Построение и анализ алгоритмов

Лекция 2

Метод ветвей и границ

1. Общая схема
2. Задача коммивояжера

Метод ветвей и границ
Branch-and-Bound Method

Вариант поиска с возвращением.

Каждое решение имеет *стоимость*.

Требуется найти *решение минимальной стоимости*.

Примечание. Поиск с возвращением применялся в ситуациях, где надо было найти *хотя бы одно*, либо *все существующие решения* комбинаторной задачи.

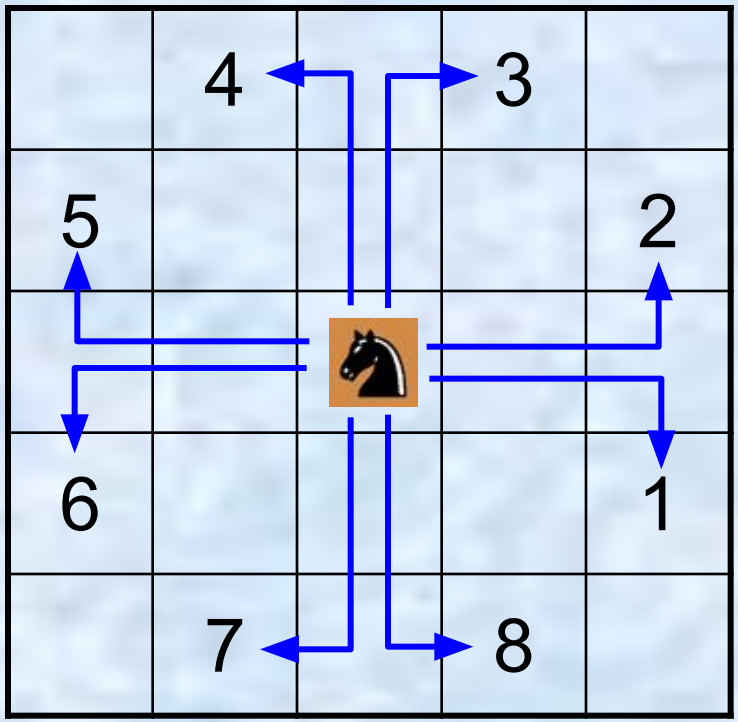
Теперь же речь идет об ***оптимизационных*** комбинаторных задачах, в которых надо найти (выбрать) из возможных решений ***наилучшее*** по некоторому критерию.

Метод ветвей и границ

Branch-and-Bound Method

Пример: задача об оптимальном маршруте коня на шахматной доске

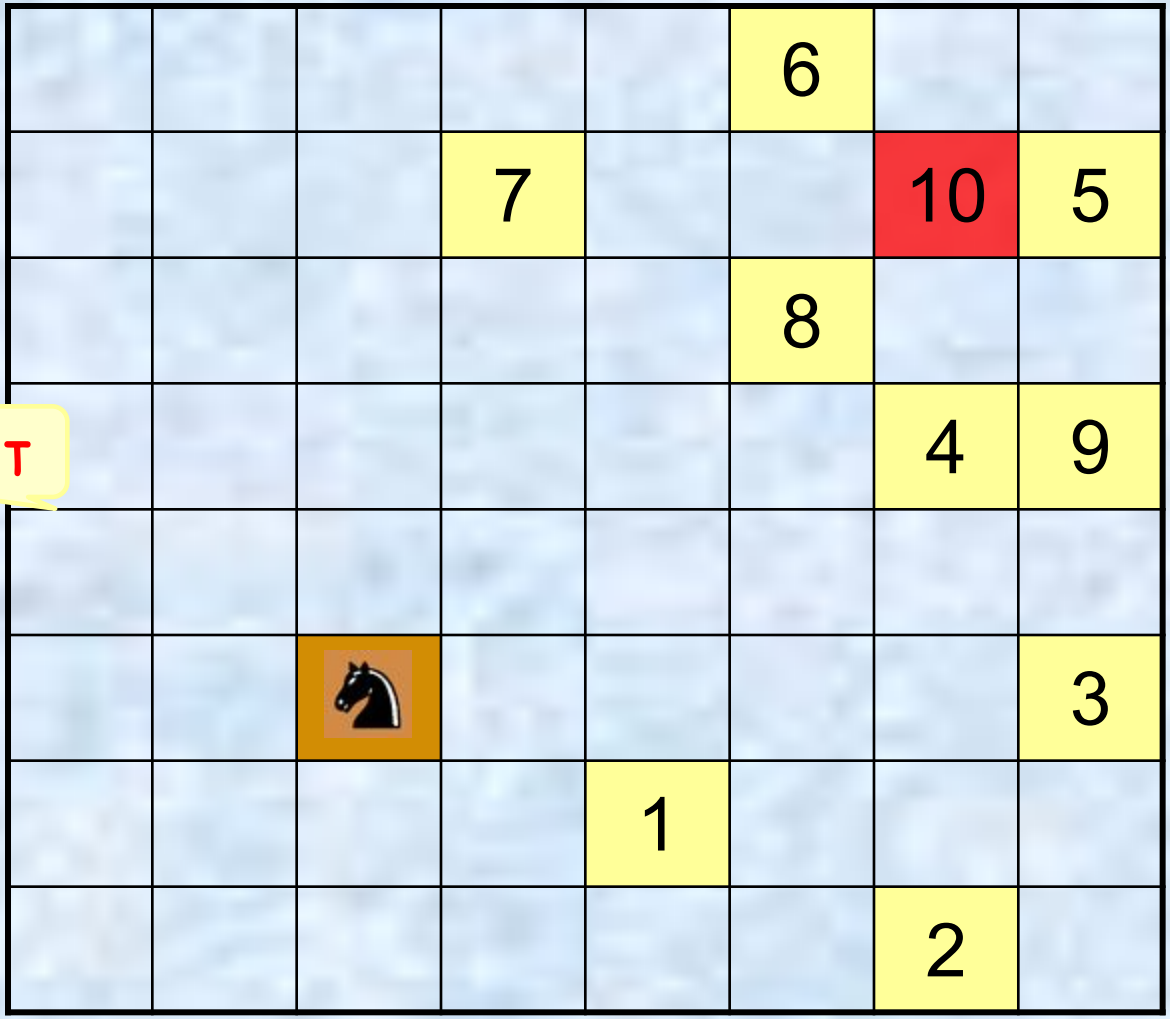
Найти путь коня из одного заданного поля в другое, содержащий минимальное число шагов



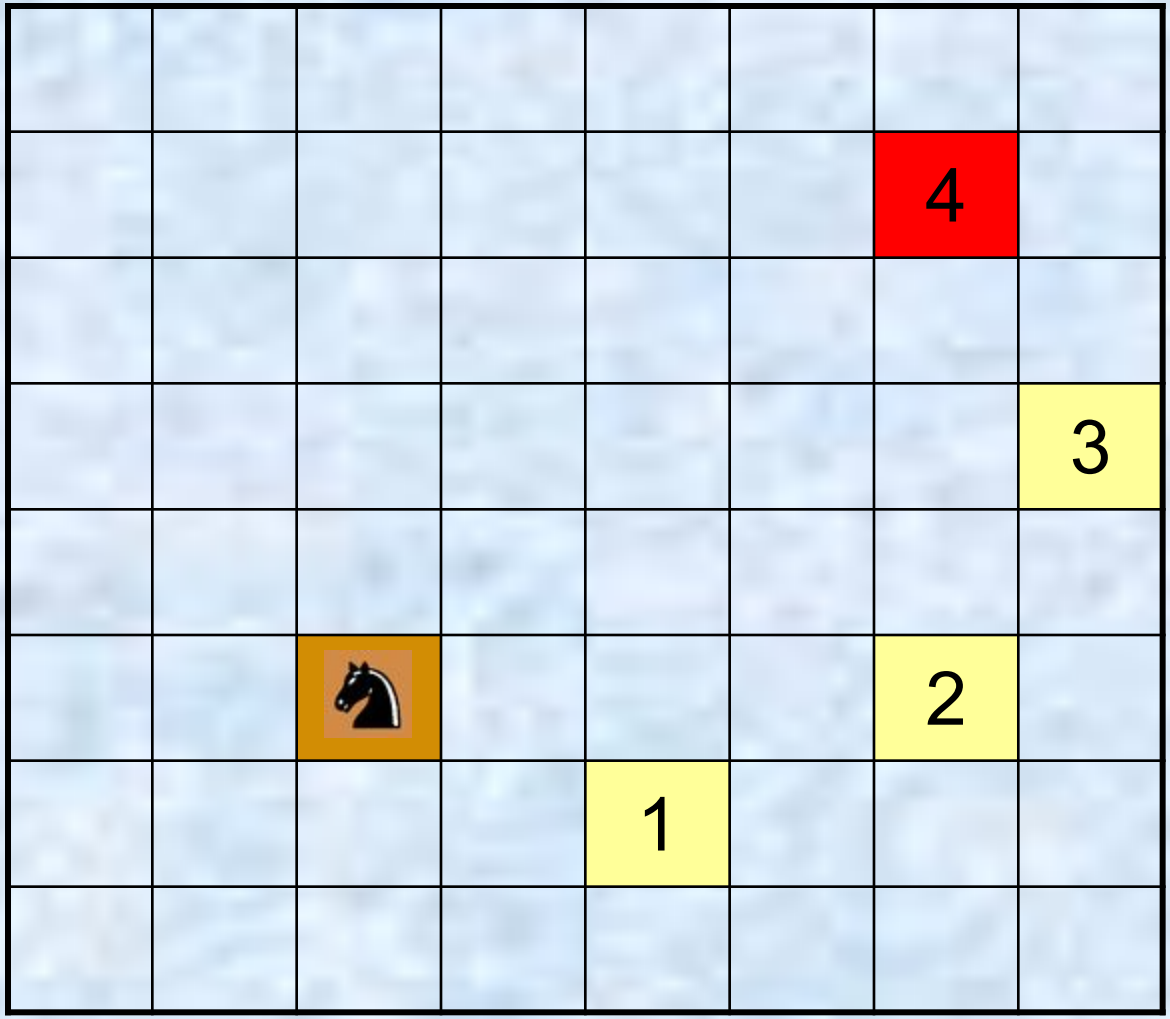
Некоторое решение

Финиш

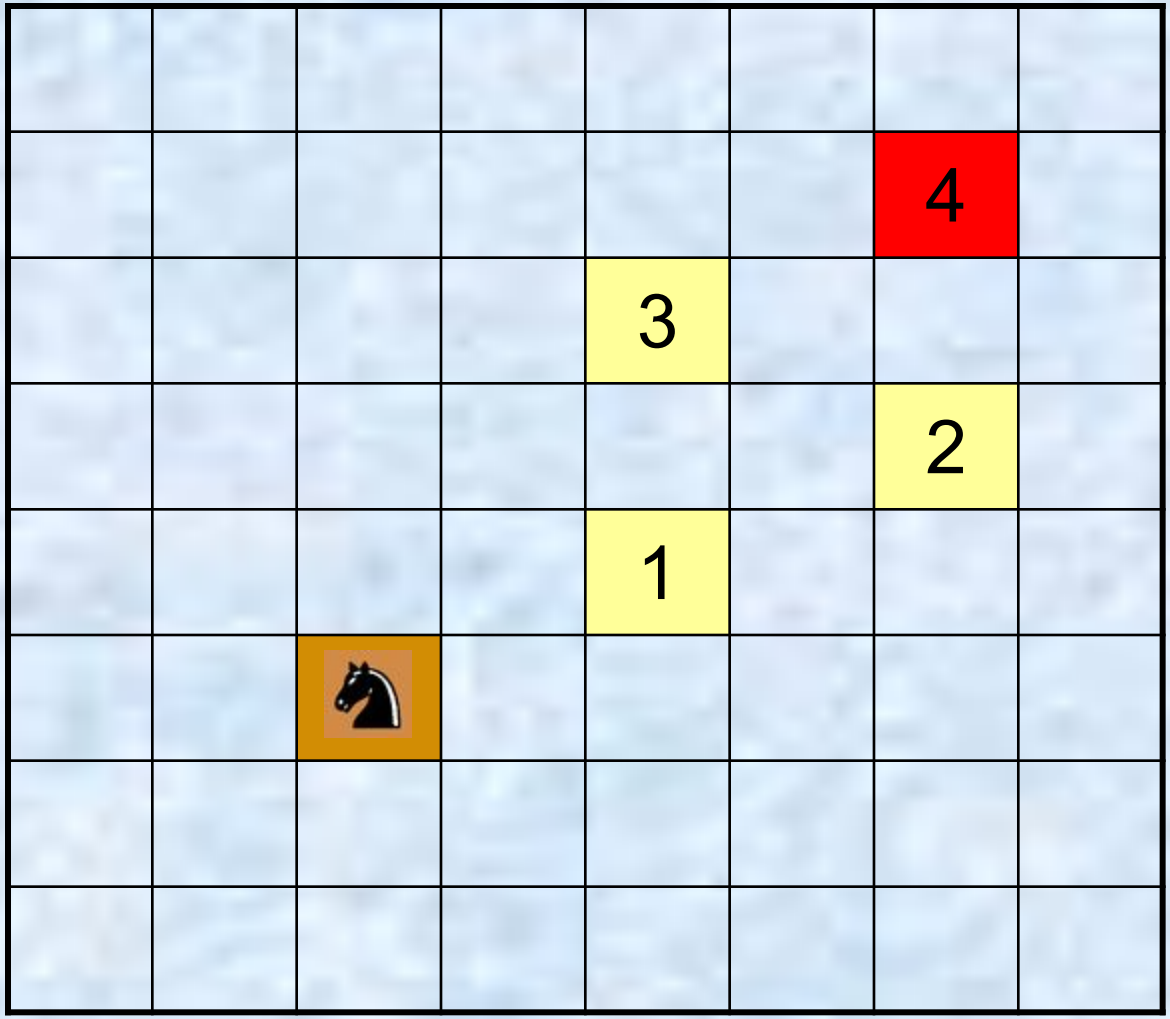
Старт



Лучшее решение



Лучшее решение



Условия применимости метода ветвей и границ (МВГ)

1. Для каждого частичного решения должна быть определена стоимость $Cost(a_1, a_2, \ldots, a_k)$
2. Для всех частичных решений и их расширений должно быть выполнено

$$Cost(a_1, a_2, \ldots, a_{k-1}) \le Cost(a_1, a_2, \ldots, a_{k-1}, a_k)$$

Тогда частичное решение $(a_1, a_2, \ldots, a_{k-1}, a_k)$ может быть отброшено, если его стоимость $\ge$ стоимости ранее вычисленных решений

В большинстве случаев $Cost(a_1, a_2, \ldots, a_k) \ge 0$ и даже

$$Cost(a_1, a_2, \ldots, a_k) = Cost(a_1, a_2, \ldots, a_{k-1}) + C(a_k),$$

где $C(a_k) \ge 0$ для всех a_k.

Общий алгоритм МВГ

$S_1 = A_1$; k = 1; cost = 0; LowCost = + ∞;
while (k>0) { //пока не все решения найдены
  while ((S_k != ∅) && (cost < LowCost)) { // продвижение вперед:
    a_k = элемент из S_k; //выбор очередного элемента из S_k
    $S_k = S_k - \{a_k\}$;
    cost = f_cost (a_1,…, a_{k-1},a_k);
    f (((a_1,…, a_{k-1},a_k) – решение) && (cost < LowCost)) { LowCost = cost;
     /*фиксировать (a_1,…, a_{k-1},a_k) как текущий минимум */}
     else {
          // переход к следующему уровню
              k ++;
              вычислить S_k;
     }
    } // end while продвижения вперед
    k --; // backtrack
    cost = f_cost (a_1,…,a_k);
} // последнее сохраняемое решение имеет наименьшую стоимость

Задача коммивояжёра (ЗК)
The Traveling Salesman Problem (TSP)

Коммивояжер – бродячий торговец – коробейник

Коммивояжёр [фр. *commis voyageur*] – разъездной агент торговой фирмы, предлагающий покупателям товары по имеющимся у него образцам, каталогам и т.п.

Условия задачи

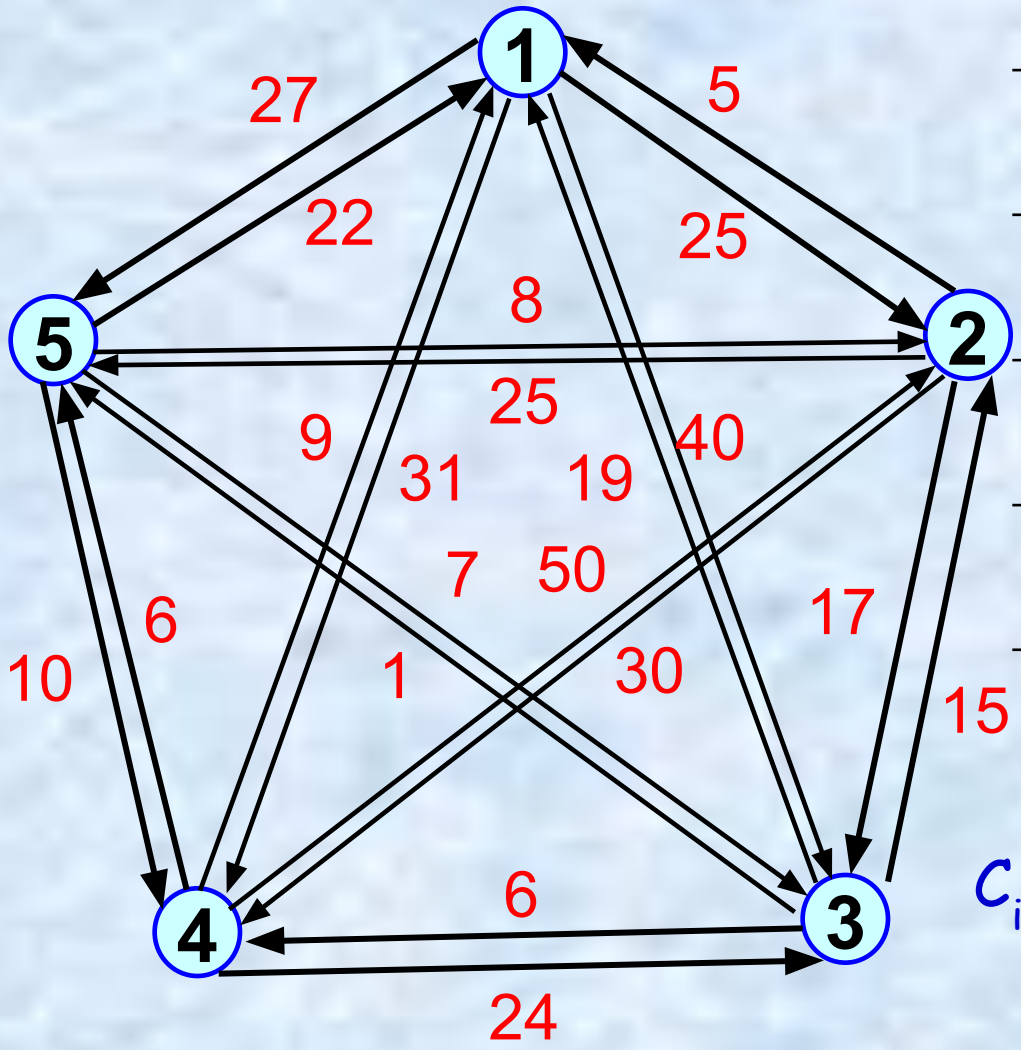
Множество городов = множество вершин графа.

Количество городов – *n*.

Ребра (дуги) графа соединяют вершины, между которыми разрешены поездки.

Стоимость поездки из одного города в другой – вес ребра графа.

Пример (n = 5)



Матрица стоимостей

	1	2	3	4	5
1	∞	25	40	31	27
2	5	∞	17	30	25
3	19	15	∞	6	1
4	9	50	24	∞	6
5	22	8	7	10	∞

c_{ij} – убыток (расходы) при переезде из i в j

Дано: 1. n – количество городов

2. $C = \{C_{ij}\}_{i,j=1...n}$ – матрица стоимостей

Найти: *маршрут объезда* всех городов (каждого по одному разу) с возвратом в исходный пункт, при этом стоимость поездки должна быть минимальной.

$$\pi = Permutation(1..n) = Перестановка(1..n)$$

$$\pi = (\pi(1), \pi(2), ..., \pi(n)),$$

$$\pi(i) \neq \pi(j), \quad \forall i, j \in 1..n : i \neq j, \pi(i) \in \{1..n\}$$

$$J(\pi) = \sum_{i=1}^{n-1} C_{\pi(i),\pi(i+1)} + C_{\pi(n),\pi(1)}$$

$$J^* = \min_{\pi \in P_n} J(\pi); \quad \pi = \arg\min_{\pi' \in P_n} \{J(\pi')\}$$

Метод ветвей и границ в ЗК

Основные идеи:

1. *Ветвление* – ***бинарное***: на каждом шаге маршруты разбиваются на две группы – ***включающие*** дугу (i, j) и ***запрещающие*** дугу (i, j)

2. Операция *«приведения матрицы»*

3. *Эвристика* в выборе *дуги* маршрута для ветвления в дереве вариантов

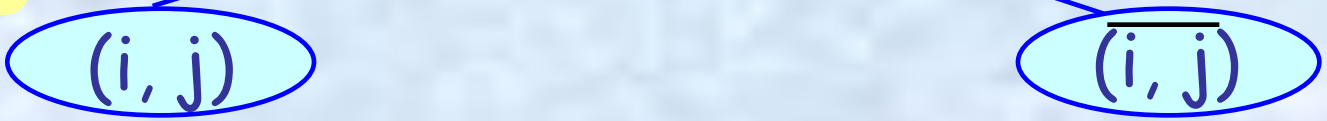
Ветвление

$Y_k=\{(i_1, j_1), \ldots,(i_{nk}, j_{nk})\}$

Включенная дуга

$X_k = (Y_k, Z_k)$

Исключенная дуга



$Y_{Left(k)} = Y_k \cup \{(i, j)\}$,
$Z_{Left(k)} = Z_k \cup \{(j, i)\}$,

Действия:

1) *Вычеркивание* i-й строки и j-го столбца (размер матрицы уменьшается на 1)
2) *добавление* в текущее решение (i, j)
3) *запрет* (j, i)

$Y_{Right(k)} = Y_k$,
$Z_{Right(k)} = Z_k \cup \{(i, j)\}$

Y_k – множество *включенных* в маршрут дуг;
Z_k – множество *исключенных* дуг

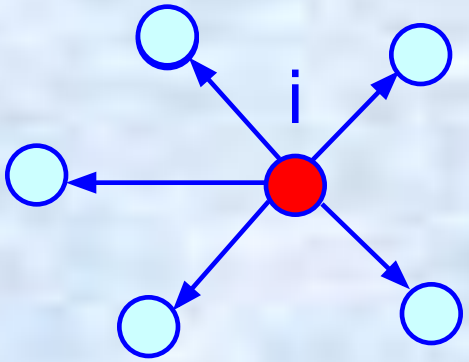
Операция «*приведения матрицы*»

1) По строкам:

для $\forall i \in 1..n$:

$g_i = \min \{C_{ij}: j \in 1..n\}$

g_i – минимальная сумма, достаточная для *выезда* *куда-либо* из города i

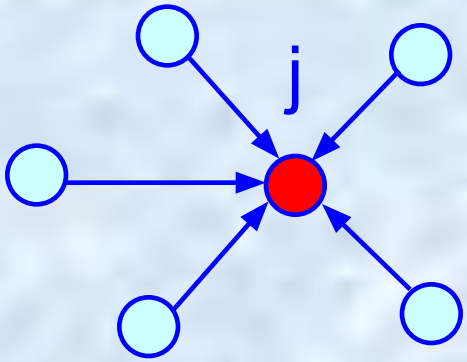


2) По столбцам:

для $\forall j \in 1..n$:

$h_j = \min \{(C_{ij} - g_i): i \in 1..n\}$

h_i – минимальная сумма, достаточная для *въезда* *откуда-либо* в город j



Приведенная матрица

$$C^*_{ij} = C_{ij} - g_i - h_j \geq 0$$

$$\Sigma_{i \in 1..n} g_i + \Sigma_{j \in 1..n} h_j = d$$

d – *нижняя граница стоимости любого решения*, не зависит от π, т.к. каждый город посещается ровно один раз.

Т.о.

$$J(\pi) = \sum_{i=1}^{n-1} C^*_{\pi(i),\pi(i+1)} + C^*_{\pi(n),\pi(1)} + d \geq d$$

Стоимость по приведенной матрице изменилась, но оптимальное решение останется тем же (стоимость всех допустимых маршрутов уменьшена на одну и ту же величину)

Пример
Вычитание по строкам

	1	2	3	4	5	
1	∞	25	40	31	27	- 25
2	5	∞	17	30	25	- 5
3	19	15	∞	6	1	- 1
4	9	50	24	∞	6	- 6
5	22	8	7	10	∞	- 7
						- 44

Вычитание по столбцам

	1	2	3	4	5
1	∞	0	15	6	2
2	0	∞	12	25	20
3	18	14	∞	5	0
4	3	44	18	∞	0
5	15	1	0	3	∞

- 3

d = 44 + 3 = 47

Нижняя граница стоимости любого решения

Результат операции *приведения матрицы*

	1	2	3	4	5
1	∞	0	15	3	2
2	0	∞	12	22	20
3	18	14	∞	2	0
4	3	44	18	∞	0
5	15	1	0	0	∞

В каждой строке и в каждом столбце имеется хотя бы один 0

Нижняя граница стоимости любого решения d = 47

Включение дуги i-j (левая ветвь дерева)
Например, 3-5

Действия над матрицей:

1) *Вычеркивание* i-й строки и j-го столбца (размер матрицы уменьшается на 1)
2) *запрет* (j, i): $c_{ji} := +\infty$

3) Затем новая *операция приведения*

	1	2	3	4	5
1	∞	0	15	3	2
2	0	∞	12	22	20
3	18	14	∞	2	0
4	3	44	18	∞	0
5	15	1	0	0	∞

+∞

Вычеркивание 3-й строки и 5-го столбца (размер матрицы уменьшается : 4×4 вместо 5×5)

	1	2	3	4	5
1	∞	0	15	3	2
2	0	∞	12	22	20
3	18	14	∞	2	0
4	3	44	18	∞	0
5	15	1	0	0	∞

	1	2	3	4
1	∞	0	15_{3}	3
2	0	∞	12_{0}	22
4	3_{0}	44_{41}	18_{3}	∞
5	15	1	∞	0

- 3

- 12

- 15

Новый шаг операции приведения

Исключение дуги i-j (правая ветвь дерева)
Например, $\overline{3\text{-}5}$

Действия над матрицей:

1) *запрет* (i, j): $C_{ij} := +\infty$
2) Затем новая *операция приведения*

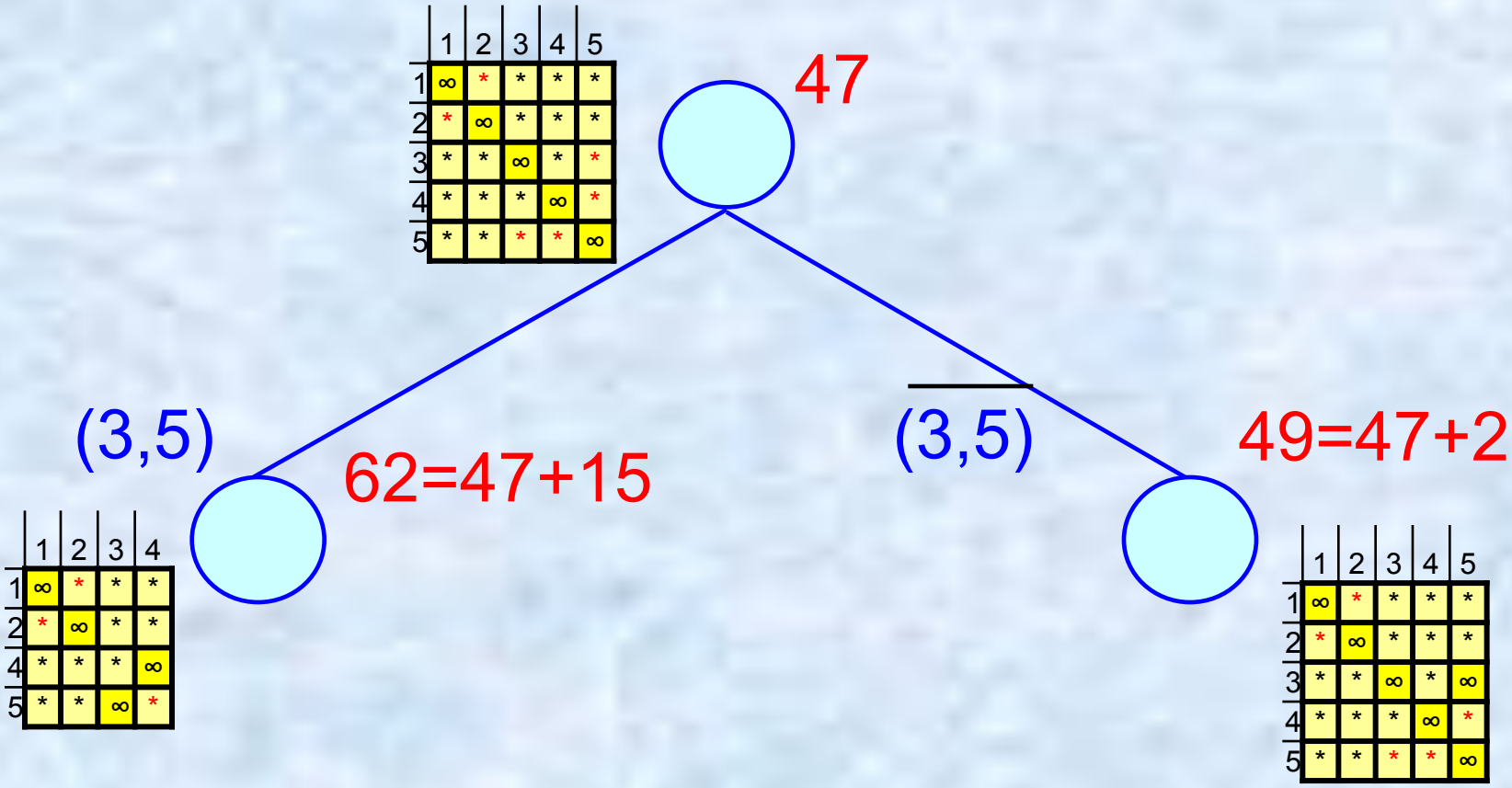
	1	2	3	4	5	
1	∞	0	15	3	2	
2	0	∞	12	22	20	
3	18	14	∞	2	∞	- 2
4	3	44	18	∞	0	
5	15	1	0	0	∞	

Новая нижняя граница стоимости любого решения

$d = 47 + 2 = 49$

$\Delta d = 2$

После ветвления по дуге (3,5)

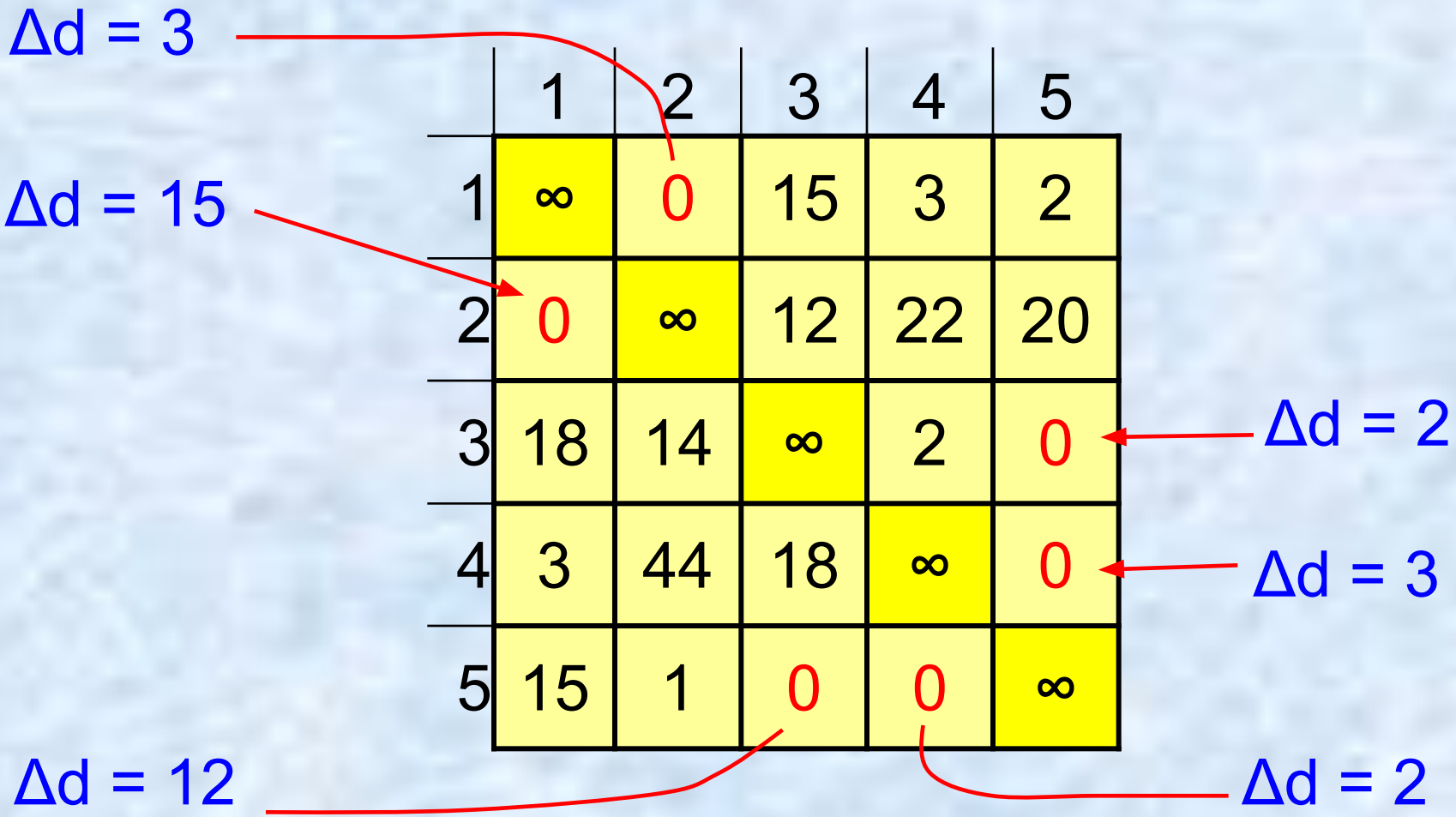


Какое ветвление сделать на первом шаге ?

Дуга (3,5) была рассмотрена в качестве примера возможного выбора.

Каковы «хорошие» варианты для выбора?

Кандидаты на ветвление (включение-исключение)



	1	2	3	4	5
1	∞	0	15	3	2
2	0	∞	12	22	20
3	18	14	∞	2	0
4	3	44	18	∞	0
5	15	1	0	0	∞

Эвристика*: выбор дуги для ветвления

При переходе в *правую* ветвь *новая нижняя граница стоимости* любого решения в этой ветви есть $d + \Delta d$.

В нашем примере для нулевых элементов матрицы рассматриваются варианты:

(i, j)	(1,2)	(2,1)	(3,5)	(4,5)	(5,3)	(5,4)
Δd	3	15	2	3	12	2

Максимум

Если $C_{ij} \neq 0$, то $\Delta d = 0$ (!).

Эвристика: выбор дуги для ветвления

Выбрать в приведенной матрице такой ***нулевой элемент***, который после замены на ∞ и последующего приведения матрицы даст максимальную *нижнюю границу стоимости* любого решения (т.е. максимальное Δd и, следовательно, d)

Метафора: самый **тяжелый** ноль!

Эвристика:

Рассмотрим множество пар (v,λ), таких, что $C^*_{v,\lambda} = 0$:

$$I = \{(v,\lambda): C^*_{v,\lambda} = 0,\ v \in I_1, \lambda \in I_2\},$$

где I_1, I_2 – множество городов для выбора по строкам и по столбцам, соответственно.

Пусть для $(v,\lambda) \in I$ имеем

$$\Delta d(v,\lambda) = \min\{C^*_{v,\rho}: \rho \neq \lambda\} + \min\{C^*_{\sigma,\lambda}: \sigma \neq v\}.$$

Выбираем $(i, j) = \mathbf{arg\ max}\{\Delta d(v,\lambda): (v,\lambda) \in I\}$.

При этом $\Delta d = \Delta d(i,j)$

Примечание: более точно везде использовать num(i), num(j) – номера городов, а i, j – индексы матрицы

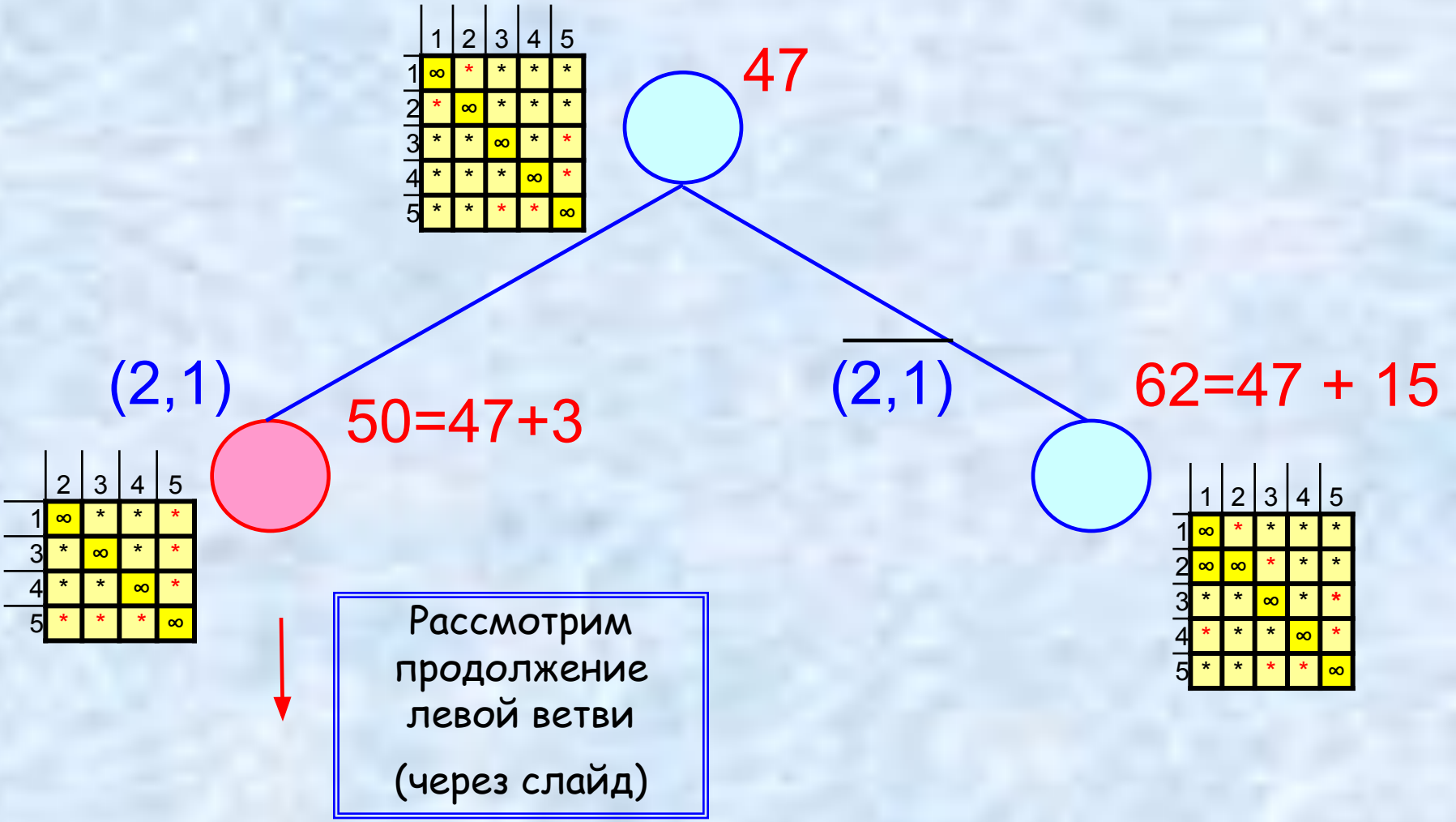
Итак, в примере, следуя указанной эвристике, выбирается ребро **(2, 1)**

Левая ветвь (включая (2, 1))

	1	2	3	4	5
1	∞	0	15	3	2
2	0	∞	12	22	20
3	18	14	∞	2	0
4	3	44	18	∞	0
5	15	1	0	0	∞

	2	3	4	5	
1	∞	15	3	2	- 2
3	14	∞	2	0	
4	44	18	∞	0	
5	1	0	0	∞	
	- 1				- 3

После ветвления по дуге (2,1)

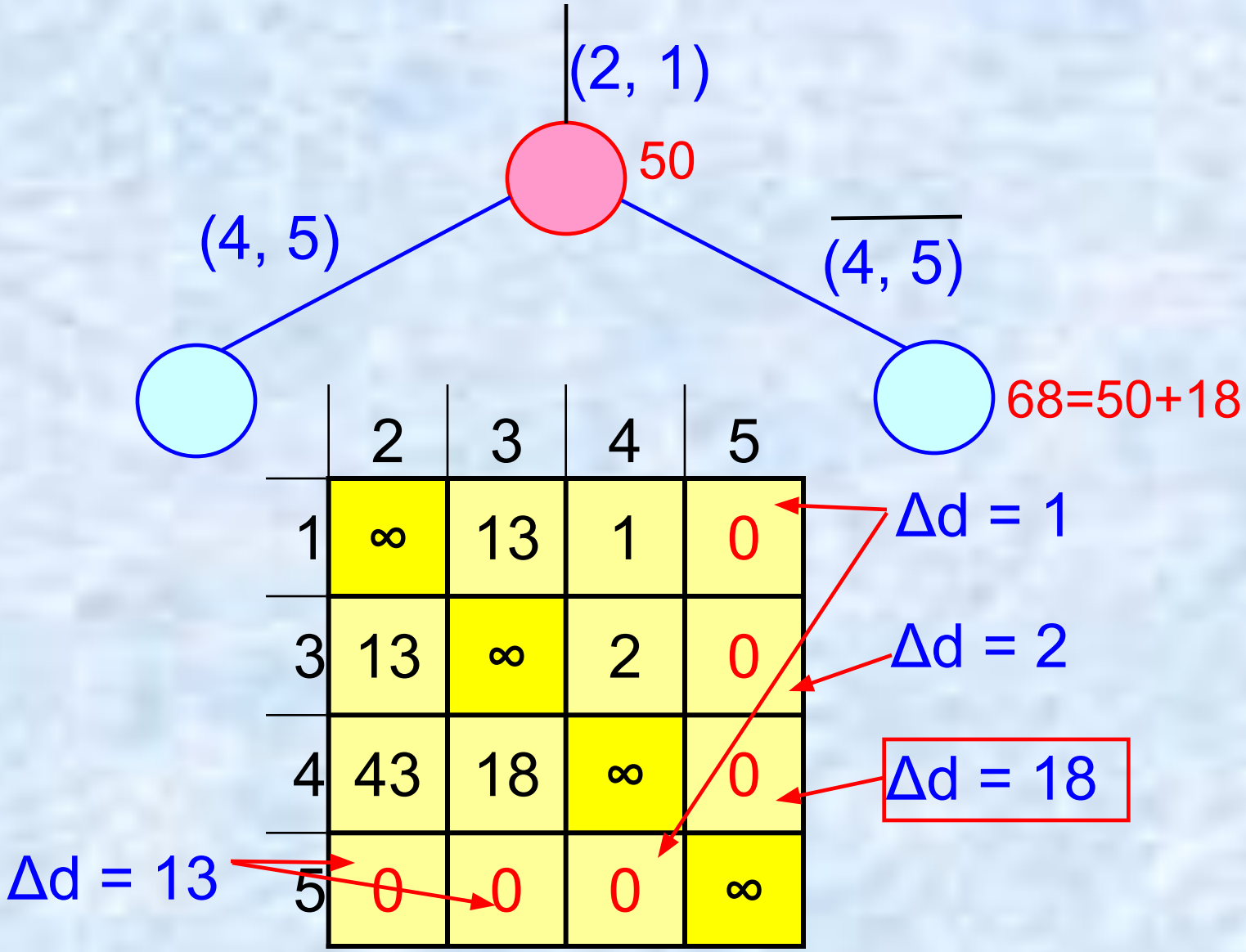


Правая ветвь (исключая (2, 1))

	1	2	3	4	5	
1	∞	0	15	3	2	
2	∞	∞	12	22	20	- 12
3	18	14	∞	2	0	
4	3	44	18	∞	0	
5	15	1	0	0	∞	
	- 3				- 15	

	1	2	3	4	5
1	∞	0	15	3	2
2	∞	∞	0	10	8
3	15	14	∞	2	0
4	0	44	18	∞	0
5	12	1	0	0	∞

Поддерево от ветки (2, 1)



Левая ветвь (включая (4, 5))

	2	3	4	5
1	∞	13	1	0
3	13	∞	2	0
4	43	18	∞	0
5	0	0	0	∞

	2	3	4	
1	∞	13	1	- 1
3	13	∞	2	- 2
5	0	0	∞	

	2	3	4
1	∞	12	0
3	11	∞	0
5	0	0	∞

53=50+3

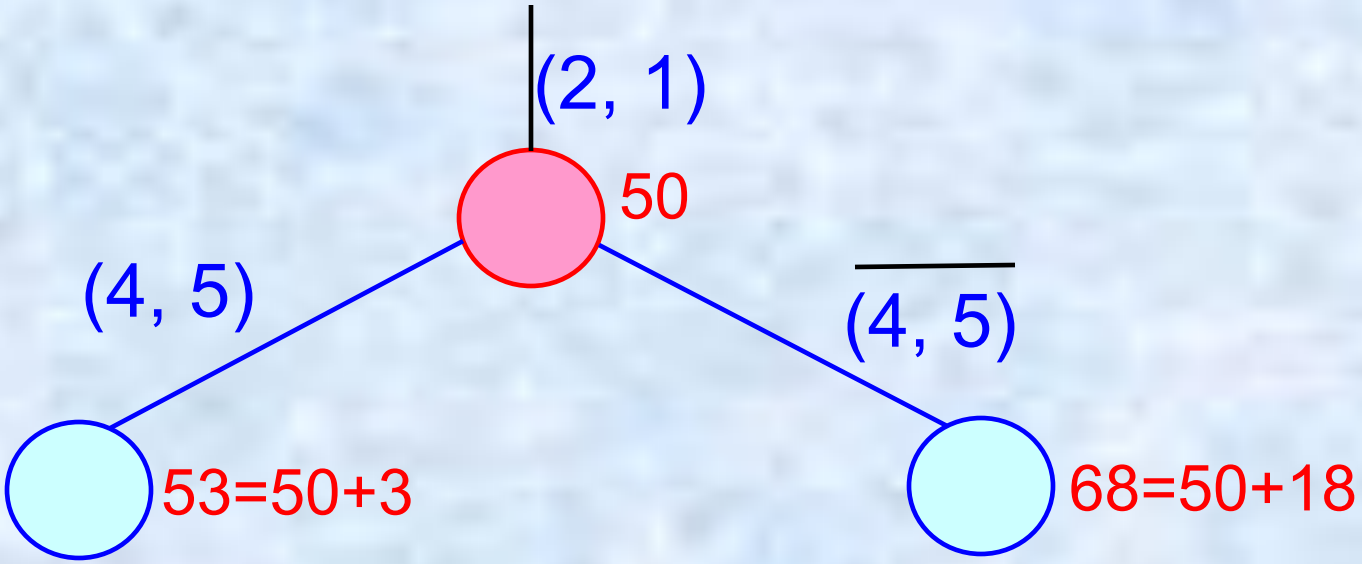
Правая ветвь (исключая (4, 5))

	2	3	4	5
1	∞	13	1	0
3	13	∞	2	0
4	43	18	∞	∞
5	0	0	0	∞

- 18

	2	3	4	5
1	∞	13	1	0
3	13	∞	2	0
4	25	0	∞	∞
5	0	0	0	∞

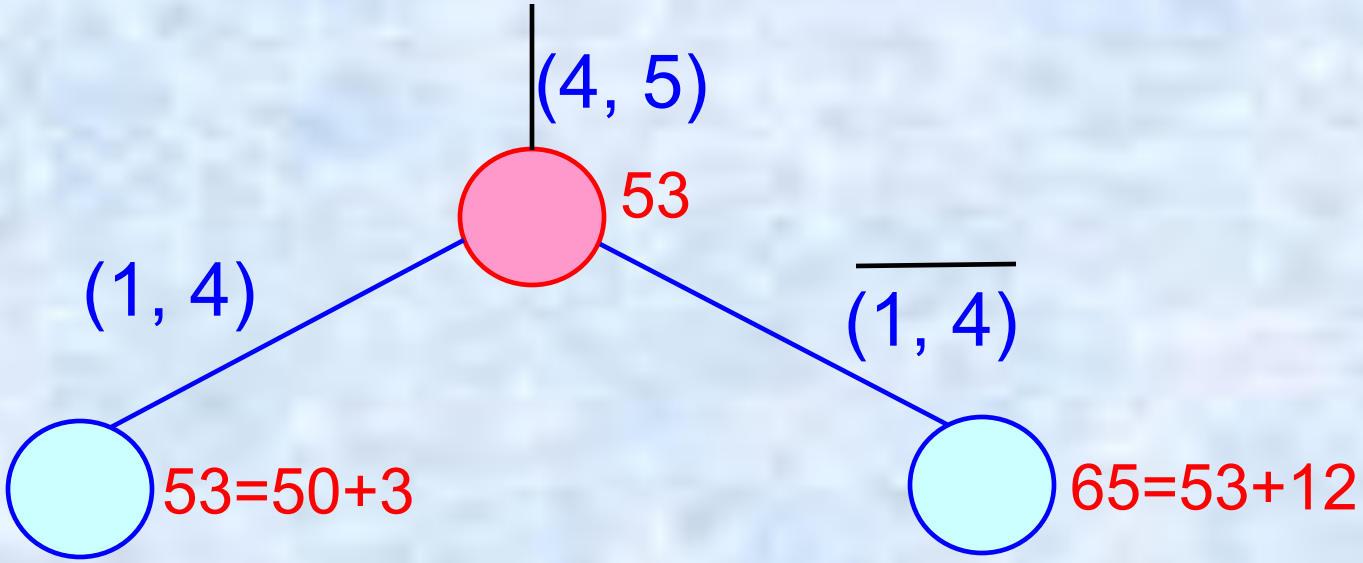
68=50+18



	2	3	4
1	∞	12	0
3	11	∞	0
5	0	0	∞

Кандидаты на ветвление узла (4, 5):
(1, 4) : Δd = 12
(5, 3) : Δd = 12

Поддерево от ветки (4, 5)



	2	3	4
1	∞	12	0
3	11	∞	0
5	0	0	∞

+ запрет (4, 1)

???

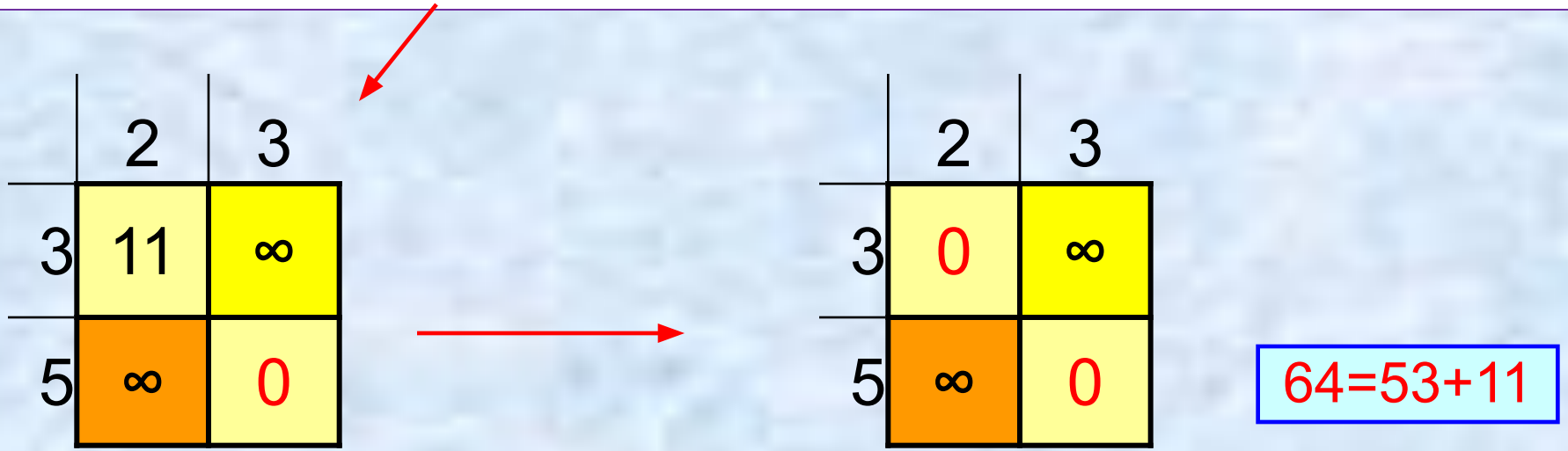
	2	3	4
1	∞	0	∞
3	11	∞	0
5	0	0	∞

Запрет (4, 1) ???

Частичное решение (2,1), (4,5), (1,4) или

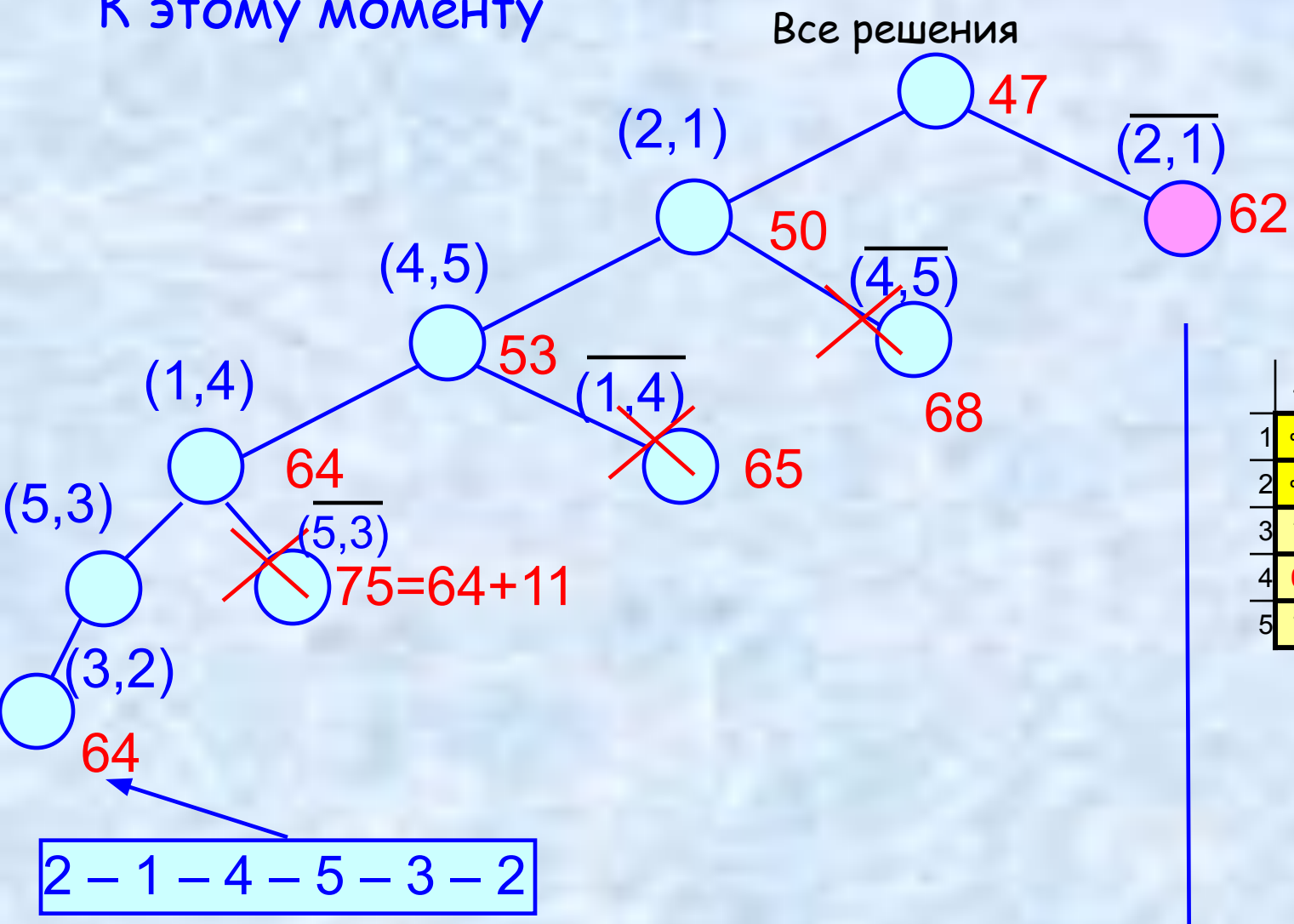
2 – 1 – 4 – 5

Запретить нужно (5, 2) !



Далее остаются (3, 2) и (5, 3),
т.о. 2 – 1 – 4 – 5, 5 – 3, 3 – 2,
т.е. решение есть маршрут (цикл) 2 – 1 – 4 – 5 – 3 – 2
Стоимость = 64. Легко проверить по матрице:
5 + 31 + 6 + 7 + 15 = 64

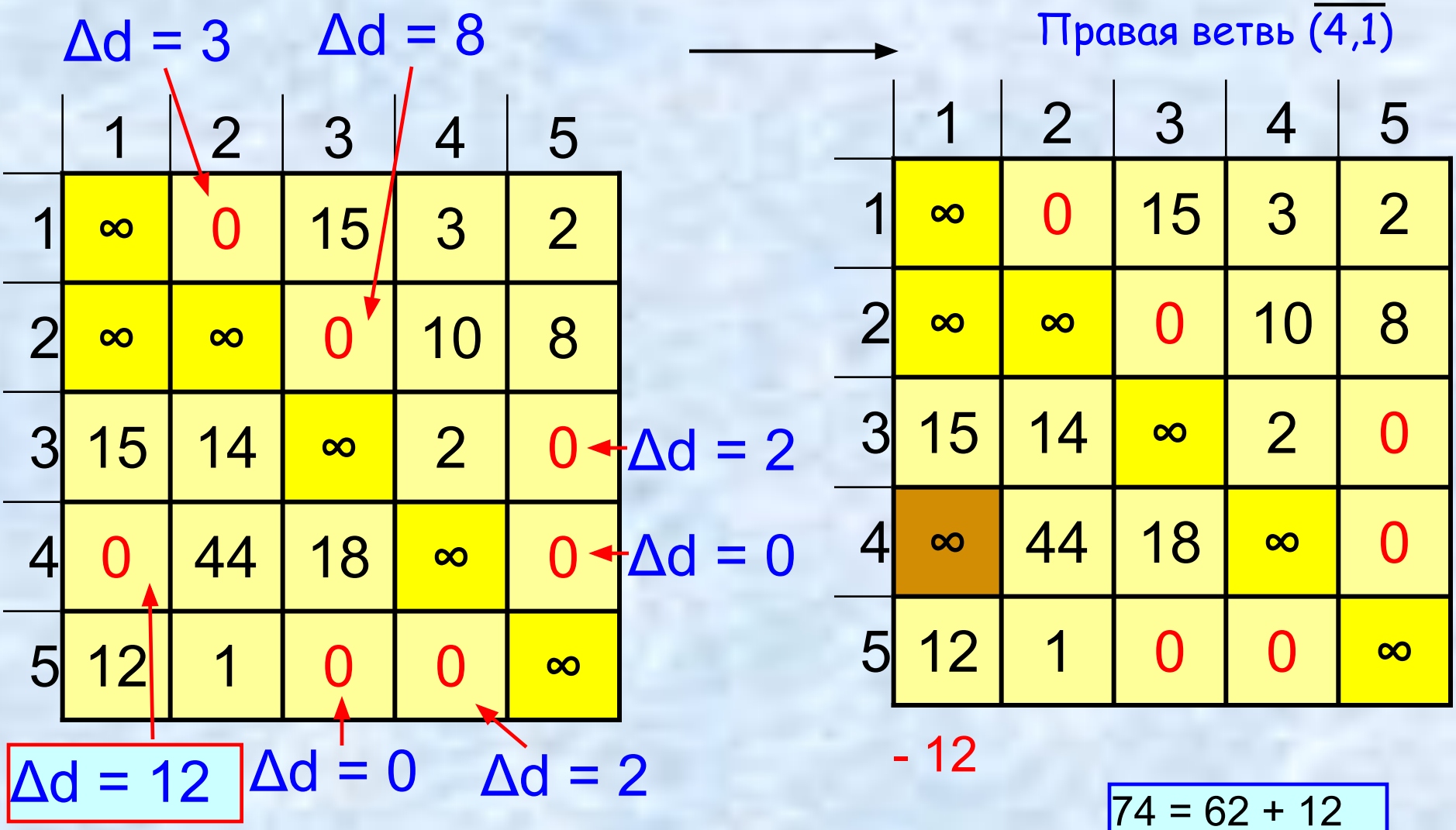
К этому моменту



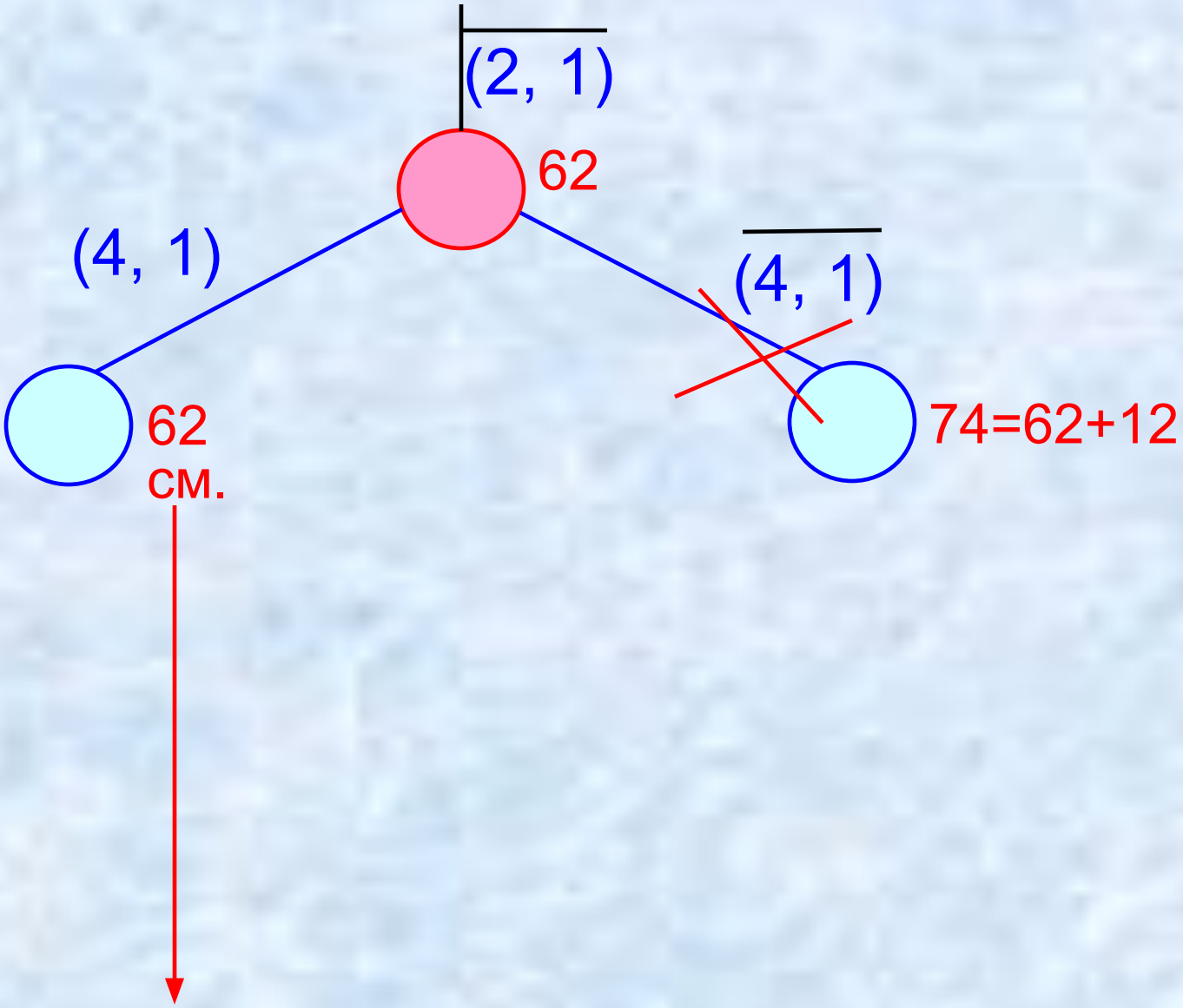
	1	2	3	4	5
1	∞	0	*	*	*
2	∞	∞	0	*	*
3	*	*	∞	*	0
4	0	*	*	∞	0
5	*	*	0	0	∞

2 – 1 – 4 – 5 – 3 – 2

Ветвление узла $\overline{(2,1)}$



Поддерево от ветки $\overline{(2, 1)}$



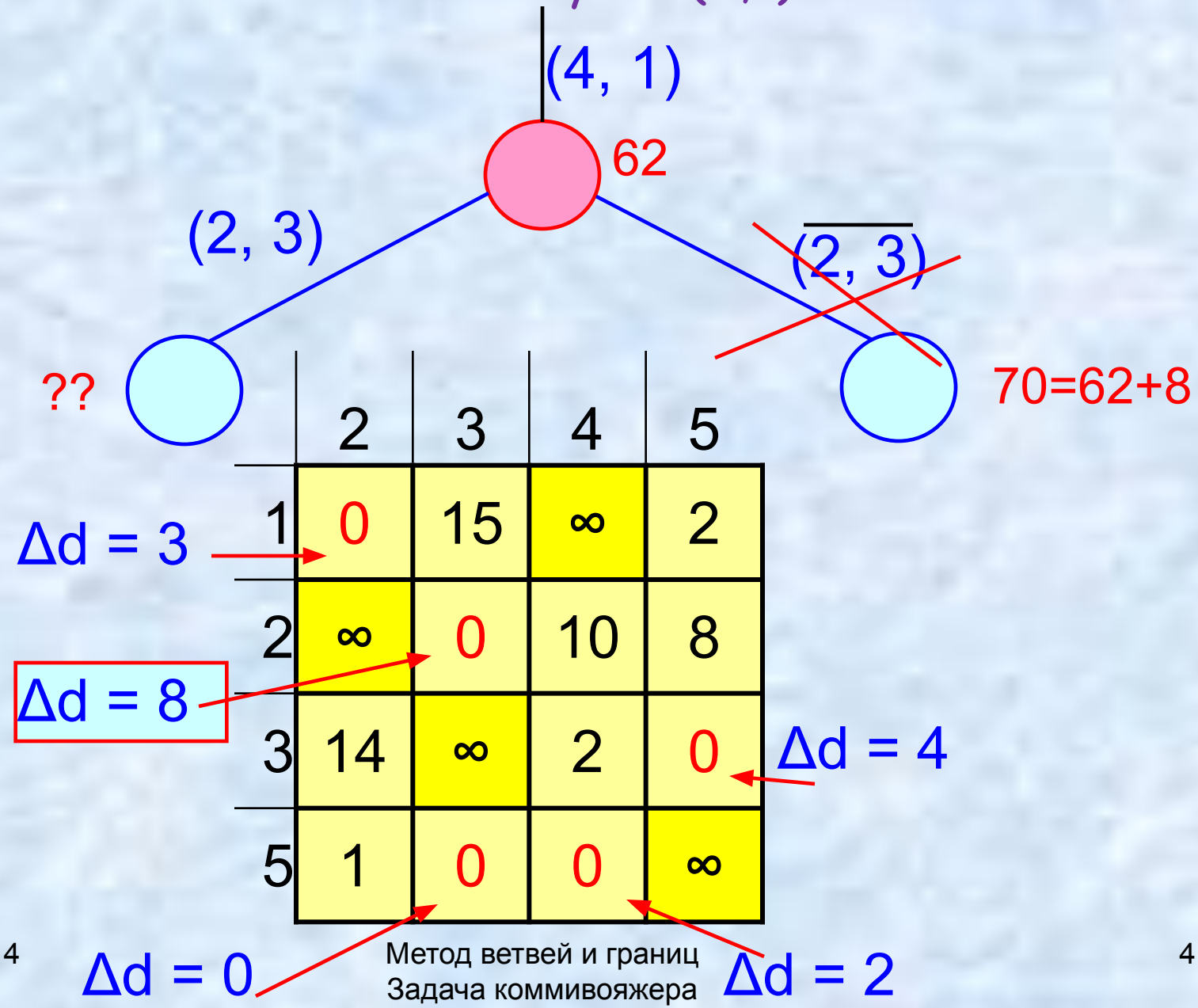
(4,1) – левая ветвь узла $\overline{(2,1)}$

	1	2	3	4	5
1	∞	0	15	3	2
2	∞	∞	0	10	8
3	15	14	∞	2	0
4	0	44	18	∞	0
5	12	1	0	0	∞

	2	3	4	5
1	0	15	∞	2
2	∞	0	10	8
3	14	∞	2	0
5	1	0	0	∞

$\Delta d = 0$

Ветвление узла (4,1)



(2, 3) – левая ветка узла (4,1)

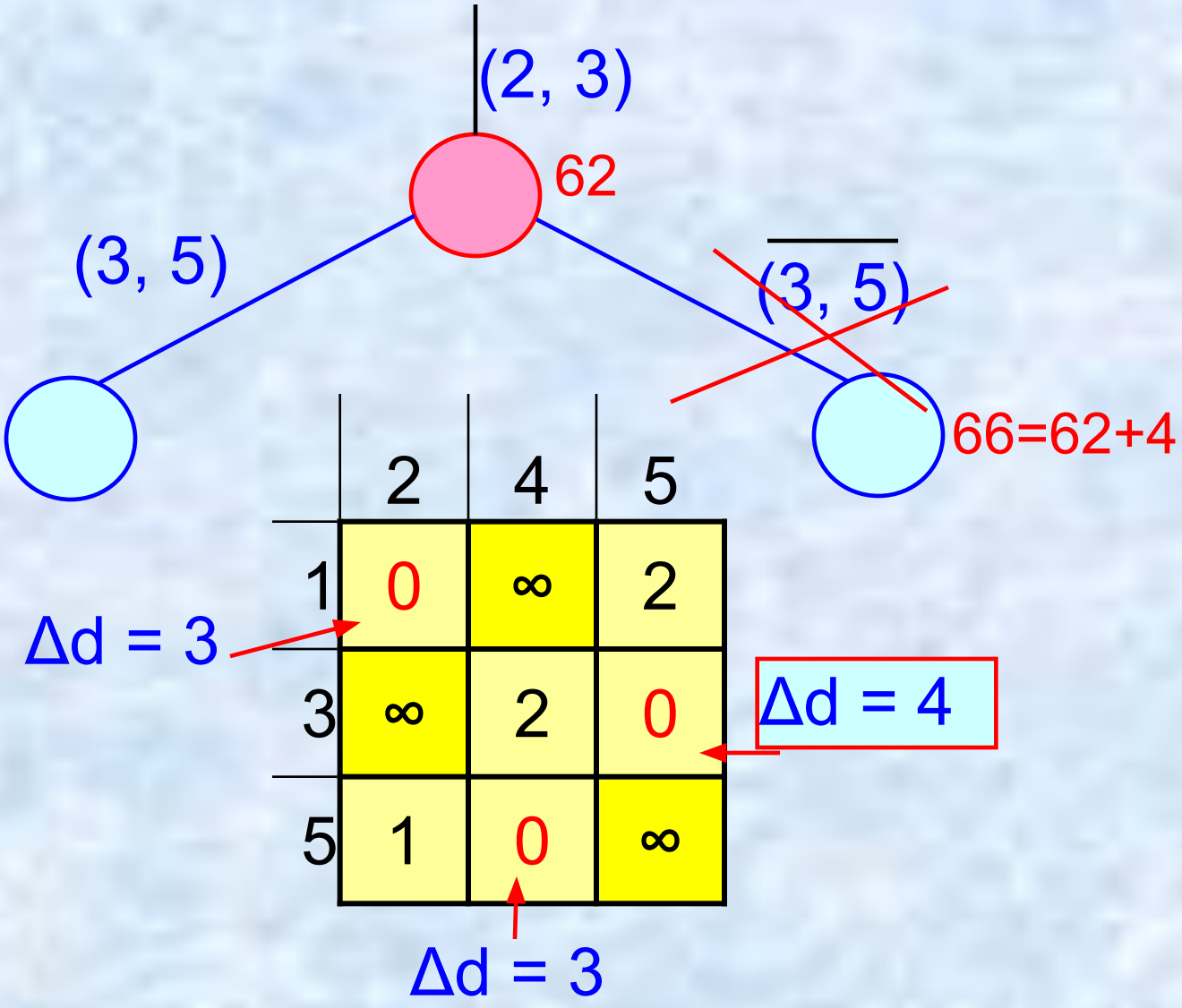
	2	3	4	5
1	0	15	∞	2
2	∞	0	10	8
3	14	∞	2	0
5	1	0	0	∞

	2	4	5
1	0	∞	2
3	∞	2	0
5	1	0	∞

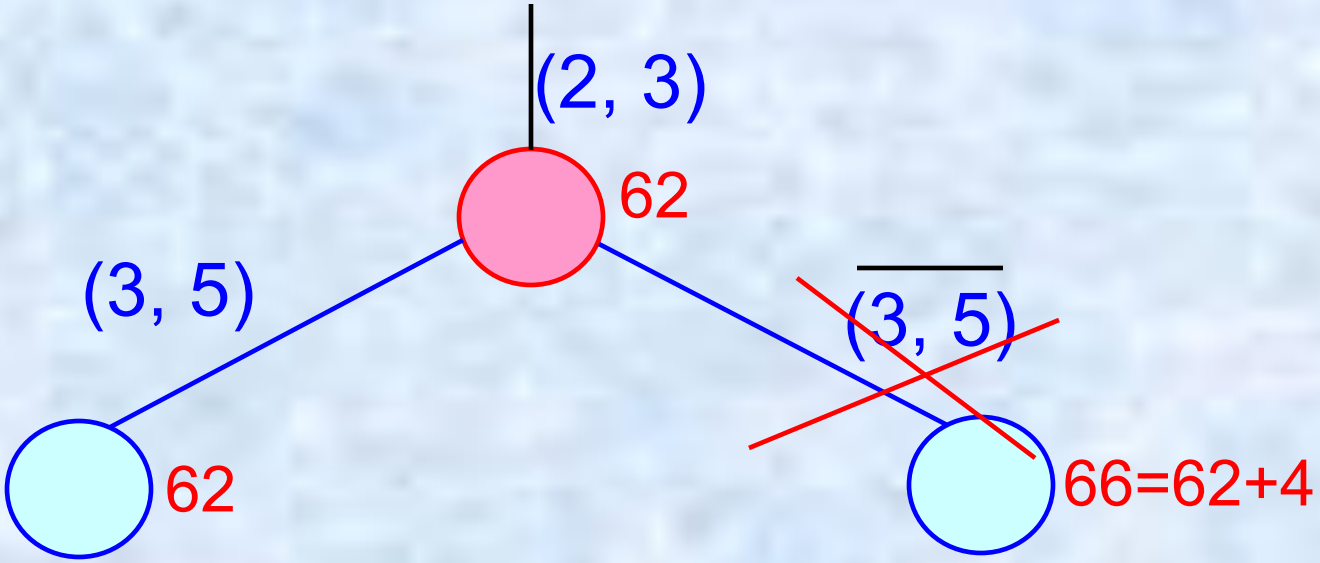
$\Delta d = 0$

$d = 62$

Ветвление узла (2,3)



Ветвление узла (2,3)



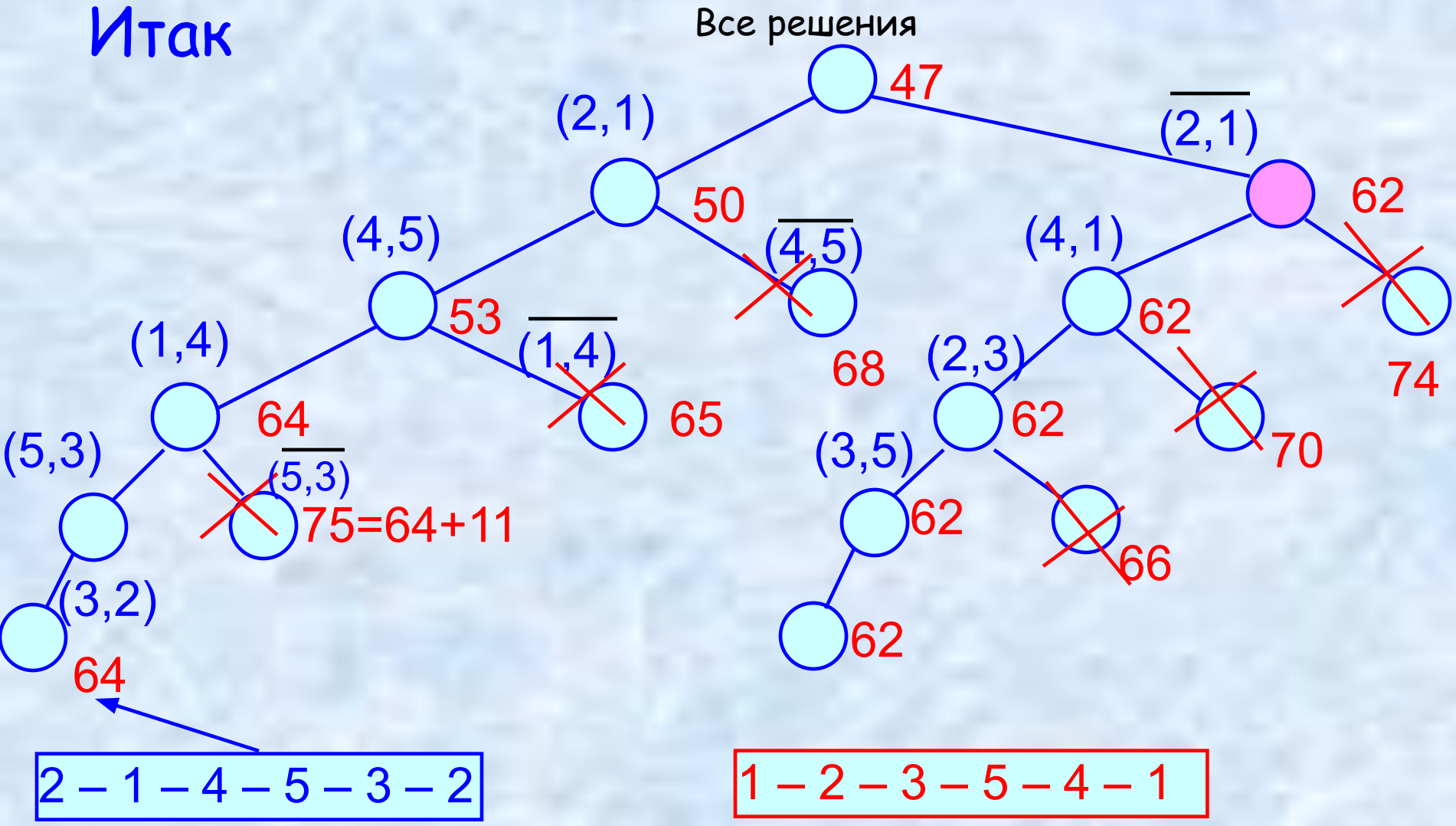
	2	4	5
1	0	∞	2
3	∞	2	0
5	1	0	∞

Решение + (1,2) + (5,4)

Т.о. (4,1), (2,3), (3,5), (1,2), (5,4)
или

1 - 2 - 3 - 5 - 4 - 1

Итак


Все решения
47
(2,1)
(2,1)
50
62
(4,5)
(4,5)
(4,1)
53
62
(1,4)
(1,4)
68
(2,3)
74
64
62
(5,3)
65
(3,5)
(5,3)
70
75=64+11
62
66
(3,2)
62
64
2 – 1 – 4 – 5 – 3 – 2
1 – 2 – 3 – 5 – 4 – 1

Сложность алгоритма

Сложность точного алгоритма ЗК (методом ВиГ) в среднем (при «случайных» матрицах стоимостей)

$\approx C^n$, где $C \cong 1.26$

Эмпирический результат (опытным путем)

Приближенные решения (не минимальной стоимости)

Зачем нужны приближенные решения?

1) «Быстрые» решения

2) Для оценки границ при поиске точного решения!

Приближенные решения (не минимальной стоимости)

1. Алгоритм ближайшего соседа (АБС)

Начиная с любого города, выбираем на каждом шаге следующий город, стоимость проезда в который из данного города минимальна

	1	2	3	4	5
1	∞	25	40	31	27
2	5	∞	17	30	25
3	19	15	∞	6	1
4	9	50	24	∞	6
5	22	8	7	10	∞

1) **1** – 2 – 3 – 5 – 4 – 1: стоимость = 25+17+1+10+9 = 62

совпадает со стоимостью оптимального решения (!).

Если элемент матрицы (4,1) заменить на 9+x, то стоимость решения АБС станет 62+x, где x любое число (!)

2) **2** – 1 – 5 – 3 – 4 – 2 : стоимость = 5+27+7+6+50= 95

3) **3** – 5 – 2 – 1 – 4 – 3 : стоимость = 1+8+5+31+24= 69

	1	2	3	4	5
1	∞	25	40	31	27
2	5	∞	17	30	25
3	19	15	∞	6	1
4	9	50	24	∞	6
5	22	8	7	10	∞

4) **4** – 5 – 3 – 2 – 1 – 4:
стоимость = 6+7+15+5+31 = 64

5) **5** – 3 – 4 – 1 – 2 – 5:
стоимость = 7+6+9+25+25 = 72

Итак АБС: 62, 95, 69, 64, 72

Ещё пример: n = 6

Оптимальное решение 1 - 4 - 3 - 5 - 6 - 2 - 1.

Стоимость = 16+25+5+5+5+7 = 63

	1	2	3	4	5	6
1	∞	27	43	**16**	30	26
2	7	∞	**16**	1	30	25
3	20	13	∞	35	5	**0**
4	21	**16**	25	∞	18	18
5	**12**	46	27	48	∞	5
6	23	5	5	9	**5**	∞

1) 1 – 4 – 2 – 3 – 6 – 5 – 1 :

Стоимость = 16+16+16+0+5+12 = 65

2) 2 – 4 – 5 – 6 – 3 – 1 – 2 :

Стоимость = 1+18+5+5+20+27 = 76

3) A: 3 – 6 – 2 – 4 – 5 – 1 – 3 :

Стоимость = 0+5+1+18+12+43 = 79

3) Б: 3 – 6 – 5 – 1 – 4 – 2 – 3 :
стоимость =
0+5+12+16+16+16 = 65 (см. 1)

	1	2	3	4	5	6
1	∞	27	43	**16**	30	26
2	7	∞	**16**	1	30	25
3	20	13	∞	35	5	**0**
4	21	**16**	25	∞	18	18
5	**12**	46	27	48	∞	5
6	23	5	5	9	**5**	∞

4) 4 – 2 – 1 – 6 – 3 – 5 – 4 :
Стоимость = 16+7+26+5+5+48 = 107

5) А: 5 – 6 – 3 – 2 – 4 – 1 – 5 :
Стоимость = 5+5+13+1+21+30 = 75

5) Б: 5 – 6 – 2 – 4 – 1 – 3 – 5 :
Стоимость = 5+5+1+21+43+5 = 80

6) А: 6 – 2 – 4 – 5 – 1 – 3 – 6 :
Стоимость = 5+1+18+12+43+0 = 79 (см. 3А)

6) Б: 6 – 3 – 5 – 1 – 4 – 2 – 6 :
Стоимость = 5+5+12+16+16+25 = 79 (см. 3А)

6) В: 6 – 5 – 1 – 4 – 2 – 3 – 6 :
Стоимость = 5+12+16+16+16+0 = 65 (см. 3Б)

Итак, 65, 75, 76, 79, 80, 107

Оценка степени приближения алгоритма ближайшего соседа (АБС)

N_n – маршрут АБС, $|N_n|$ – его длина (стоимость).

O_n – оптимальный маршрут,

$|O_n|$ – его длина (стоимость).

Утверждение. Если матрица $\{C_{ij}\}$ – (*а*) симметрична и (*б*) удовлетворяет неравенству треугольника

$$C_{ij} \le C_{ik} + C_{kj}\ ,\ \forall\, i,\ j,\ k,$$

то

$$\frac{|N_n|}{|O_n|} \le \frac{1}{2}\left(\lceil \log_2 n \rceil + 1\right).$$

2. Алгоритм включения ближайшего города (АВБГ)

Если есть цепочка $v_{i(1)} - v_{i(2)} - \ldots - v_{i(k-1)} - v_{i(k)}$,
то следующим выбирается город v_j,
ближайший к этой цепочке,
т.е. имеющий минимальную из стоимостей $c_{i(q),j}$
(для $q=1,\ldots,k$), и этот город
вставляется в текущий маршрут
вслед за городом $v_{i(q)}$.

$$v_{i(1)} - v_{i(2)} - \ldots - v_{i(q)} - v_j - v_{i(q+1)} - \ldots - v_{i(k-1)} - v_{i(k)},$$

Пример: n = 6

Оптимальное решение 1 – 4 – 3 – 5 – 6 – 2 – 1.

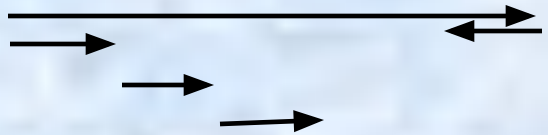
Стоимость = 16+25+5+5+5+7 = 63

	1	2	3	4	5	6
1	∞	27	43	16	30	26
2	7	∞	16	1	30	25
3	20	13	∞	35	5	0
4	21	16	25	∞	18	18
5	12	46	27	48	∞	5
6	23	5	5	9	5	∞

1) 1 – 4 – 2 – 3 – 6 – 5 – 1 :

Стоимость = 16+16+16+0+5+12 = 65 (совпадает с АБС !)

2) 2 – 3 – 6 – 5 – 1 – 4 – 2 :



Стоимость = 16+0+5+12+21+16 = 70

3) 3 – 6 – 2 – 1 – 4 – 5 – 3 :

0+5+7+16+18+27= 73

5) 5 – 6 – 3 – 2 – 1 – 4 – 5 :

5+5+13+7+16+18 = 64 !!

Оценка степени приближения АВБГ

I_n – маршрут АВБГ, $|I_n|$ – его длина (стоимость).
O_n – оптимальный маршрут,
$|O_n|$ – его длина (стоимость).

Утверждение. Если матрица $\{C_{ij}\}$ – (*а*) симметрична и (*б*) удовлетворяет неравенству треугольника

$$C_{ij} \le C_{ik} + C_{kj}, \ \forall i, j, k,$$

то

$$\frac{|I_n|}{|O_n|} \le 2$$

Сложность приближенных алгоритмов

АБС $\approx C_1 n^2$

АВБГ $\approx C_2 n^2$, если для каждого города не включенного в маршрут хранить данные о ближайшем к нему из уже включенных в маршрут. На каждом шаге эти данные корректировать за счет нового включенного.

Есть и другие приближенные решения (см. след. раздел – минимальное остовное дерево графа)

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ