Математическая биология и дискретная оптимизация

В.А. Любецкий, К.Ю. Горбунов

мех-мат МГУ, кафедра:

Математической логики и теории алгоритмов

http://logic.math.msu.ru/staff/lyubetsky/

http://logic.math.msu.ru/staff/lyubetsky/mb/

Институт проблем передачи информации РАН

лаборатория:

Математических методов и моделей в биоинформатике

http://lab6.iitp.ru/

весна 2022

12-ая лекция курса 2021-22:

Определения структуры и

двух наборов операций,

постановка

задач Преобразования и Реконструкции.

Перевод 2й задачи на языке паросочетаний.

Основные статьи по этим двум Задачам:

K.Yu. Gorbunov, V.A. Lyubetsky.

Linear time additively exact algorithm for transformation of chain-cycle graphs for arbitrary costs of deletions and insertions.

Mathematics, 2020, 8, 11;

И

Multiplicatively exact algorithms for transformation and reconstruction of directed path-cycle graphs with repeated edges. Mathematics, 2021, 9, No. 20, Art. 2576.

Их русские варианты будут выложены.

Ещё ссылки по задаче преобразования:

Горбунов К.Ю., Любецкий В.А. Почти точный линейный алгоритм преобразования графов из цепей и циклов, с оптимизацией суммы цен операций // Доклады Академии наук, 2020, том 494, № 6, стр. 26–29. (только формулировки)

K.Yu. Gorbunov, V.A. Lyubetsky. Linear time additively exact algorithm for transformation of chain-cycle graphs for arbitrary costs of deletions and insertions. Mathematics, Nov 10 2020, Vol. 8, No. 11, Art. 2001, 30 pp. (сложное доказательство)

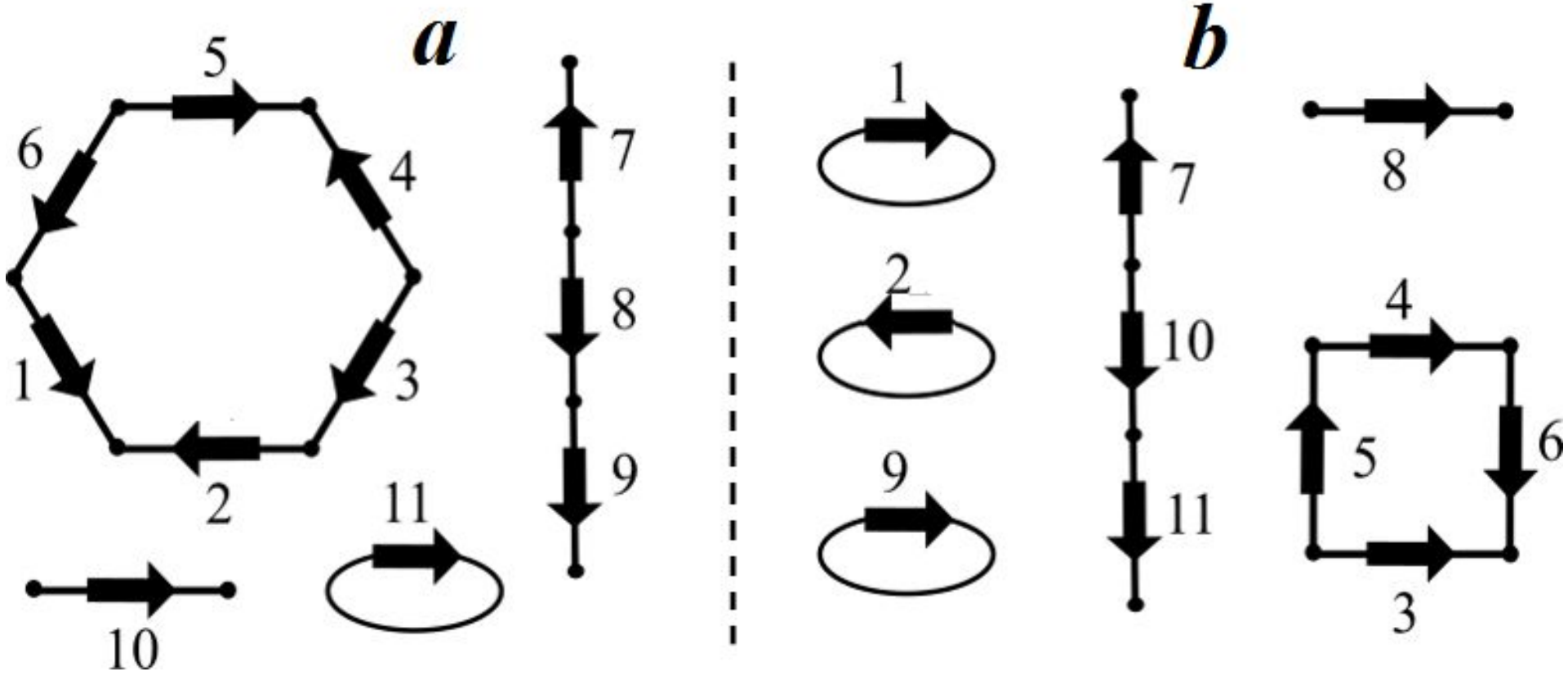
K.Yu. Gorbunov, V.A. Lyubetsky. An almost exact linear complexity algorithm of the shortest transformation of chain-cycle graphs. Eprint, **arXiv**:2004.14351, Apr 29 2020. (много полезных рисунков)

K.Yu. Gorbunov, V.A. Lyubetsky (2017). Linear algorithm of the minimal reconstruction of structures. Probl of Inform Transmission 53(1):55–72. (особо просто и на русском)

Определение. **Структура *a*** (или ***b***, ***c***, …) – ориентированный граф, состоящий из циклов (включая петли) и цепей длины ≥1 с **именами** рёбер (=натуральными числами или иными идентификаторами). Т.е. структура – нагруженный ориентированный граф с вершинами степени 1 или 2.

ЗАДАЧА ПРЕОБРАЗОВАНИЯ. Фиксирован набор операций, см. его ниже. Даны переменные цены (>0) этих операций и переменные структуры ***a*** и ***b***. Найти (алгоритмом линейной сложности) последовательность ***X**** операций (можно с их повторениями) от ***a*** к ***b***, на которой достигает минимума суммарная цена ***X**** (**по всем последовательностям *X* от *a* к *b***). Такую последоват-сть ***X****, как и её цену ***c***(***X****), назовём **кратчайшей**.

Пример структур ***a*** и ***b*** (преобразование идёт именно от ***a*** к ***b***). В вершинах присутствуют **склейки двух краёв**, кроме краёв цепей, в которых нет склейки. Для ребра 5 имена краёв 5_1 и 5_2. Нужно задать операции разрешённые в построении преобразования ***a*** в ***b*** (см. их ниже):



Первый набор операций, **SCJ**, в Задаче преобразования:

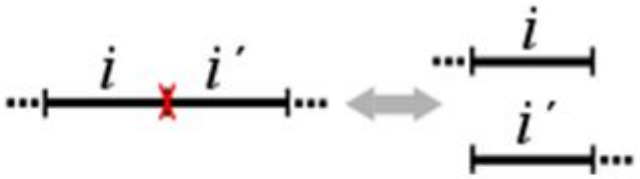
расклеить склейку или, обратная операция,

склеить два свободных края (одинарные переклейки),

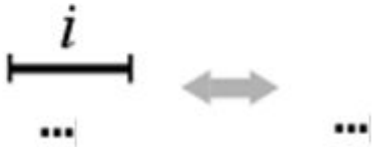
удалить изолированное ребро,

добавить изолированное ребро.

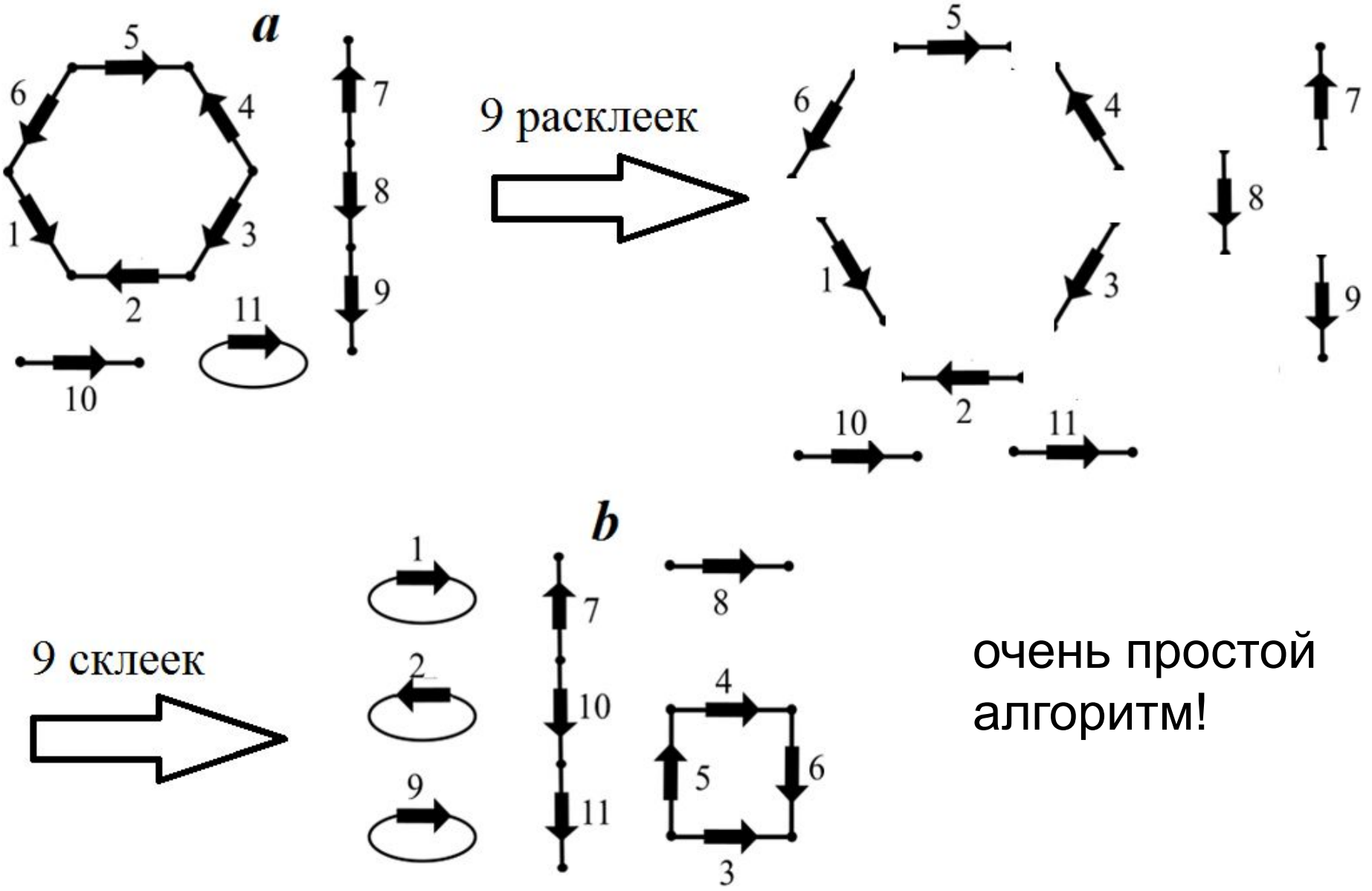
a) Расклейка и склейка



b) Удаление и добавление ребра



SCJ-преобразование для структур ***a*** и ***b*** выше:



очень простой алгоритм!

Алгоритм *SCJ*-преобразования и *SCJ*-расстояние :

SCJ-кратчайшее преобразование ***a*** в ***b*** состоит в расклейке всех склеек в *a**b*, удалении всех изолированных рёбер из *a**b*, добавлении всех изолированных рёбер из *b**a* и добавлении всех всех склеек из *b**a*; в порядке перечисления. Здесь *X**Y* – множество склеек или рёбер из *X*, не принадлежащих *Y*.

SCJ-расстояние между *a* и *b* заранее известно: оно равно взвешенной сумме (т.е. сумме с учётом цен операций) числа склеек в *a**b*, чисел рёбер в *a**b* и в *b**a* и числа склеек в *b**a*.

Структуры ***a*** в ***b*** называются с равным составом, если множества их имён (=их рёбер с их именами) совпадают.

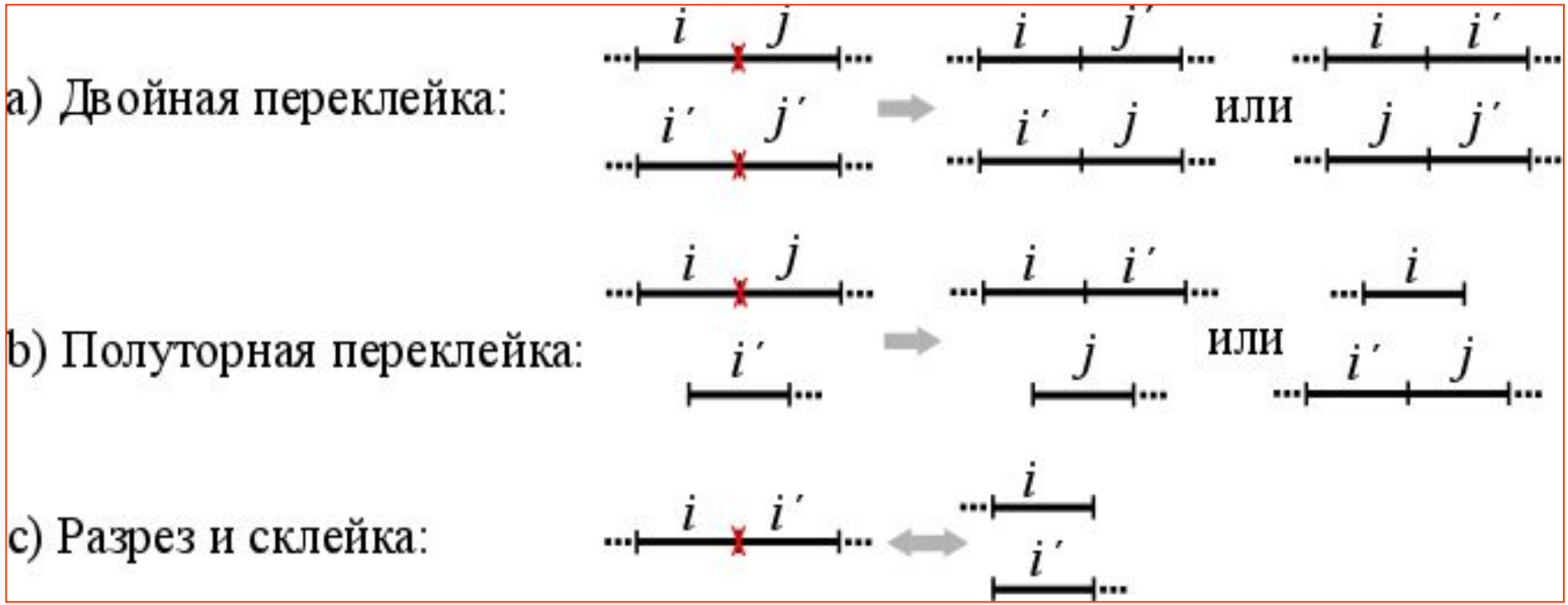
Задача: описать *SCJ*-преобразование и *SCJ*-расстояние между *a* и *b*, если *a* и *b* с равными составами.

Второй набор операций, **DCJ**, в Задаче преобразования:

расклеить две склейки и склеить четыре образовавшиеся свободных (т.е. степени 1) края рёбер по-другому (двойная переклейка);

расклеить одну склейку и склеить один из образовавшихся краёв ребра с каким-то свободным краем (полуторная переклейка);

расклеить склейку или, обратная операция, **склеить** два свободных края (одинарные переклейки).



Итак, 4 операции (будут ещё две):

расклеить какую-либо вершину (***Cut***), как было;

склеить два свободных (т.е. степени 1) края (***OM***), как было;

расклеить вершину (не край цепи) и **склеить** один из образовавшихся свободных краёв с уже свободным краем (***SM***);

расклеить две вершины (обе не края цепей) и **склеить** образовавшиеся свободные края (***DM***).

(***SM*** и ***DM*** – композиции ***Cut*** и ***OM***. В этом смысле для преобразования хватает только операций ***Cut*** и ***OM***).

Эти четыре операции названы ***DCJ-операциями***, см. figure 1 в нашей статье в **arXiv**. Они называются соот-но **разрез**, **склейка**, **полуторная переклейка** и **двойная переклейка**.

К этим четырём операциям добавляются

ещё **две операции**:

удалить (***Rem***) связный участок рёбер

с именами из *a*, но не из *b*;

вставить (***Ins***) связный участок участок рёбер

с именами не из *a*, но из *b*.

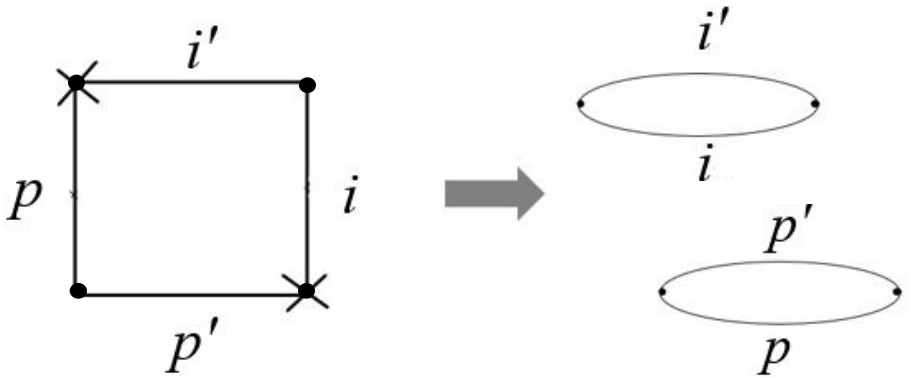
Полученные 6 операций называют также **DCJ**.

Примеры этих операций над структурой ***a***, которые цепочкой преобразуют ***a*** в ***b***:

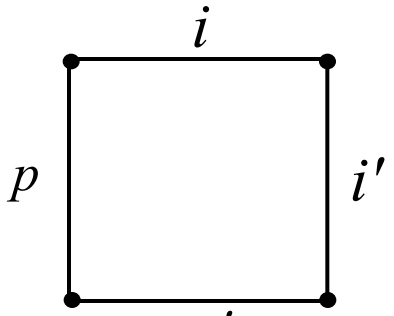
1) **удалить связный** (не обязательно изолированное ребро) **участок** рёбер с именами из ***a***, но не из ***b***,

2) **вставить связный** (как выше) участок рёбер с именами из ***b***, но не из ***a*** :

3) **Переклейка *DM***: расклеить две вершины и склеить по-другому образовавшиеся свободные вершины (= края ст 1). Пусть разрезы в **одном** цикле:

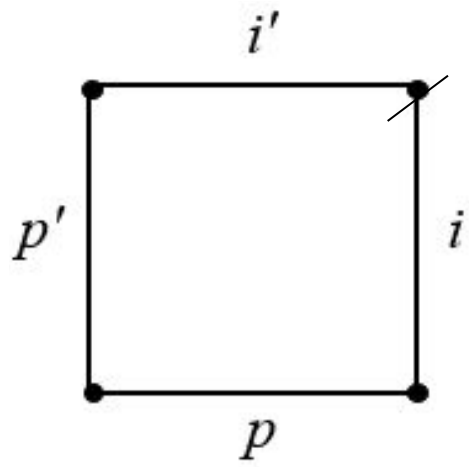


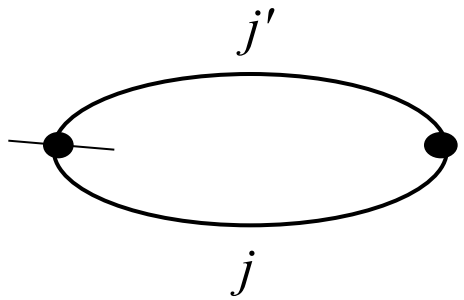
Вариант 1

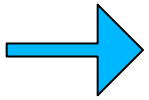


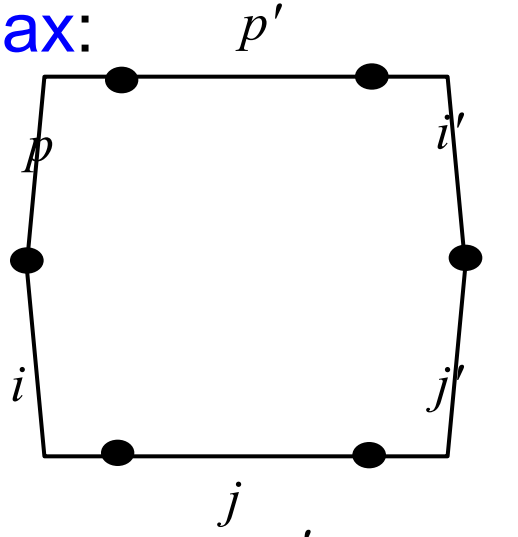
Вариант 2

Пусть разрезы в разных циклах:

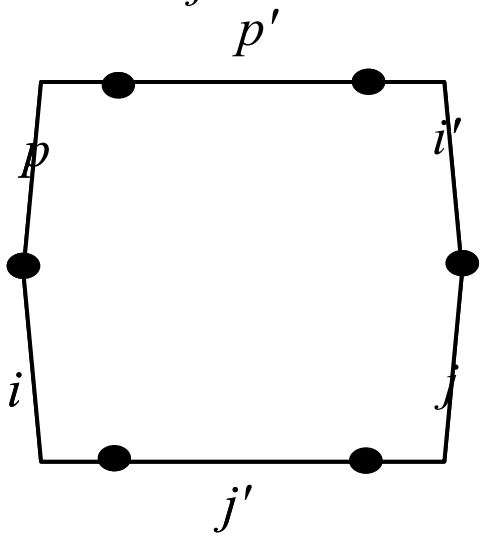








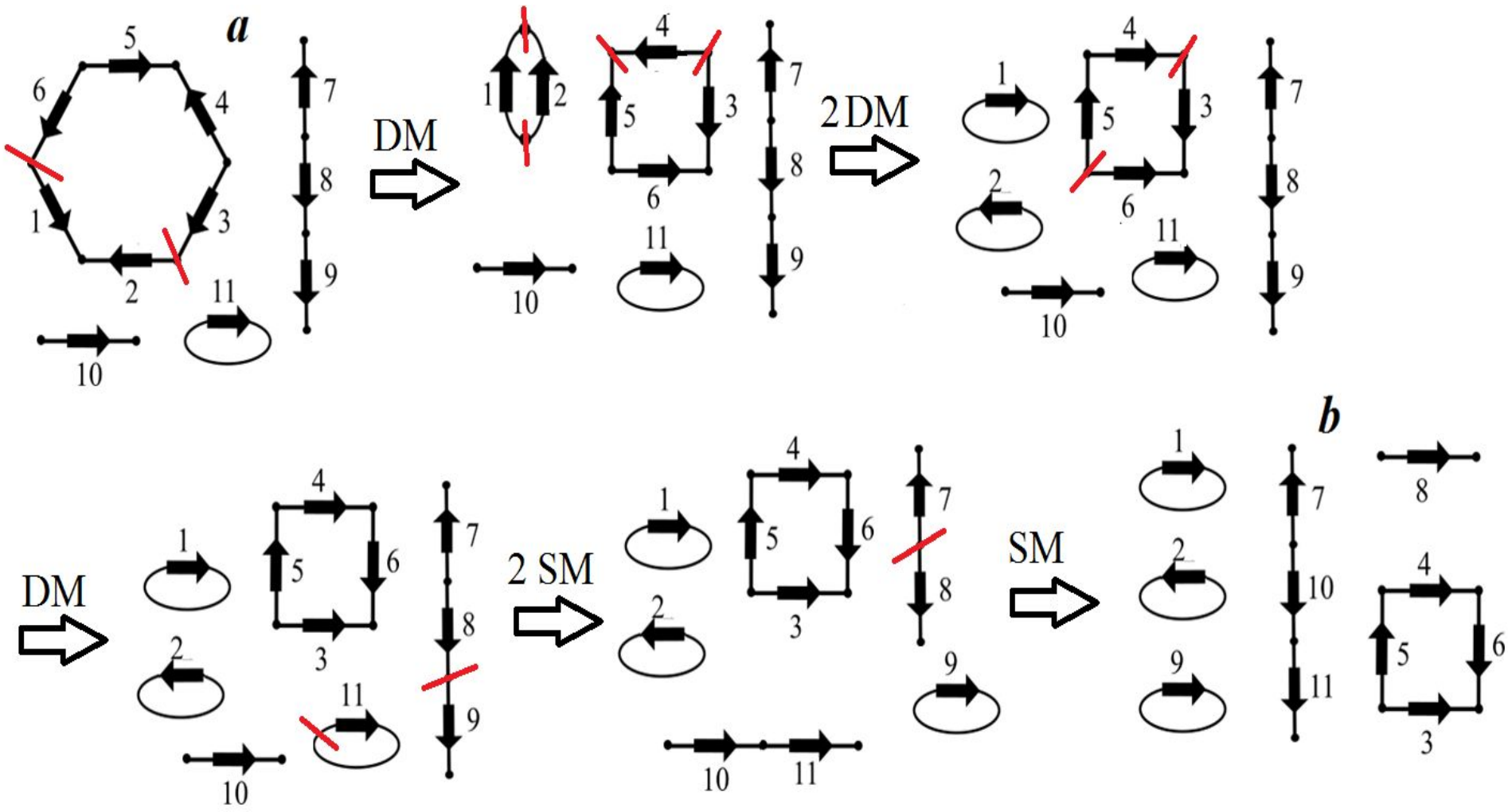
- вариант 1



- вариант 2

Каждой операции приписана **ЦЕНА** – строго положительное рациональное число. В Лекции 9 алгоритм приведён для равных цен, т.е. каждая операция имеет цену равную 1.

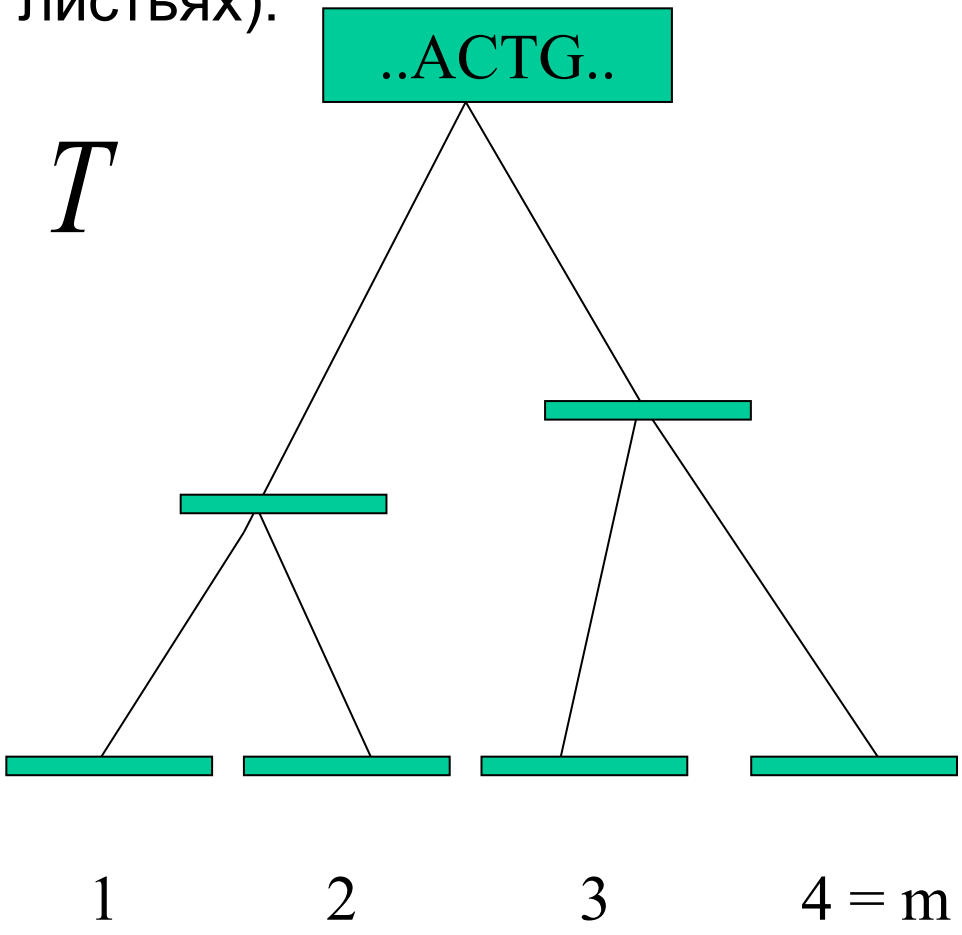
DCJ-преобразование для структур выше:



для **DCJ** алгоритм с равными ценами сложный (придумайте!), а с неравными ценами – проблема, не решённая в мировой науке!!

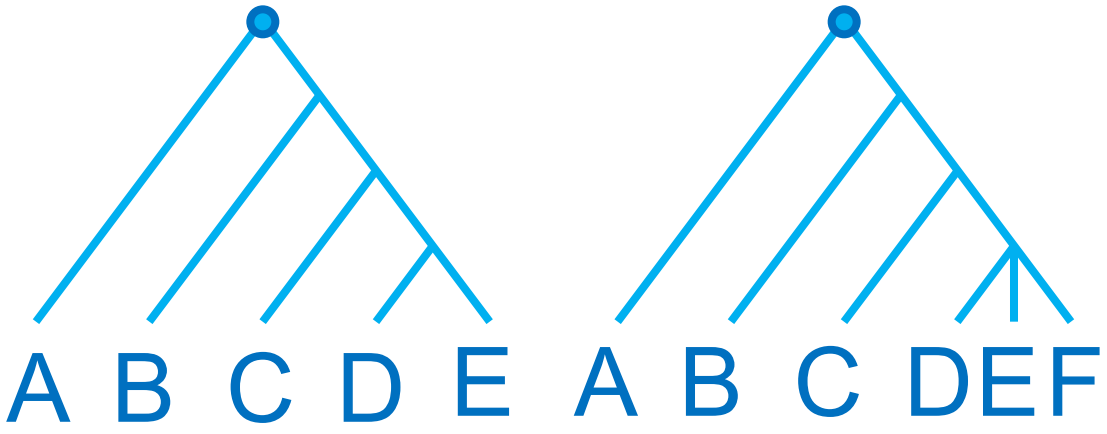
Понятие дерева с данными в листьях:

Дано дерево *G*, у которого **длины рёбер** соответствуют времени перехода от предка (в начале ребра) к потомку (в конце ребра); в листьях **даны** современные **последовательности** или **структуры**. Неизвестный праотец послед-ть или струк-а расположен в корне (известны **только! современные виды** – т.е. те, что в листьях):



Найти все предковые последовательности, включая самого праотца. Тем самым, найти эволюцию праотца, т.е. **предка** всех **современных данных** в листьях 1, …, 4=m.

А также: **найти само дерево** (=топологию) и **длины** рёбер.



На предыдущем слайде было «уравновешенное» дерево; а эти деревья называются «гребёнками».

Мы рассматриваем деревья с **корнем**. При наличии корня подразумевается **ориентация** рёбер: на каждом ребре задаётся направление от корня к листьям.

Важны и **неукоренённые** деревья – в них никакая из вершин не объявлена корнем.

Приведём всё это как **Математические определения**.

Неукоренённое дерево – связный ациклический граф любые две вершины (=узла) соединяет единственный путь. Докажите.

Укоренённое дерево – ориентированное дерево, в котором только одна вершина имеет нулевую степень входа (в неё не ведут рёбра), а все остальные вершины имеют степень входа 1 (в них ведёт ровно одно ребро). Вершина с нулевой степенью входа называется **корнем** дерева, вершины с нулевой степенью выхода (из которых не выходят рёбра) называются концевыми вершинами или **листьями**. Ориентация от корня к листьям.

Теория ЭВОЛЮЦИИ (**ФИЛОГЕНЕТИКА**) – это:

происхождение и развитие Живого в физическом или дискретном времени, его видов, частей его клеток («органелл»), отдельного белка, отдельного гена, отдельного регуляторного сигнала, участка ДНК, и т.д.

<Вид у половых организмов – множество особей, способных к взаимному скрещиванию, и пр. = множество близкородственных организмов = виды с попарно очень близкими ДНК.>

Главные проблемы:

каков **был** геном у **общего предка современных организмов**;

каков **был** геном у **первой клетки**,

у **первого многоклеточного организма,**

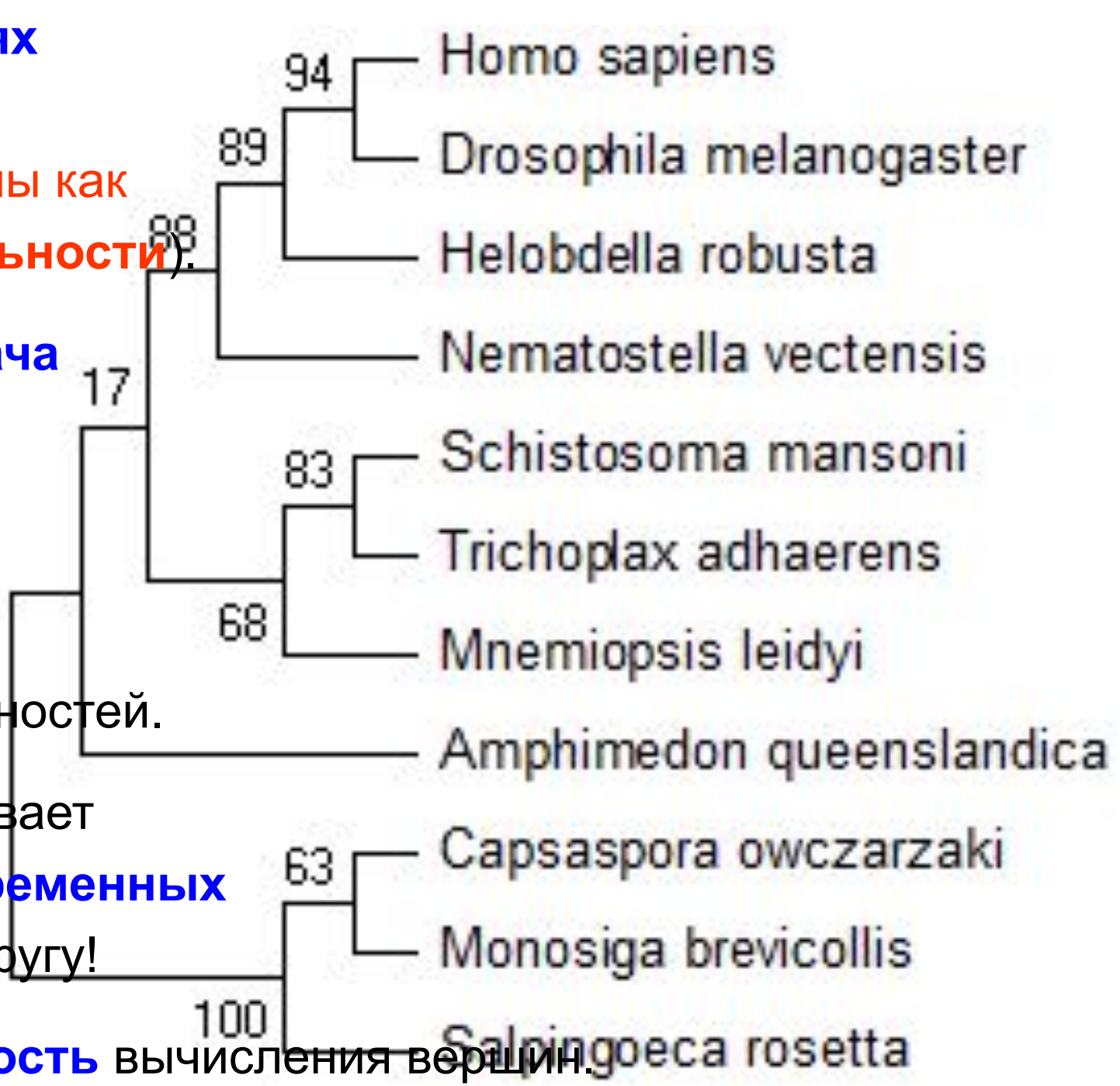
у **первого организма на суше**, и т.д.

Здесь в **листьях даны** виды (здесь их геномы как **последовательности**).

И решена задача построения самого дерева; здесь без предковых последовательностей.

Дерево показывает **близость современных видов** друг к другу!

И ещё **надёжность** вычисления вершин.



задача РЕКОНСТРУКЦИИ –

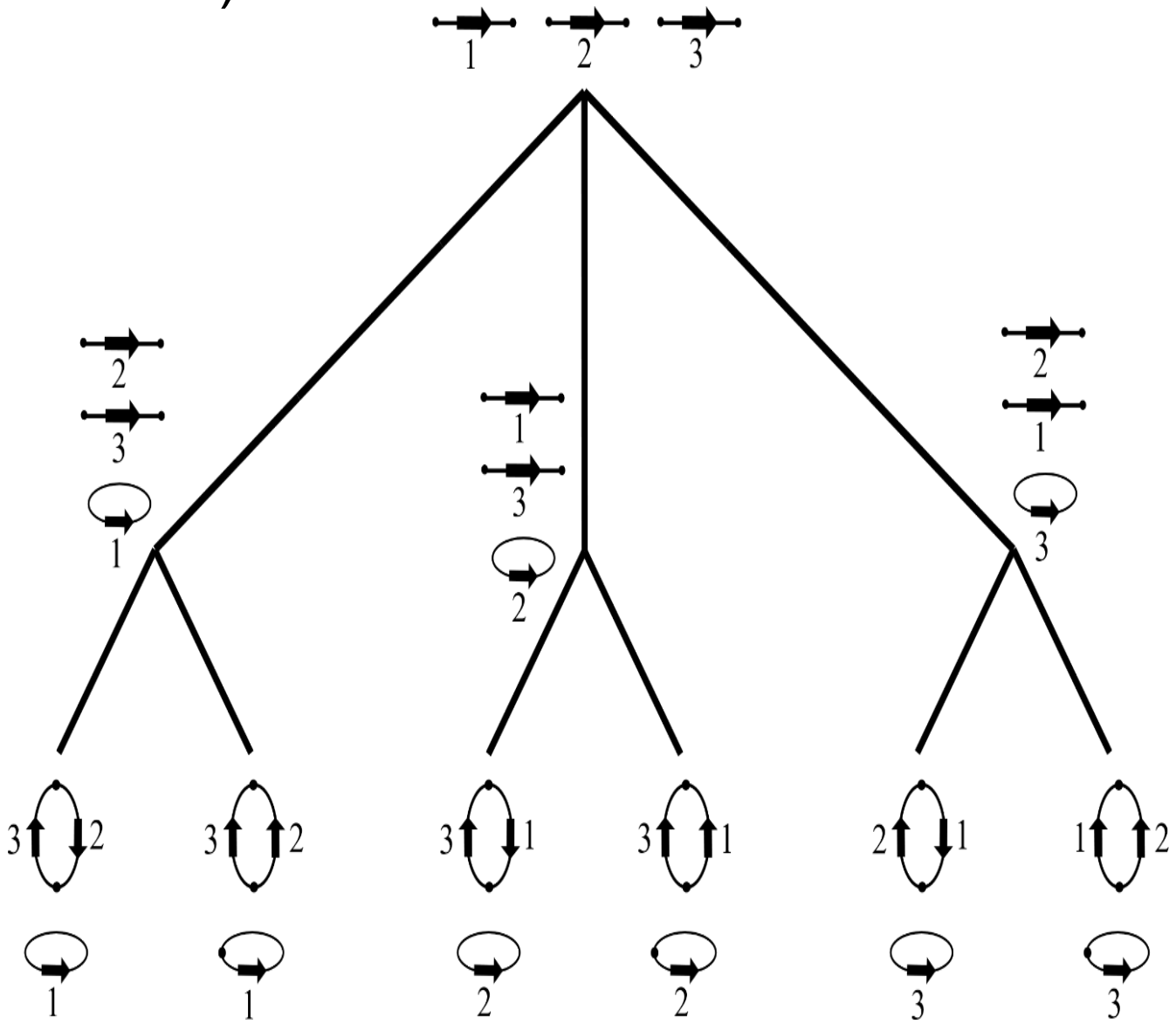
продолжить последовательности или **структуры**, заданные в листьях данного дерева, на его внутренние вершины (= **реконструировать** последовательности или соот-но структуры во внутренних вершинах дерева).

Далее рассмотрим **случай СТРУКТУР**.

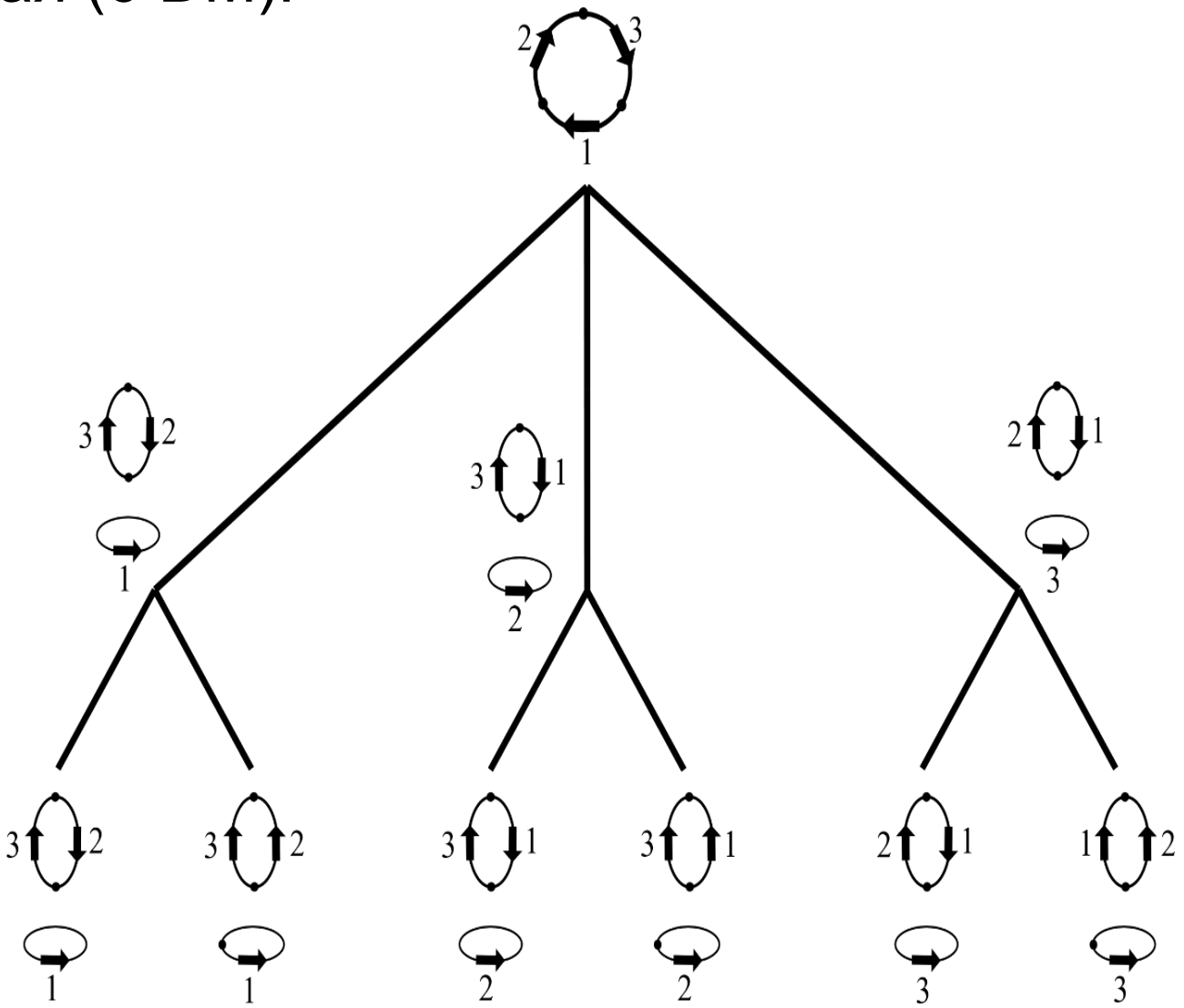
Расстановкой по дереву называется приписание структуры каждой внутренней вершине дерева. **Суммарной ценой** расстановки называется сумма *SCJ*-расстояний по всем рёбрам (или аналогично *DCJ*-расстояний).

Итак, ищем **кратчайшую расстановку**, т.е. ту, на которой эта суммарная цена минимальна (среди всех расстановок).

Пример 1. Расстановка по дереву при равных ценах операций, SCJ–кратчайшая (15 склеек); но DCJ–не-кратчайшая (15 склеек):



Пример 2. Расстановка по дереву при равных ценах операций с теми же данными в листьях; SCJ-не-кратчайшая (12 расклеек и 12 склеек);
но DCJ-кратчайшая (6 DM):



Варианты постановки задачи Реконструкции:

в листовых структурах разрешены повторения имён

(= т.е. там возможны паралоги) или их нет;

цены операций разные или равные.

Задача Преобразования – частный случай

задачи Реконструкции:

чтобы решать вторую нужно много раз решать первую.

Мы начнём с задачи Реконструкции и для неё

рассмотрим гораздо более простой случай **SCJ-расстояния** (так как его легче вычислять).

Итак, даны **дерево** и в каждом его листе **структура**.

(Удобно считать, что она **беспетлевая** ,т.е. не содержит петель. Или нужны равенства цена раск_пет = цене добавл, и аналогично для склейки.)

Лист ***v*** и его структура (также обозначаемая ***v***) **не различаются**.

Разрешённое ребро – то с его именем, которое входит в структуру одного из листьев.

Состав структуры – множество рёбер с их именами, которые в входят в структуру.

Переход к равенству составов в листьях выполняется так: в каждый лист (=в структуру листа) добавим *k*-петлю для каждого разрешённого ребра *k*, отсутствующего в этом листе. В результате составы листьев выравняются. По всему дереву (=в любой расстановке) рассматриваем структуры, которые включают все разрешенные рёбра и только их.

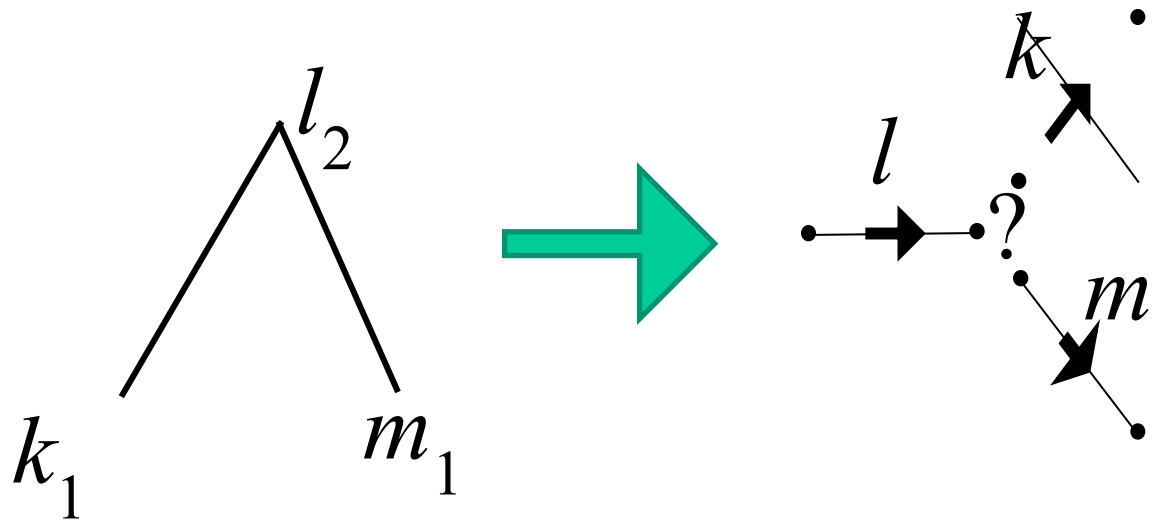
Расклейка петли (с образованием изолированного ребра) имеет цену добавления ребра;
склейка краёв изолированного ребра (с образованием петли) имеет цену удаления ребра.

Для данной структуры $\mathcal{a}$ обозначим M множество краёв рёбер из $\mathcal{a}$, т.е. M состоит из имён краёв вида k_1 и l_2 (эти имена назовём *точками*). Пусть M полный граф, который состоит из этих точек и любые две разные точки соединены ребром. По $\mathcal{a}$ определим подграф P в M: склейку краёв k_1 и l_2 изобразим ребром между точками k_1 и l_2. Итак, по $\mathcal{a}$ построили P. В P никакие два ребра не пересекаются их краями, т.е. в P никакие два ребра не инцидентны. И наоборот: по любому подграфу P (в M) с таким свойством однозначно образуется структура $\mathcal{a}$ (с краями из M). Подгр-аф P в M с таким свойством называется *паросочетанием*.

Переход от структуре ***a*** к подграфу ***P***, который

показывает склейки краёв в ***a***:

два ребра в ***P*** не могут быть инцидентны, поскольку каждый край (здесь l_2) не может быть склеен с двумя краями (а может только с одним или ни с каким):

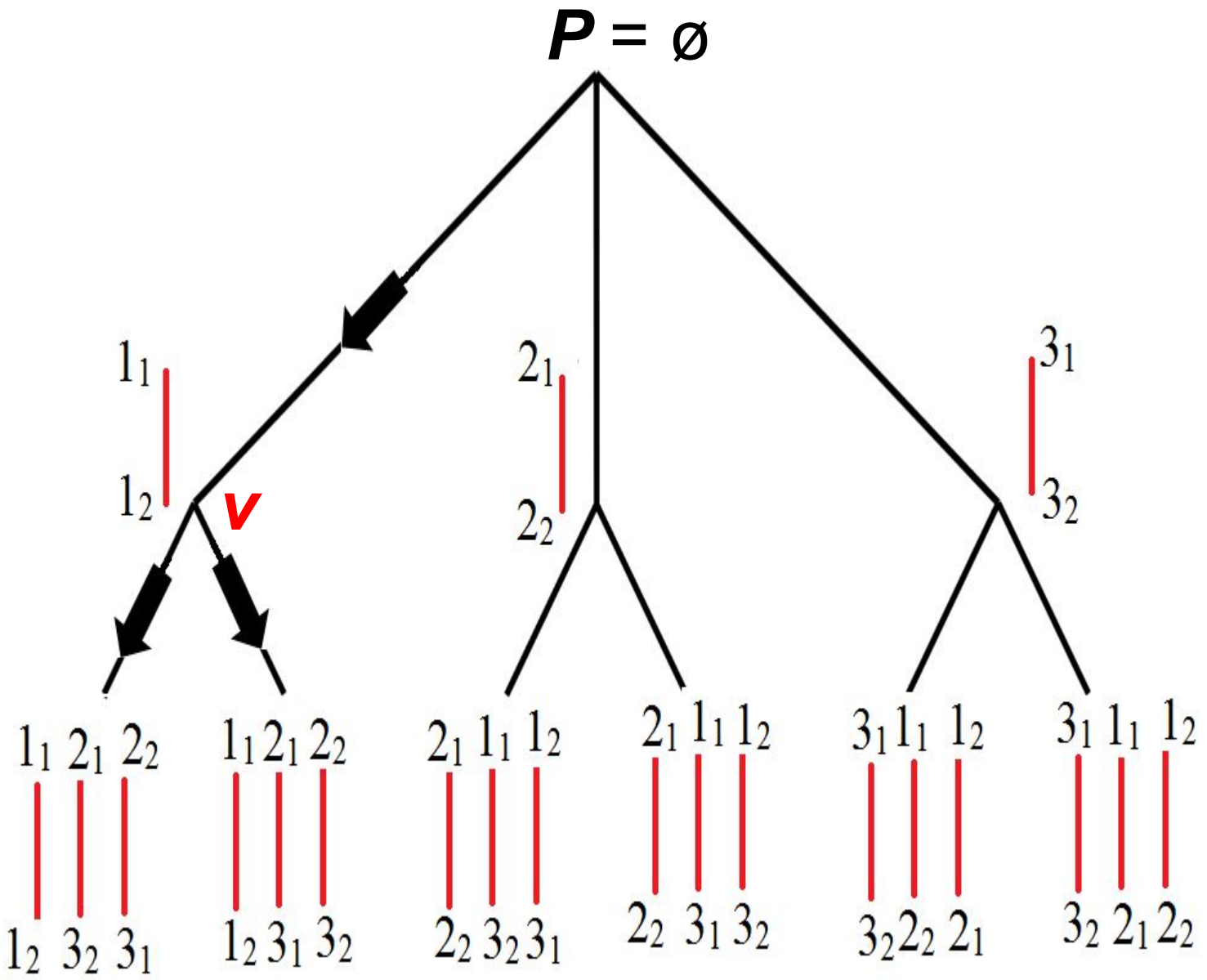


предположительно в ***P***

тогда в соот-ей «структуре» ***a*** будет вершина степени 3, что запрещено.

к Примеру 1. Та самая расстановка из паросочетаний.

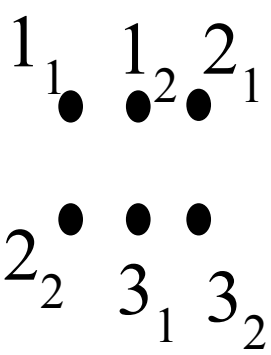
Рёбра звезды в вершине ***v*** показаны стрелками:

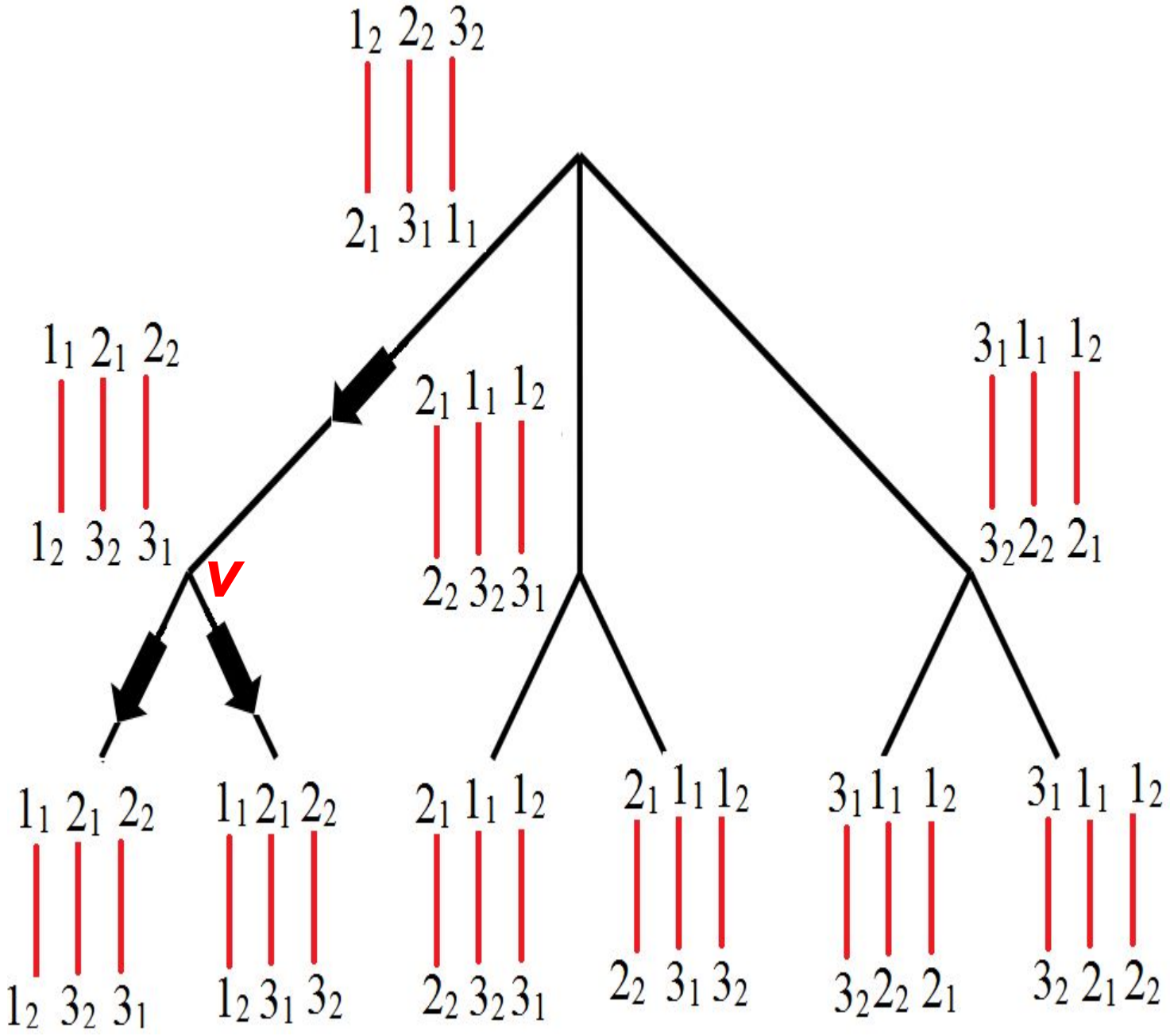


к Примера 2. Аналогичный переход к паросочетаниям для его данных:

вершины в ***M***,

$P \subset M$





Локально минимальной назовём расстановку по дереву, для которой в любой одной вершине замена структуры на любую другую не строго уменьшает суммарную цену расстановки.

Задача: кратчайшая расстановка является локально минимальной, но не наоборот. При равных ценах SCJ-операций все ли структуры в Примерах 1-2 находятся в локальном минимуме относительно их звёзд? В каждом из Примеров 1-2 указать такие цены SCJ-операций и вершину, в которой структура не находится в локальном минимуме относительно её звезды. □

Ответ: цены склейки 3, расклейки 1. В Примере 2 при единичных ценах присутствует не локально минимальная структура. Это – корень.

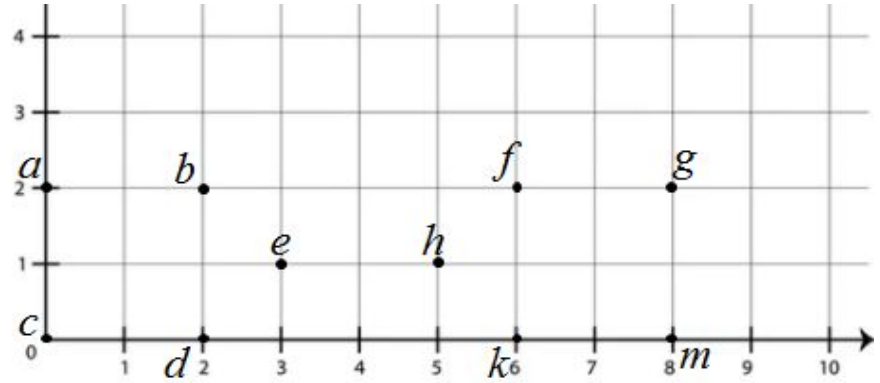
Итак, дан неориентированный без петель полный граф ***M***, в нём каждому ребру в дальнейшем приписано натуральное или рациональное число («вес/длина ребра»), т.е. ***M*** будет! нагруженный.

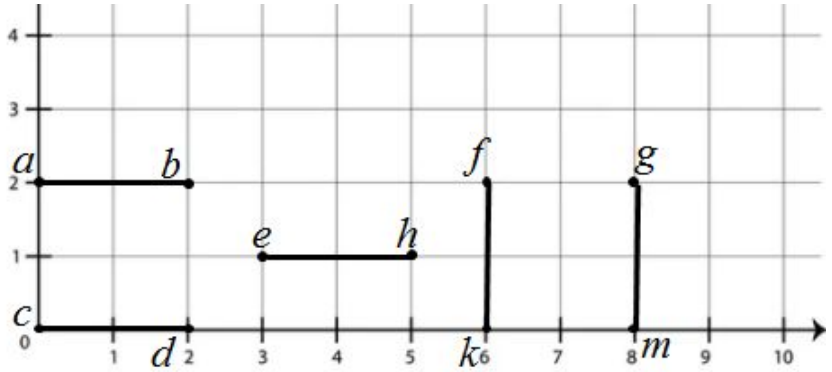
Паросочетанием называется подграф этого графа, в котором никакие два ребра не инцидентны.

Полным называется паросочетание, в котором участвуют все вершины из ***M***.

Максимальным называется паросочетание, на котором достигает максимума сумма длин его рёбер со **строго положительной длиной**.

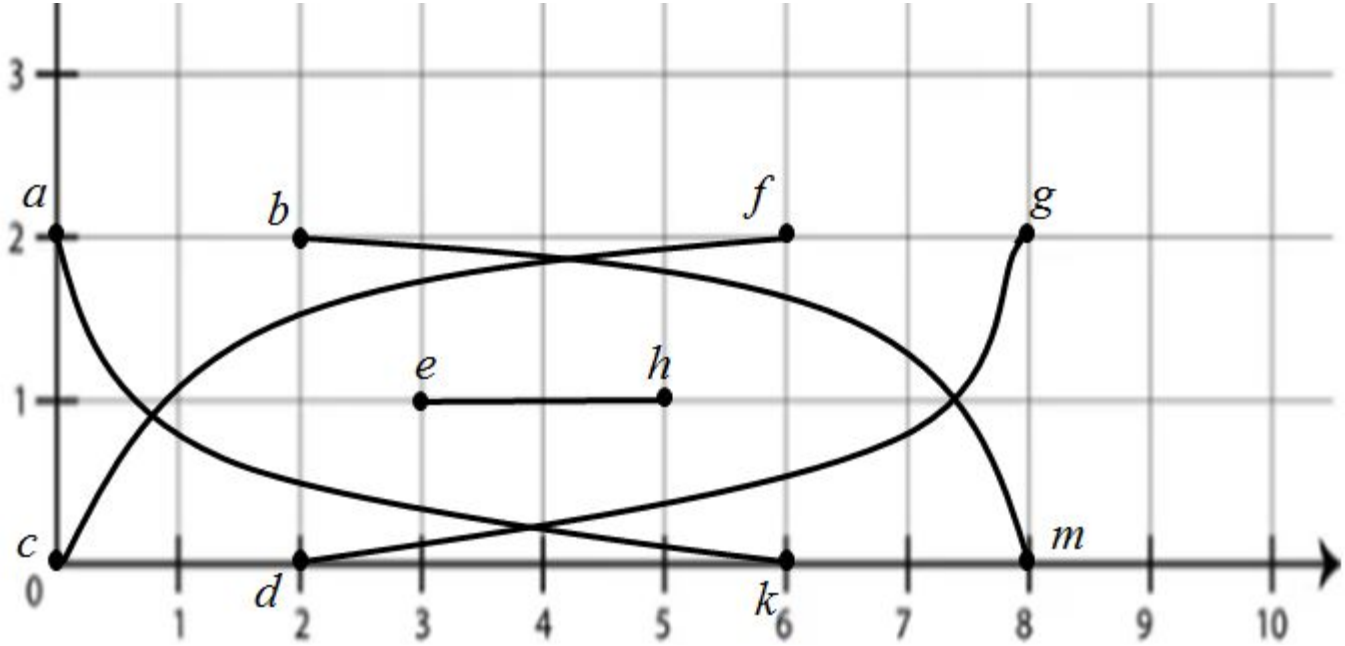
Пример графа *M*: на плоскости даны 10 точек и вес каждой пары разных точек (=ребра) равен расстоянию между ними (см. слева):





Минимальное полное паросочетание имеет вес 10 (см. справа).

Максимальное паросочетание: здесь полное, хотя это не обязательно, и имеет вес $2+8\sqrt{10}$ (см. определения ниже):



Минимальным называется полное паросочетание, на котором достигает минимума сумма длин его рёбер.

Минимальное (автоматически: полное) паросочетание может включать ребра любого веса.

А максимальное паросочетание только строго положительного веса.

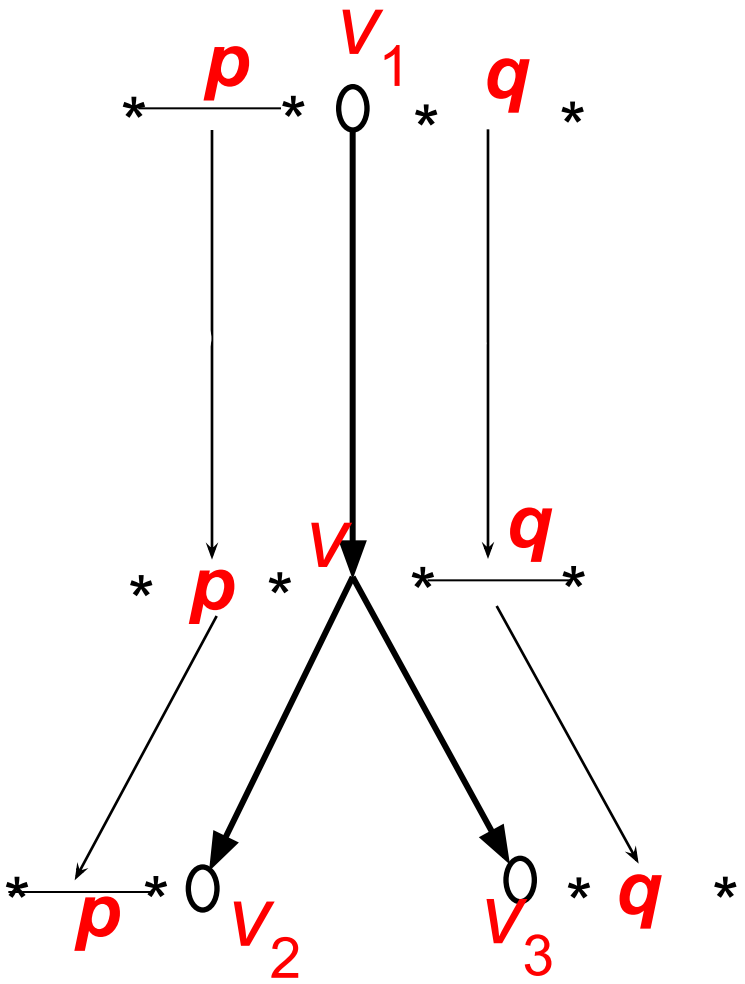
Даны дерево ***T*** и расстановка по ***T***.

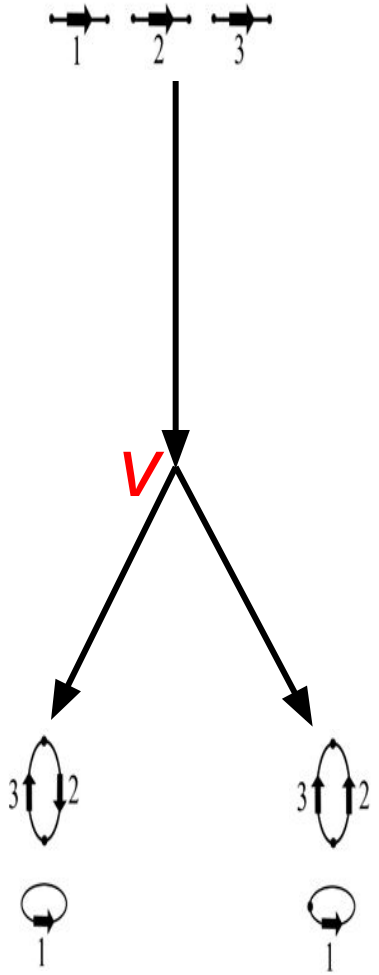
Звездой называется подграф, состоящий из какой-то вершины ***v*** в ***T*** (***v*** называется *центром*) и вершин в ***T***, с которыми в ***T*** центр соединён, и самих этих рёбер.

Листья звезды назовём её *краями*.

Для звезды ***v*** образуем полный граф ***Mv***, состоявший из всех пар разных краёв в ***v***. (Если 3 ребра-непетли, то в *Mv* 6 кра и 15 рёб.)

Важны паросочетания в краях звезды, паросочетание в её центре не используется (=его как бы нет там) !!





без учёта цен

$$p_{no} = 0 + \begin{cases} 2 & ,если \quad 1_1 - 1_2 \\ 1 & ,если \quad 2_i - 3_j \ \forall i,j \\ 0 & ,другая \quad пара \end{cases}$$

$$q_{yes} = 1 + \begin{cases} 0 & ,если \quad 1_1 - 1_2 \\ 1 & ,если \quad 2_i - 3_j \ \forall i,j \\ 2 & ,другая \quad пара \end{cases}$$

Пусть *p* пара разных краёв (=ребро) в ***Mv*** (см. сл. 31-32):

в предположении ***p*** *не* склеена в структуре ***v*** :

p_{no} – число расклеек на **входящем** ребре, которые нужно сделать на нём, + число склеек, которые нужно сделать на **выходящих** рёбрах, с учётом каждой цены;

<чтобы привести к состоянию внизу !! >

в предположении ***q*** склеена в структуре ***v*** :

q_{yes} – число склеек на **входящем** ребре + число расклеек на **выходящих** рёбрах, с учётом каждой цены.

Значения p_{no} и q_{yes} не зависят от структуры в ***v*** !!

Задача. Для примеров 1-2 и звезд в в них вычислите значения p_{no} и p_{yes} (все без данных в центре звезды). □

Для примера 1 и в нём звезды v и ребра $q = 1_1–1_2$ (с данными о склейках в центре звезды)

имеем q_{no} =2, q_{yes} =1, $q_{yes} – q_{no} = – 1$;

S_0 = суммарная цена такой расстановки по этой звезде:

□ склеек в центре и те же данные на краях; получим = 6;

суммарная цена этой звезды 5 = $q_{yes} – q_{no} + S_0$. □

При практическом вычислении суммарной цены S любой расстановки лучше использовать просто сумму SCJ-расстояний, которая вычисляется легко. □

Лемма 1. Для звезды в ***v*** (как отдельного дерева ***T***)

S равна

$$S \to \min$$

$$[\sum_{q \text{ квазиклики}} q_{yes}(v) - q_{no}(v)] + S_0$$ где S_0 = суммарная

цена такой расстановки: □ склеек в центре и те же данные на краях (S_0 константа!, а ищем **min** для звезды).

$$\Leftrightarrow (*) \sum_{q \text{ квазиклики}} q_{no}(v) - q_{yes}(v) \to \max$$

таким образом, max по индексам суммирования {*q*}, который образуют паросочетание *P* .

Лемма 2. Суммарная цена ***S*** исходной расстановки равна

$$\frac{1}{2}[\sum_{\text{неребра } P} p_{no}(v) + \sum_{v, T \setminus q \text{ склеены}} q_{yes}(v)] \to \min$$

Алгоритм для звезды в ***v*** :

Припишем каждому ребру ***q*** в ***Mv*** вес $q_{no}(v) - q_{yes}(v)$, и ищем максимальное паросочетание в этом нагруженном графе. Слагаемые в (*) как раз суть рёбра в искомом паросочетании!

Таким образом, **роль паросочетаний в том, чтобы выполнить максимизацию в (*); и всё**. □

Лемма 3. Если расстановка такова, что в центре каждой звезды находится максимальное паросочетание (относительно её краёв), то расстановка локально минимальная; и наоборот. □

Итак, наша задача: найти низкой сложности алгоритм, который строит локально минимальную расстановку – по дереву и по данным в листьях.

Пусть уже дана некоторая **НАЧАЛЬНАЯ расстановка**.

В начальной расстановке переберём все звёзды, и для каждой (с центром в ***v***) вычислим

$\Delta(v) = c(P) - c(P_1)$, где P исходные паросочетания в ***v***,

а P_1 новые (максимальное в центре) паросочетания в ***v***;

и ***c***(P) суммарная цена расстановки.

Если $\Delta(v)=0$ во всех вершинах ***v***, то

алгоритм кончил работу.

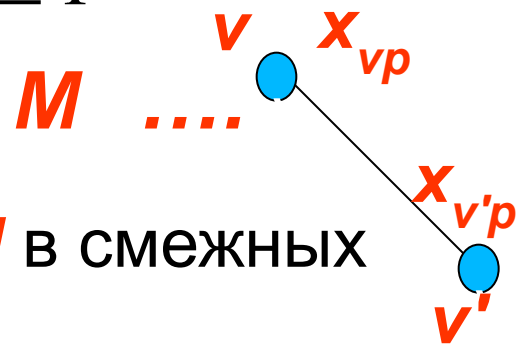
Иначе среди всех вершин найдём ***v1*** с наибольшим □ (***v1***). В ***v1*** подставим максимальное паросочетание P_1 для краёв звезды ***v1***. Вычислим новые □(*v*) для звёзд с центром в *v*, которые пересекаются по ребру со звездой в ***v1***. Остальные □(*v*) оставим прежними.

Повторим итерацию. □

Число уменьшений/число шагов алгоритма не превысит цену $c(\overline{P}_0)$ начальной расстановки, если считать пометки пар натуральными числами (что не уменьшает общности), так как она уменьшается на каждом шаге.

Остался вопрос, как найти **НАЧАЛЬНУЮ расстановку**?

Каждой (неупорядоченной) паре ***p*** различных краёв рёбер в **вершине *v*** исходного дерева сопоставим **вероятность x_{vp} : «пара *p* входит в искомое паросочетание *P* в *v*», т.е. края из *p* склеены в *v***. Наложим ограничения $0 \le x_{vp}$. В листе эти вероятности известны: 1 на склеенной ***p*** и 0 на несклеенной ***p***. В каждой внутренней *v* и любых краёв k_i-l_j наложим ограничение: $\sum_{l_j \neq k_i} x_{vk_il_j} \le 1$.



Соседними назовём пару $p = k_i l_j$ краёв в ***M*** в смежных вершинах ***v*** и ***v'*** дерева и соответствующую им пару переменных x_{vp} и $x_{v'p}$. Целевфу $F = \sum_{v,v',p} (x_{vp} - x_{v'p})^2 \to \min$

Минимизация ***F*** при описанных ограничениях – задача выпуклого квадратичного программирования, которая стандатно решается.

Положим: вероятности $\{x_{vp}\}$ – это веса в вершине ***v***, т.е. веса в полном графе ***M***. **С этими весами** в каждой внутренней вершины ***v*** (независимо от других вершин) решим задачу построения максимального взвешенного паросочетания ***P***.

(На итерации веса брали с краёв звезды.)

Эти паросочетания (решения выпуклого программирования) образуют **НАЧАЛЬНУЮ РАССТАНОВКУ**.