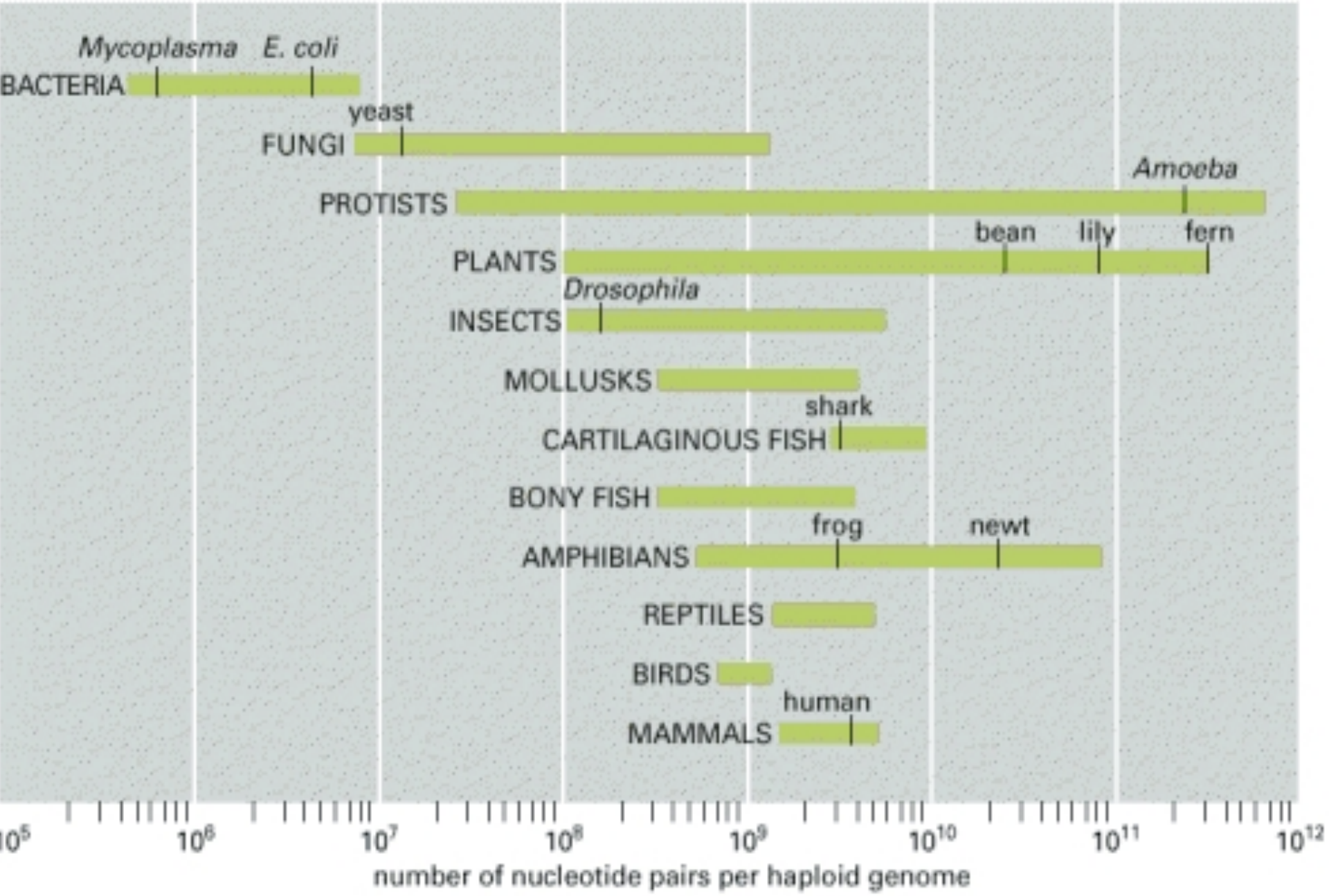
В. Любецкий

Эффективный параллельный алгоритм поиска плотных подграфов. Построение исходного графа

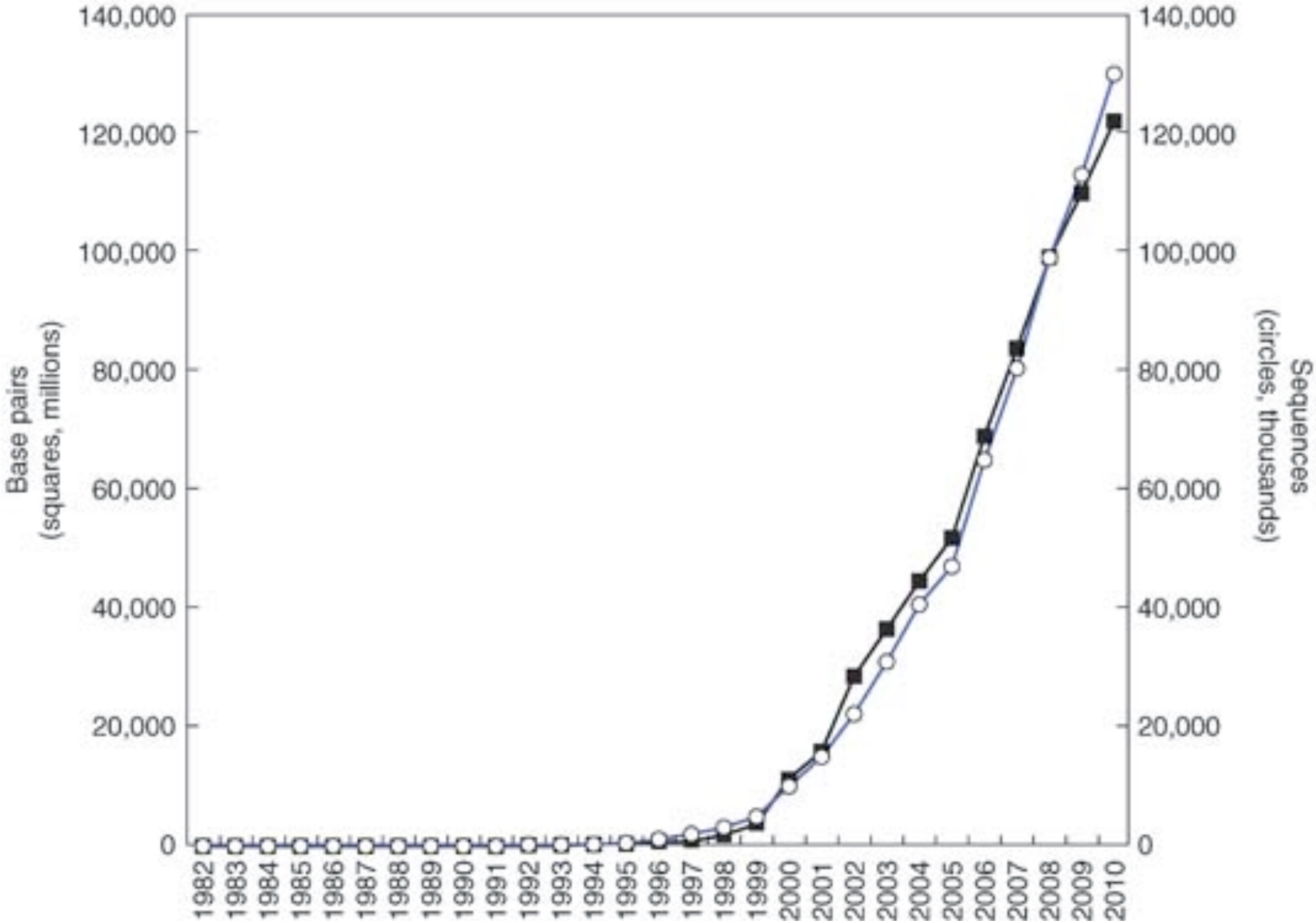
Пример **многодольного графа**, в котором ищем **клику**:



Длины последовательностей



Рост числа последовательностей в одной базе данных



Итак, **о размере графа**. Может потребоваться: число долей 10^8, длина последовательности 10^9-10^{12}, берутся короткие слова, т.е. **вершин 10^{19}** и соответственно рёбер (порядки).

Кроме того, **задача поиска клики «неразрешима»**: поиск *m*-клики в *m*-дольном графе, если каждая доля которого содержит хотя бы по три вершины, – **NP-трудная задача**. Это показано для графа с очень высокой плотностью рёбер. Если плотность рёбер низкая, то сложность этой задачи не известна. Но??

Поэтому вместо клики ищут **плотные подграфы**.

Подграф многодольного графа называется ***m*-плотным**, если он связный и каждая его вершина соединена ребром с вершиной в не менее чем (*m*-1)ой долях.

Иногда **рёбрам** приписывается редакционное **расстояние** слов на его концах.

Или иная мера их близости, сходства, называемая **весом ребра**.

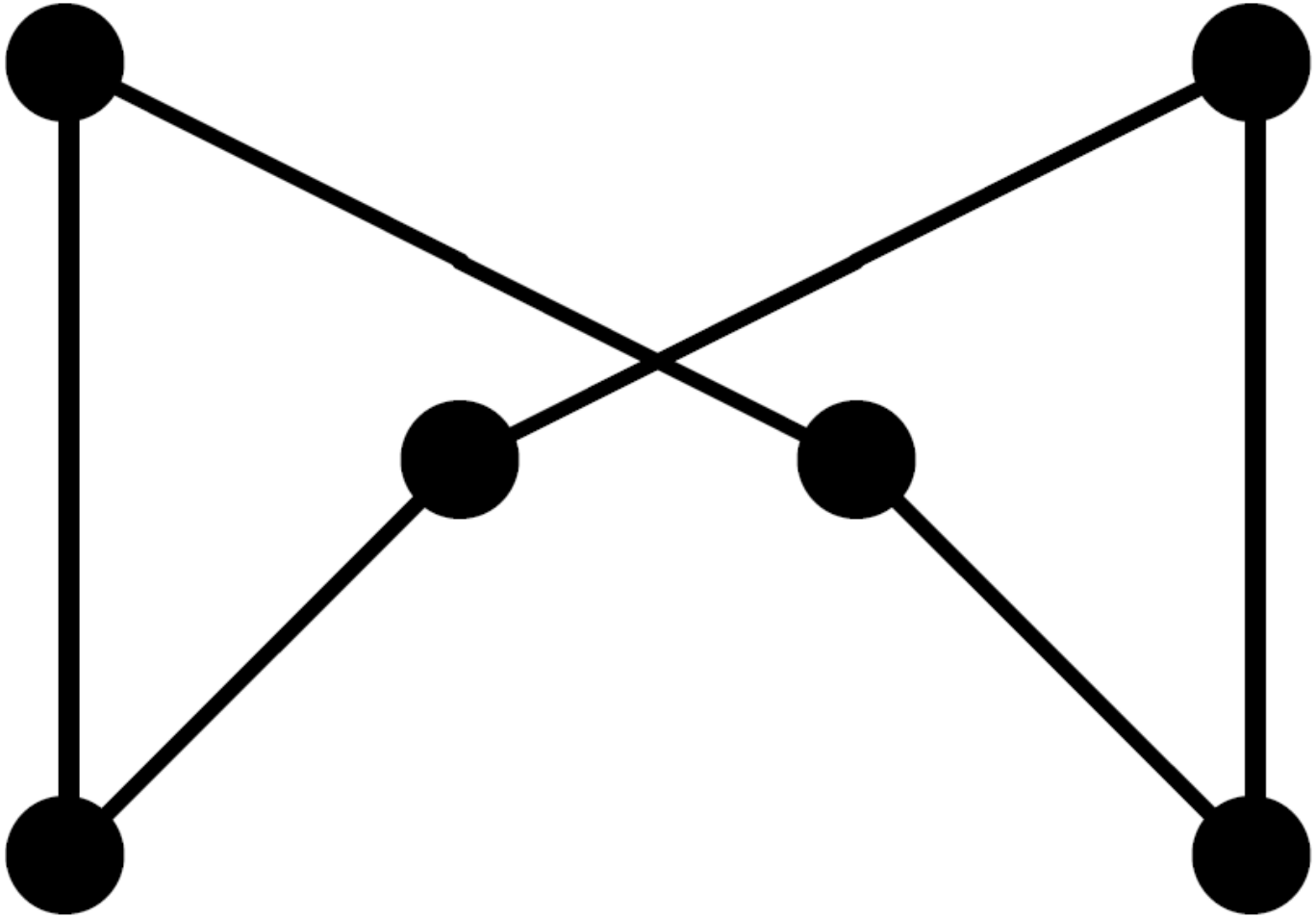
Кластер – 1) связный *m*-плотный подграф, который 2) состоит из рёбер большого веса;

часто добавляют ещё два требования:

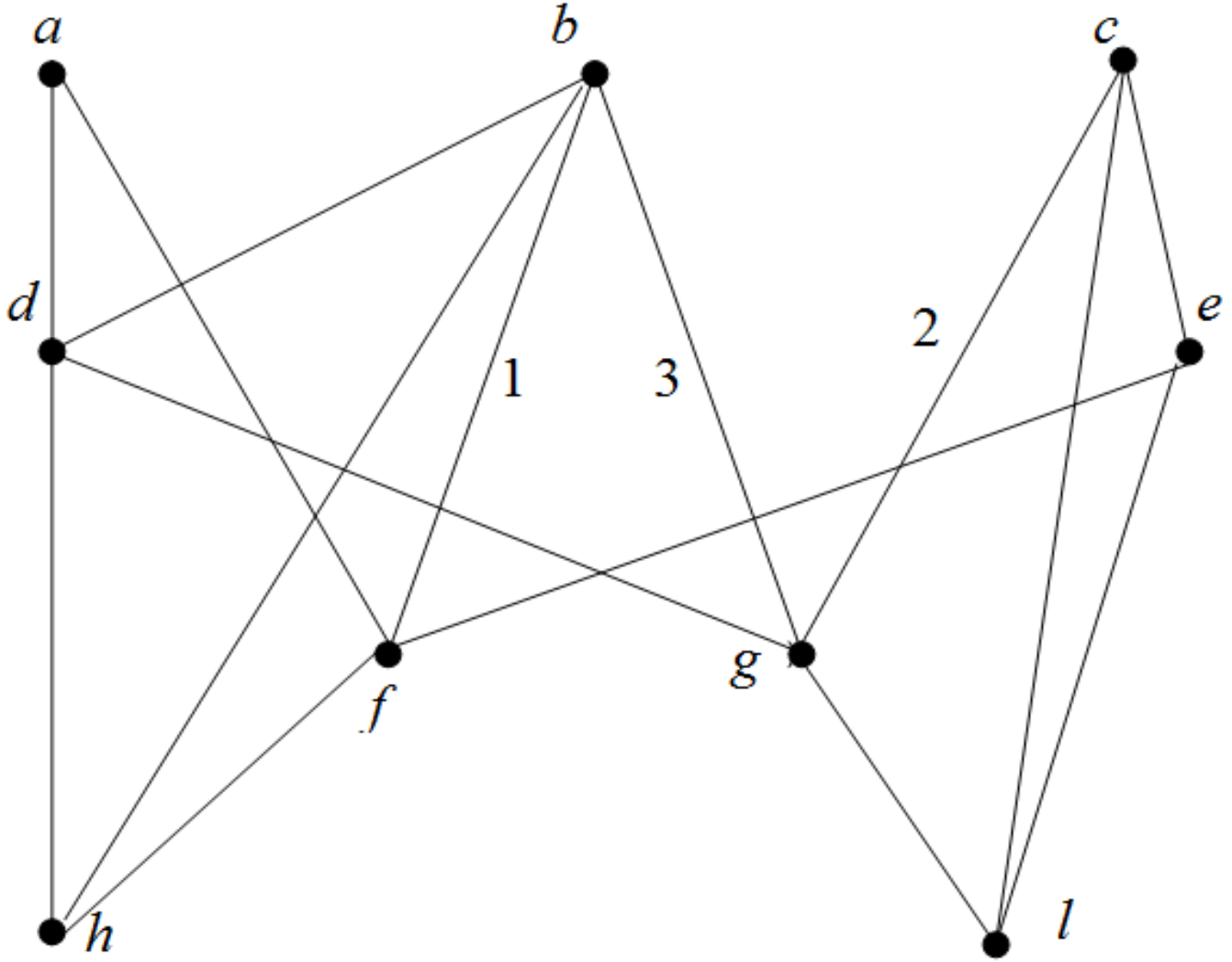
3) не содержит много вершин из одной доли,

4) слова высокой колмогоровской сложности.

Пример **3-дольного графа**,
он же – **3-плотный** граф
(в нём нет 3-клик):



4-дольный граф, содержащий 4-плотный подграф: он состоит из всех вершин, кроме a. Ребра помечены весами (показана часть весов).



Нами предложен **алгоритм поиска кластеров**, допускающий **эффективное распараллеливание**.

Однако для применения алгоритма нужно ещё приготовить **исходный граф** – задача, не менее сложная, чем поиск самих кластеров.

Поиск кластеров универсальный, но **построение исходного графа** сильно зависит от задачи, которую хотят решить.

Метод = алгоритм построения исходного графа + алгоритм поиска кластеров.

Описание метода: **1)** выделяются пары слов (т.е. рёбра) с высокой колмогоровской сложностью (например, слабо сжимаемые алгоритмом Лемпеля-Зива), которые близки в смысле редакционного расстояния, или как-то.

2) строится многодольный граф, вершинам приписаны слова из последовательностей долей, рёбрам – веса; доля соответствует одной последовательности;

3) в нём ищутся кластеры.

В пунктах 1-2 строится **исходный граф**.

В пункте 3 применяется **универсальный алгоритм поиска кластеров**.

Начнём с п. 3, т.е. с **изложения самого алгоритма**.

В каждой вершине графа алгоритм независимо и одновременно выполняет следующие операции:

1) Если вершина соединена рёбрами менее чем с (m–1) долей, то она и все инцидентные ей рёбра удаляются. Если вершина соединена с какой-то долей лишь одним ребром, то оно помечается (такое ребро удаляется только вместе с одним из его концов).

Если граф изменился, то снова выполняется вышесказанное.

2) Если инцидентное вершине ребро не помечено, а его вес строго меньше весов всех других непомеченных инцидентных ей рёбер (если они имеются), то ребро удаляется.

Если после выполнения шага 2 в графе произошли изменения, то возвращаемся к шагу 1.

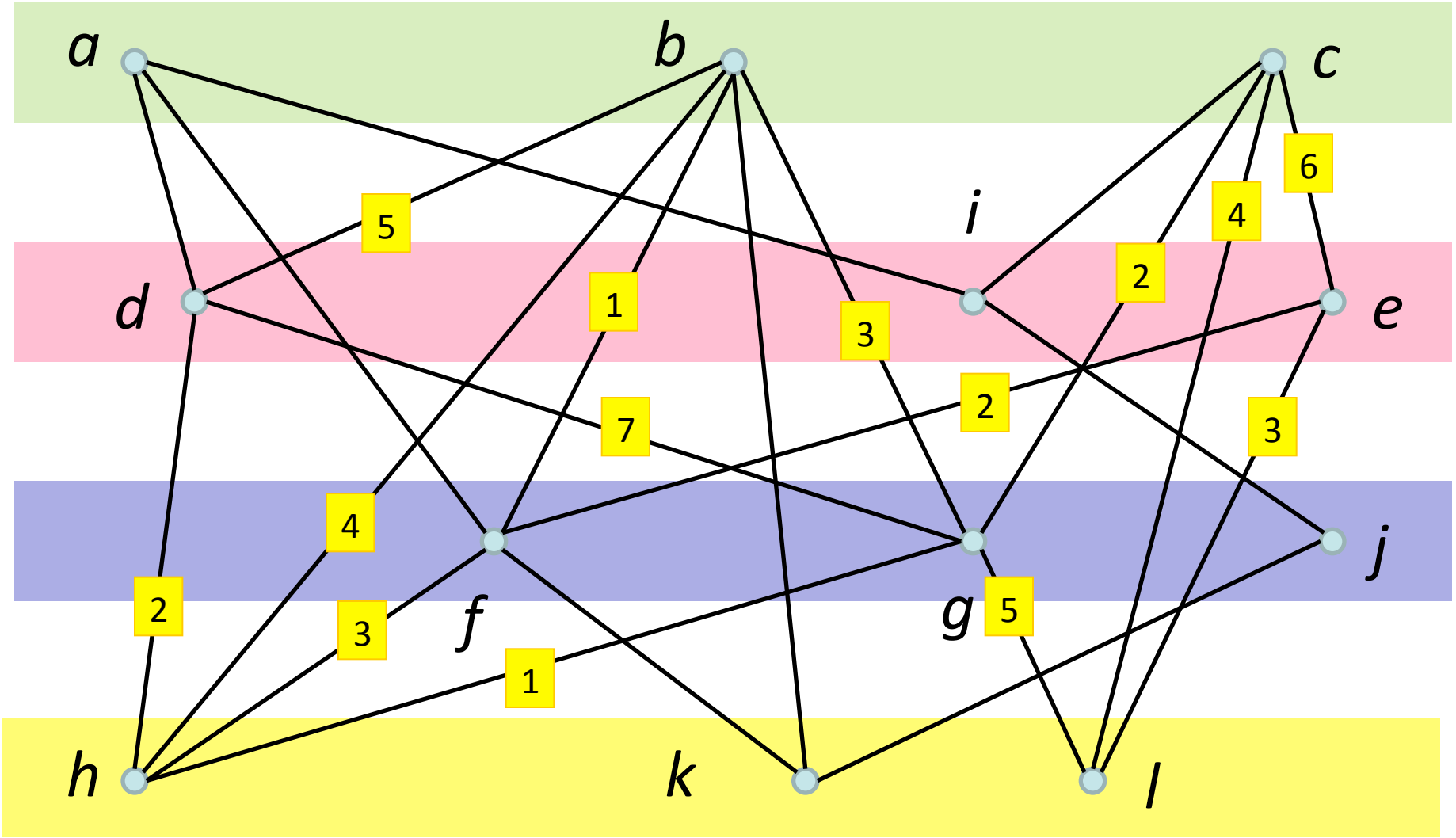
В противном случае алгоритм заканчивает работу. ///

Каждая компонента связности полученного в результате графа – искомый кластер. ///

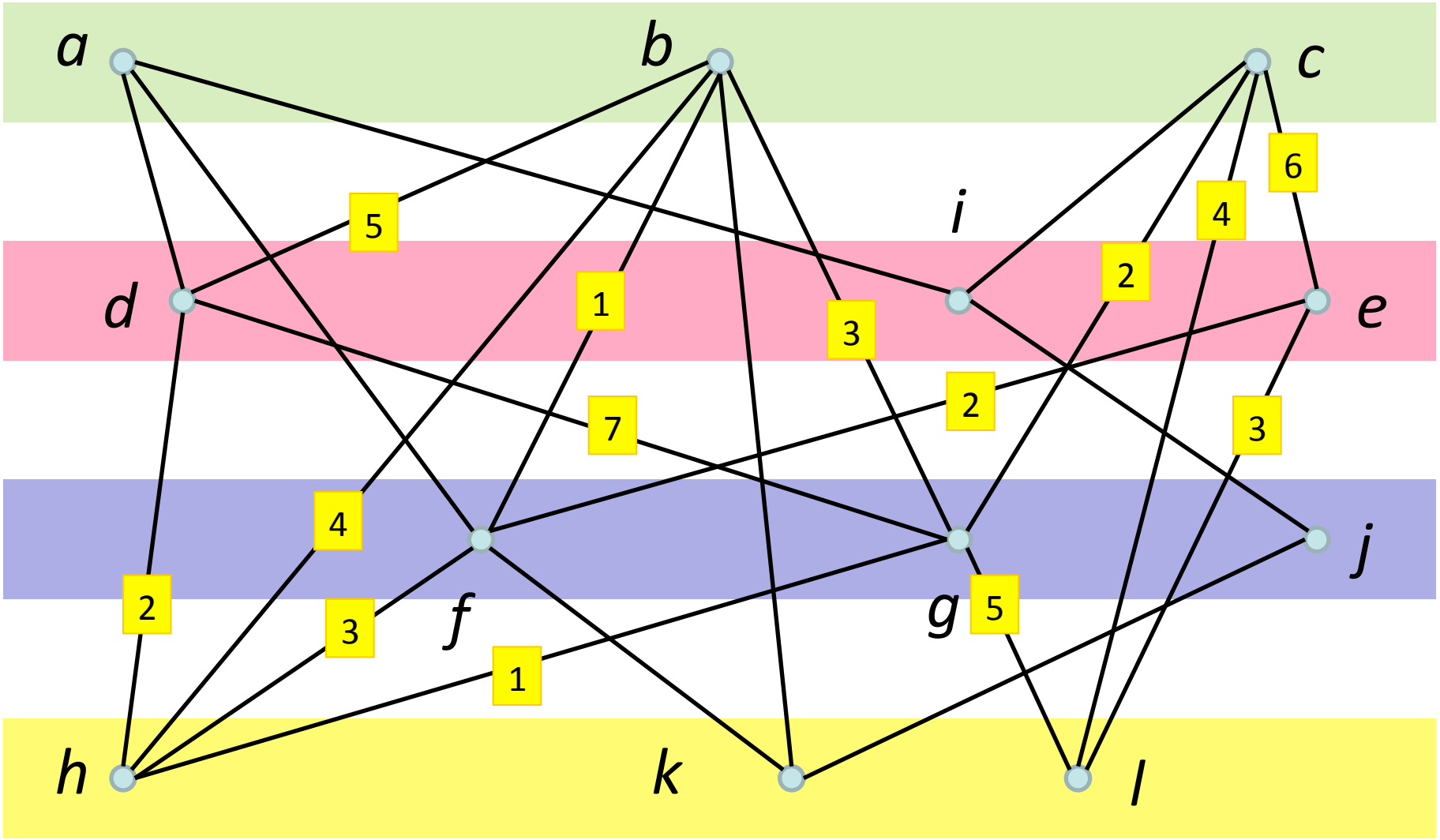
О программировании (кратко): 1) После каждого шага выполняется синхронизация процессов во всех вершинах графа: следующий шаг начинается после окончания предыдущего шага во всех вершинах.

2) Подразумевается некоторый межпроцессный обмен: если на другом конце ребра инцидентного данной вершине ребро удалено, то оно удаляется и в ней; если помечено – помечается. Представьте себе работу клеточного автомата:«клетка»=вершина, «соседние клетки» = смежные вершины.

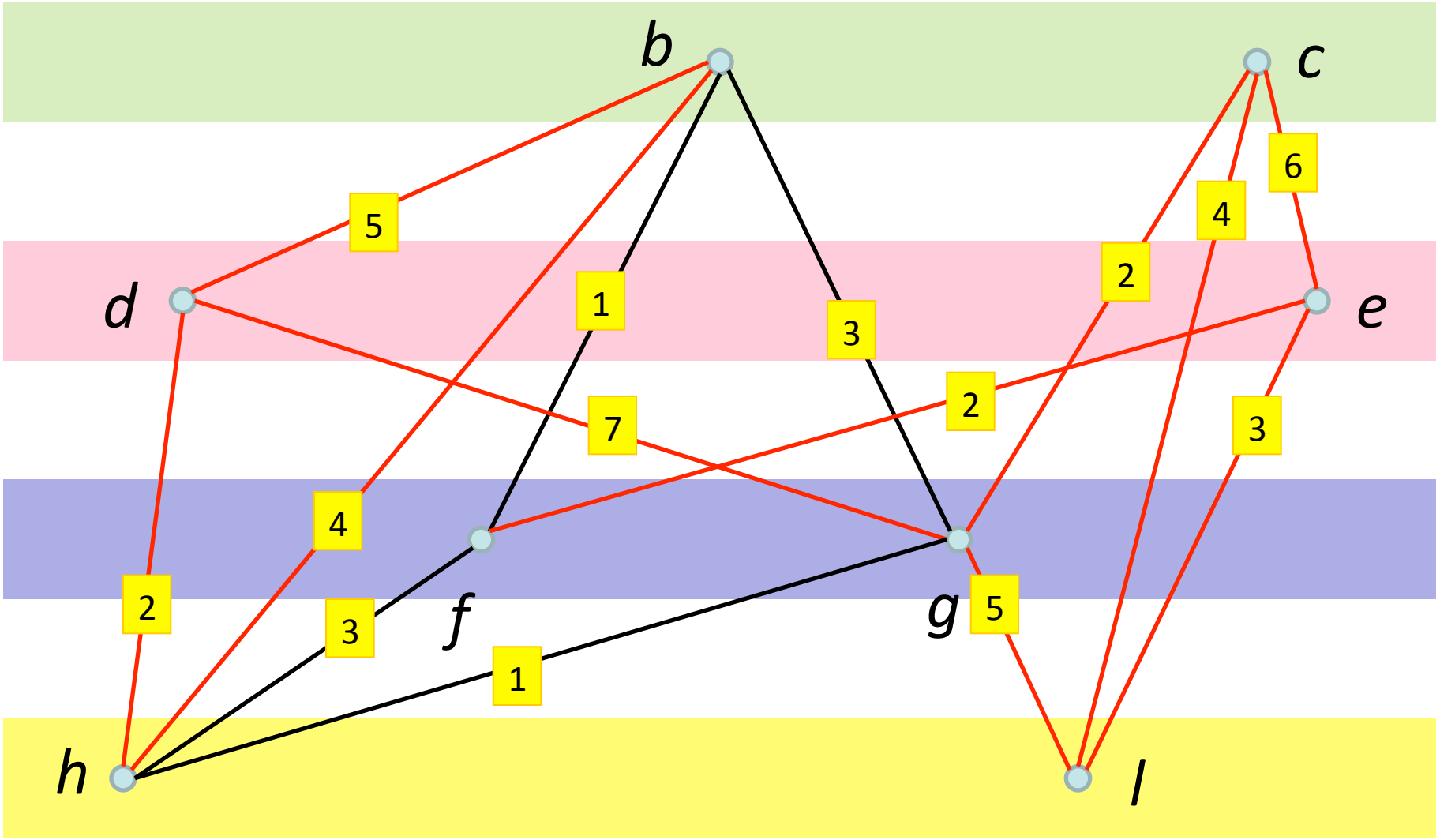
Пример: 4-дольный граф, содержащий 4-плотный подграф. Он состоит из всех вершин, кроме *a, i, j, k*. Некоторые рёбра помечены своими весами.



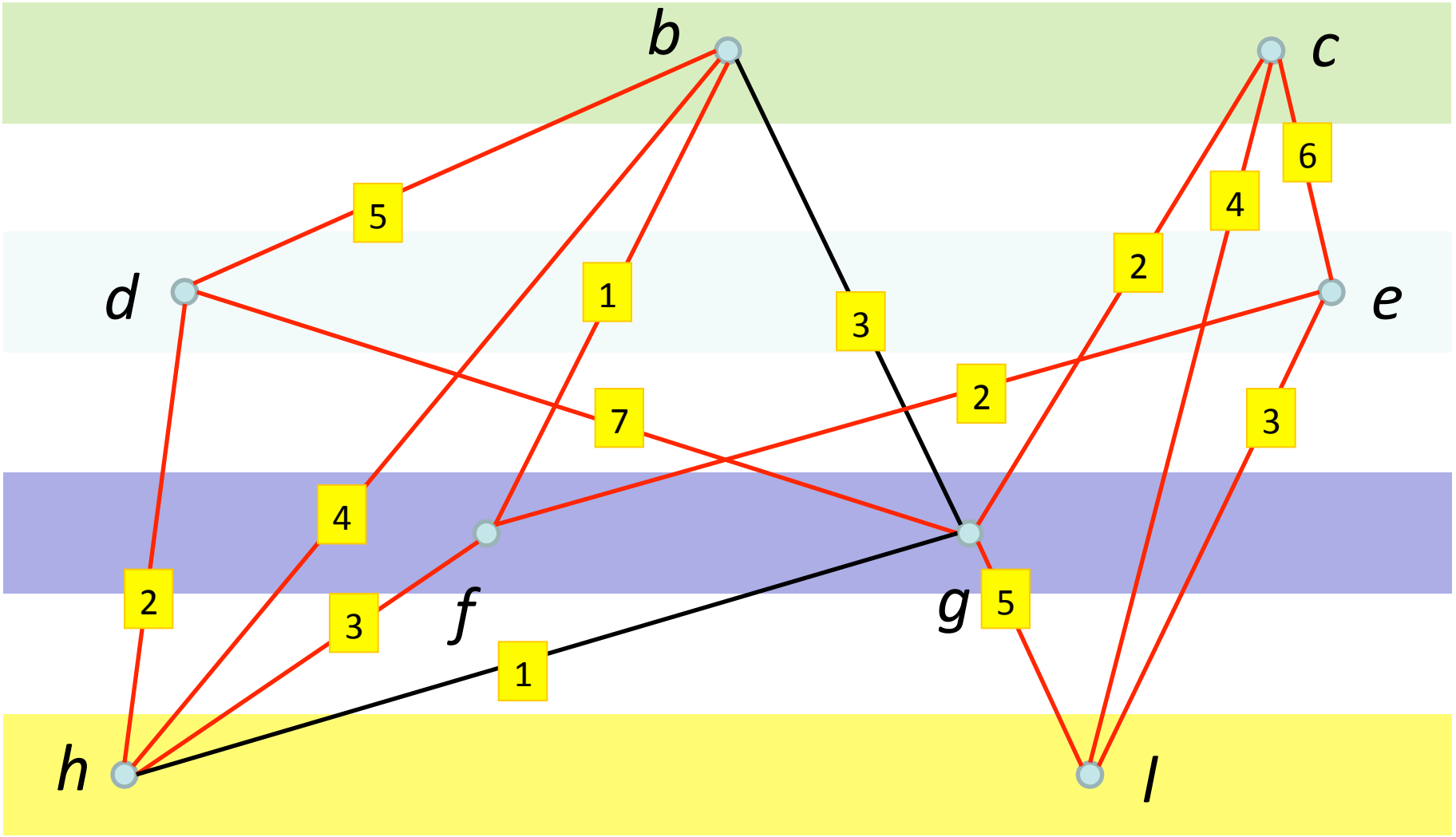
Шаг 1.1: Удаляются вершины *a, i, j, k* со своими рёбрами. Часть других рёбер помечаются как «защищенные» (показаны красным).



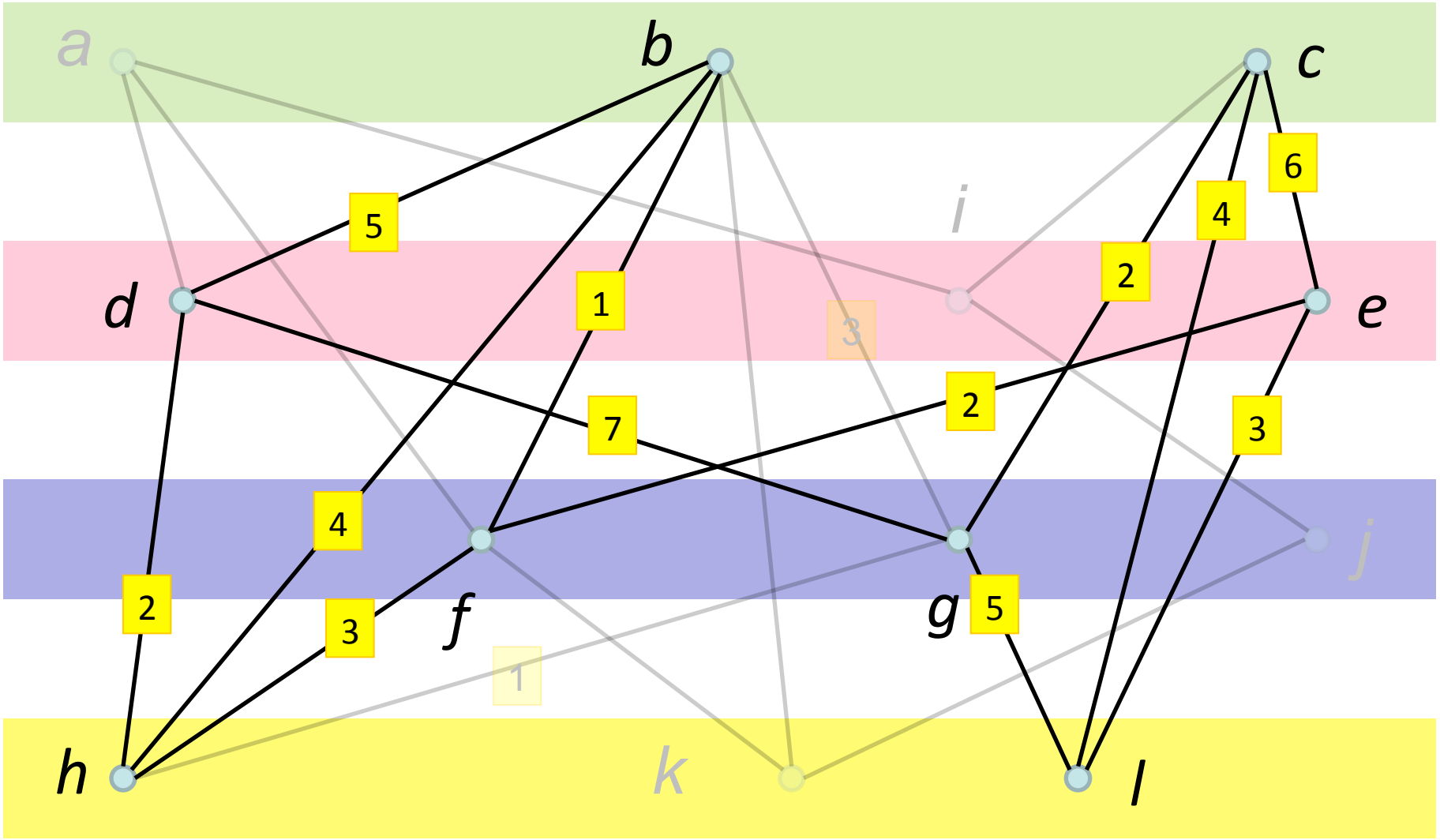
Шаг 1.2: Дополнительно рёбра *fb* и *fh* помечаются как «защищенные» (а *bg* и *gh* – нет). После этого шаг 1 больше не применим.



Шаг 2.1: Удаляется незащищенное ребро *hg* при вершине *h* и ребро *bg* при вершине *b* (но только *gh* при вершине *g*). После этого шаги 1 и 2 больше не применимы. Конец.



4-плотный подграф, построенный алгоритмом:



Наш опыт показал, что параллельный алгоритм, работающий по принципу клеточного автомата, завершает работу после небольшого числа итераций даже для графов огромного размера.

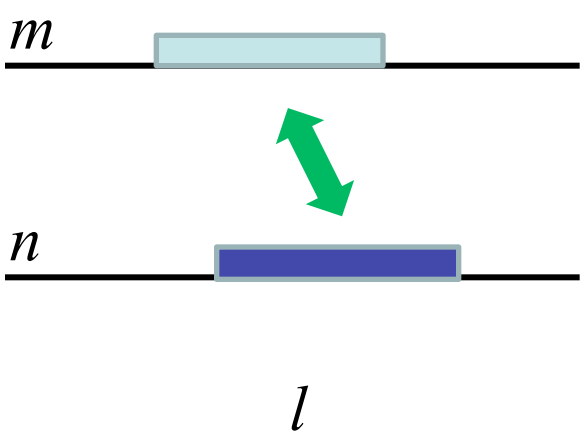
Здесь было бы интересно провести статистически значимое тестирование. Как его организовать?

Вернёмся к совершенно нетривиальной проблеме – **построению исходного графа**. В зависимости от задачи различают два случая: требуются **большие доли** или можно ограничиться **маленькими долями**.

Начнём со случая, когда задача требует больших долей. Тогда для построения исходного графа выполняют:

1. Попарное сравнение последовательностей и нахождение рёбер.
2. Склейка вершин и построение самого исходного графа.

1. Даны две нуклеотидные последовательности длиной *m* и *n*. Найти все пары **слов** длины более *l*, имеющие сходство выше заданного (например, веса больше ε) и высокую колмогоровскую сложность (например, сжимаемые алгоритмом Лемпеля-Зива с коэффициентом менее *r*).



Сложность наивного алгоритма $O(mnl^2)$ – неприемлема для поиска пар слов в последовательностях, тем более сразу во многих (тогда нужно ещё умножить на N^2). Необходим очень быстрый алгоритм.

Случай больших ε, т.е. слова буквально совпадают на связных компонентах пересечения, пусть наименьшая длина компоненты k (< l), например, k = 24…32

1) По первой последовательности составляем хэш-таблицу из ключей с их позициями (всех длины k). Хэш-таблица занимает порядка 4-16 гигабайт для последовательностей длиной до 120 млн.

2) Во второй последовательности перебираем подряд все ключи длины k и для каждого из них проверяем, совпадает ли оно с каким-то ключом из таблицы.

3) Сопоставляем окрестности каждой найденной пары «ключ – ключ», чтобы проверить, есть ли в них искомая пара слов. Если находим, то **образуем из них ребро**.

4) Продолжаем так с каждой следующей последовательностью, используя построенную таблицу для 1й последовательности (одну и ту же).

5) То же самое делаем для 2й последовательности против списка всех оставшихся посл-тей; и т.д.

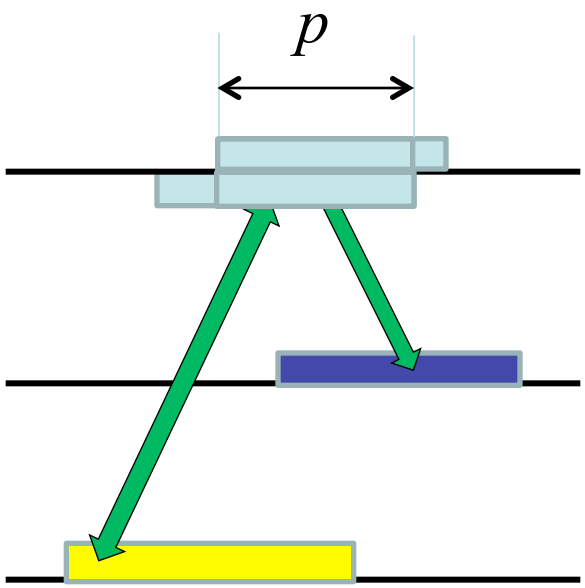
Полученные пары участков составят набор рёбер.

Вычислительная сложность шагов 1-3: $\leq O((m+n)\cdot l^2)$ и даже линейная, совпадение ключей указанной длины в двух последовательностях встречается весьма редко, типично < 10% ключей.

Память: $O(m\cdot k+n)$. Расширение на другие последовательности линейно.

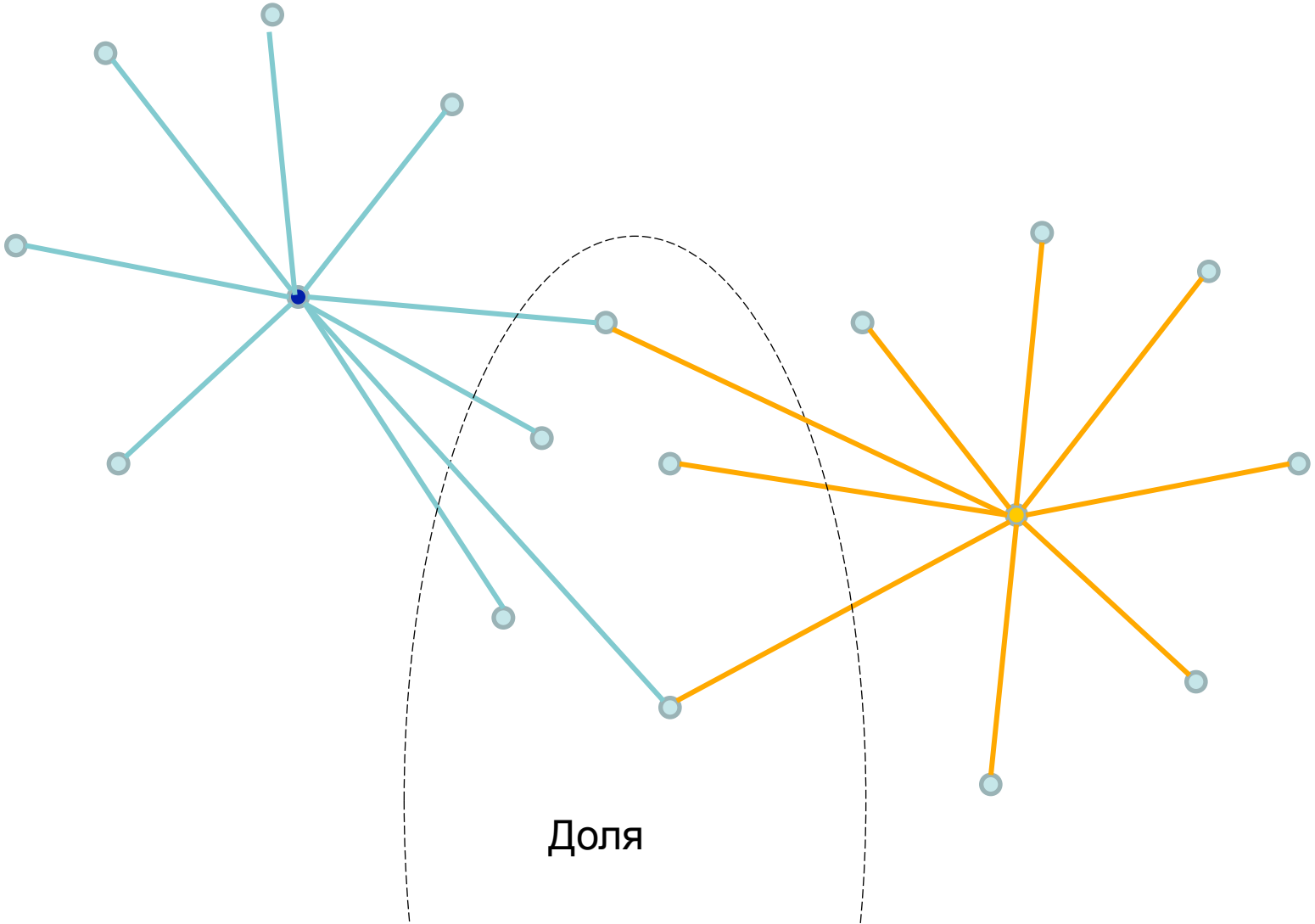
2. Построение исходного графа

Элементарная операция: вершины двух рёбер в одной последовательности склеиваются, если соответствующие слова пересекаются на длине не менее p ($\leq l$). Новое слово (=вершина) равно пересечению участков. По умолчанию p=120 для l=150.



Операция применяется к каждой вершине по отдельности, не трогая другой конец ребра.

Вместо отдельных рёбер эффективнее рассматриваем звёзды со многими откликами, и склеиваем отклики из одной доли:



Эффективное вычисление редакционного расстояния:

для этого нами найден

линейной сложности алгоритм

(полулокальное

выравнивание с одним закреплённым концом).

Стандартный алгоритм глобального выравнивания

Needleman, Wunsch (1970)

a b →		G	C	G	T	T	G
↓	0	2	4	6	8	10	12
A	2	1	3	5	7	9	11
C	4	3	1	3	5	7	9
G	6	4	3	1	3	5	7
C	8	6	4	3	2	4	6
T	10	8	6	5	3	3	5
T	12	10	8	7	5	3	4
G	14	12	10	8	7	5	**3**

Инициализация:

$$F[0][0] = 0;$$

$$F[i][0] = id,\ P[i][0] = \text{UP},\ \ i = \overline{1, M};$$

$$F[0][j] = jd,\ P[0][j] = \ \text{LEFT},\ j = \overline{1, N}$$

Заполнение матриц:

$$F[i][j] = \min \begin{cases} F[i-1][j-1] + g(i,j) & (\text{case1}) \\ F[i-1][j] + d & (\text{case2}) \\ F[i][j-1] + d & (\text{case3}) \end{cases}$$

где $g(i,j) = \begin{cases} 0, a[i] = b[j] \\ 1, a[i] \neq b[j] \end{cases}$; $d = 2$ (пример)

$$P[i][j] = \begin{cases} \text{DIAG} \ \textbf{if} \ (\text{case1}) \\ \text{UP} \ \ \textbf{if} \ \ (\text{case2}) \\ \text{LEFT} \ \textbf{if} \ (\text{case3}) \end{cases}$$

Редакционное расстояние: $F[M][N] = 3$

Обратный ход:

GCGCTTG

ACG–TTG

Сложность $O(MN)$, память $O(MN)$

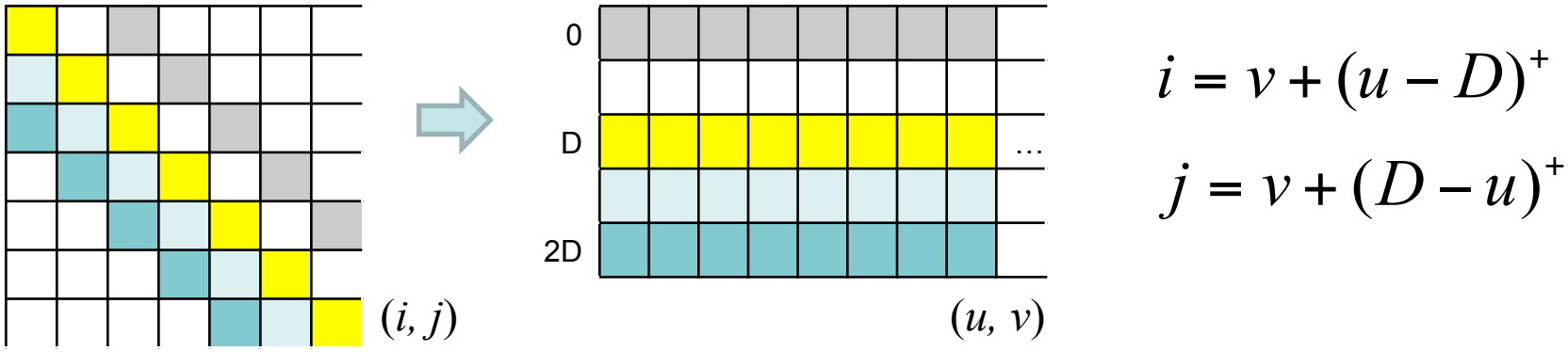
Недостатки стандартного алгоритма:

•глобальный (нужно точно знать начало и конец обоих участков),

•трудоемкий (квадратичный),

•нельзя ограничить вычисления заданным пределом расстояния.

Выполним преобразование части матриц *F* и *P*:



$$i = v + (u - D)^+$$

$$j = v + (D - u)^+$$

т.е. рассматриваем главную диагональ матрицы и *D* соседних с ней диагоналей с каждой стороны

Модифицированный алгоритм:

•полулокальный (закреплен один конец каждого участка, другой заранее неизвестен);

•имеет сложность $O(DM)$ (линейный, если $D << M$);

•вычисления проводятся по столбцам в порядке нумерации на рисунке, поэтому их можно остановить, когда минимальное расстояние в столбце достигает заданного предела;

•<u>достаточное</u> условие оптимальности алгоритма – число делеций в оптимальном выравнивании не превышает D;

•<u>необходимое</u> условие – траектория обратного хода стандартного алгоритма лежит внутри рассматриваемого пояса шириной $2D+1$ (число делеций может быть и больше D).

0	4	9	14	19	24	
	2	7	12	17	22	
D	1	6	11	16	21	…
	3	8	13	18	23	
2D	5	10	15	20	25	

Задача позволяет ограничиться маленькими долями

Тогда кроме указанного алгоритма нами предложен **альтернативный алгоритм**: **разбиение остовного дерева на поддеревья**. Кластер – множество слов, приписанных листьям в финальном наборе деревьев

Исходный граф строится проще: локальным выравниванием всех «специальных слов» (белков из данной последовательности, методом blast).

После чего, оставляем ребра с весами выше порога.

Работа выполнена совместно с
Л. Рубановым и А. Селиверстовым.

Частично поддержана РФФИ
(проект 13-04-40196-Н)

СПАСИБО за внимание