Лекция 6.
Графы (продолжение).

План лекции

- Проблема поиска кратчайших путей
- Поиск кратчайших путей обходом в ширину
- Алгоритм Дейскстры
- Алгоритм Флойда
- Алгоритм Беллмана-Форда
- Поиск кратчайших путей на практике

Проблема поиска кратчайших путей

Путь в графе – это некоторая последовательность вершин, последовательно связанных рёбрами.

Кратчайший путь – это путь с наименьшей суммой весов рёбер (в невзвешенном графе считают вес ребра = 1)

(Проблема = Задача)

Проблема поиска кратчайших путей

Поиск кратчайших путей – очень распространённая задача. Люди сталкиваются с ней каждый день, прикидывая свои маршруты передвижения. Если взять все возможные пути передвижения и объединить их в один граф, то можно решать эту задачу алгоритмическими методами.

Решение обходом в ширину

Вспомним, в чём заключается обход в ширину:

Алгоритм разбирает вершины графа по очереди, для каждой вершины приписывая всех её непосещённых соседей в конец очереди.

Решение обходом в ширину

Обойдя всех соседей вершины 1 – 2 и 3, алгоритм примется за соседей вершин 2 и 3 – это 4, 6, 7, 8, а затем обработает оставшуюся вершину 5.

Решение обходом в ширину

В итоге, в обходе в ширину обход вершин графа идёт в порядке возрастания расстояния от начальной вершины.

Решение обходом в ширину

```
vector <int> a[100];
int was[100];

void bfs(int start) {
    was[start]=1;
    queue <int> q;
    q.push(start);
    while(!q.empty) {
        int v = q.front();
        for(int i=0;i<a[v].size();i++)
            if(!was[a[v][i]]) {
                was[a[v][i]] = was[v + 1];
                q.push(a[v][i]);
            }
        q.pop();
    }
}
```

Единственная строка, в которой нужны изменения!

Решение обходом в ширину

Вывод: обход в ширину – простой способ найти расстояние от вершины до всех остальных вершин невзвешенного графа с асимптотикой O(n).

Алгоритм Дейкстры
(Dijkstra’s algorythm)

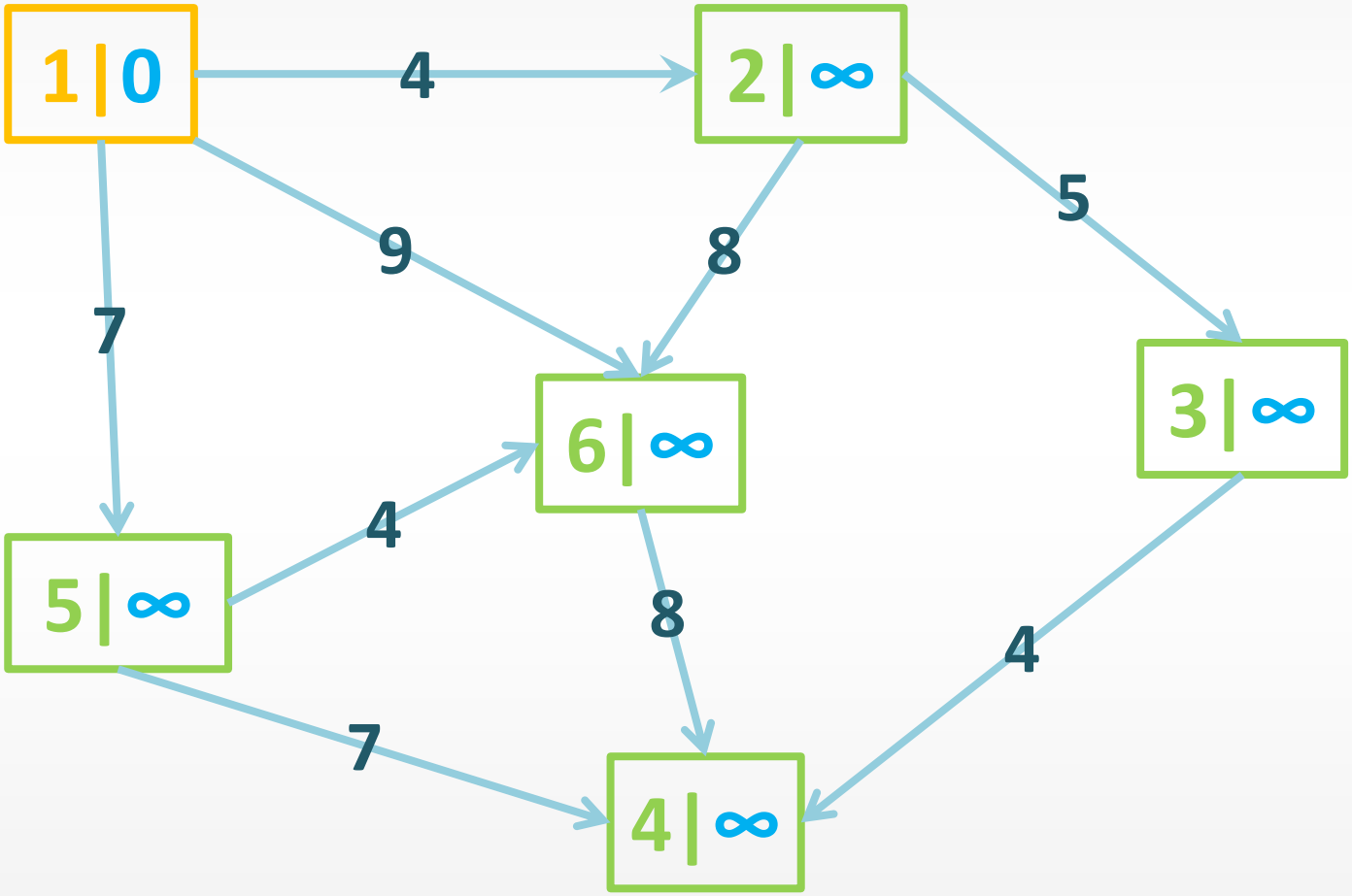
Для взвешенных графов существует решение сложнее, хотя и немного похожее в идее. Этот алгоритм был предложен нидерландским учёным Эдсгером Дейкстрой в 1959 году.

Алгоритм Дейкстры

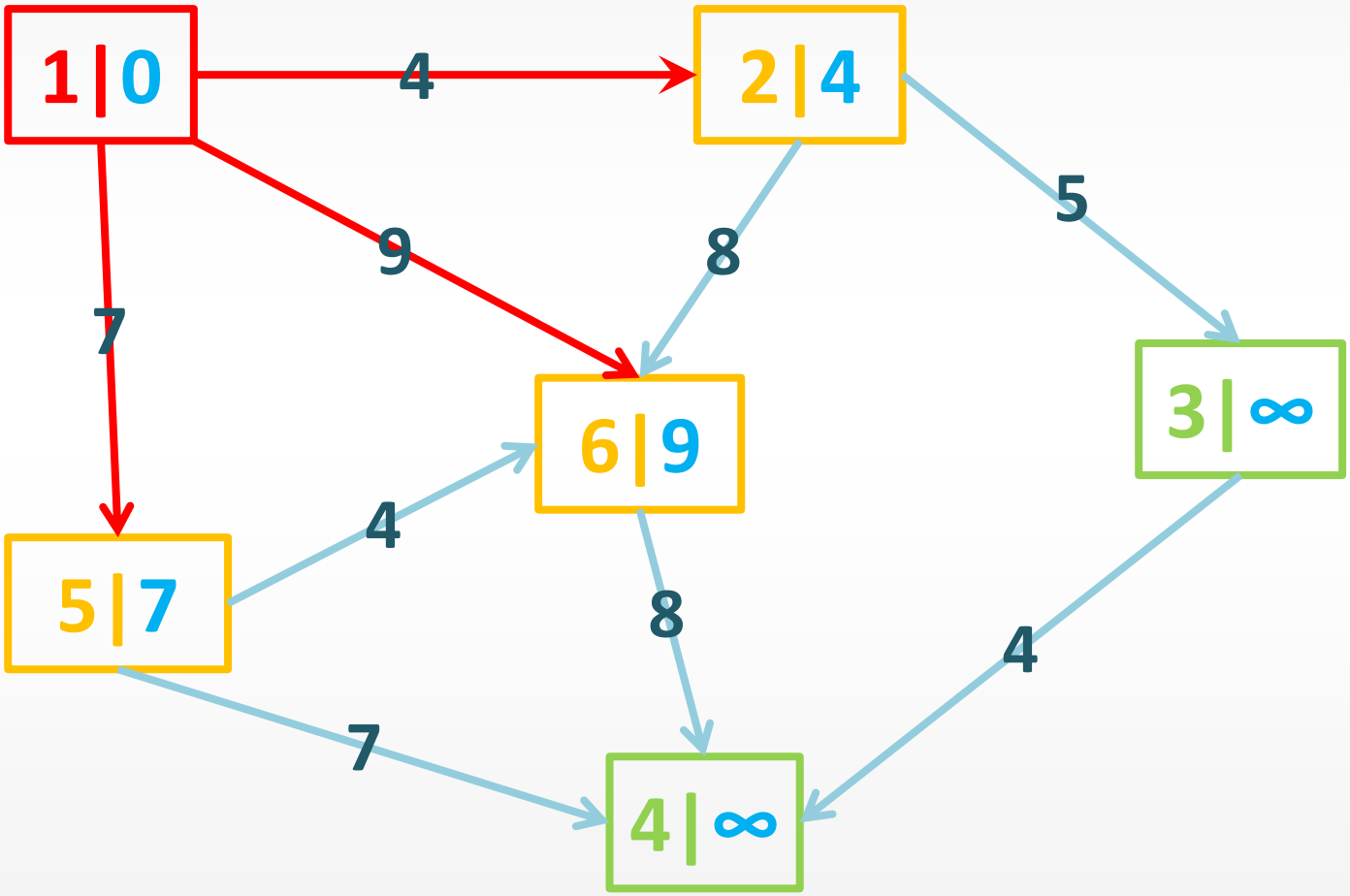
Суть алгоритма:

- Изначально расстояние для всех вершин, кроме первой считается бесконечно большим, а для первой – 0.
- Пусть обрабатываемая вершина – **a**, известное расстояние до неё – **d[a]**, а вес ребра между вершинами **a** и **b** – **w[a][b]**.
- Для каждого необработанного соседа **b** вершины **a** расстояние **d[b]** пересчитывается по формуле **d[b] = min (d[b], d[a] + w[a][b])**.
- Когда расстояния для всех необработанных соседей вершины **a** пересчитаны, вершина **a** считается обработанной.
- После этого алгоритм переходит к той вершине, известное расстояние до которой – наименьшее.

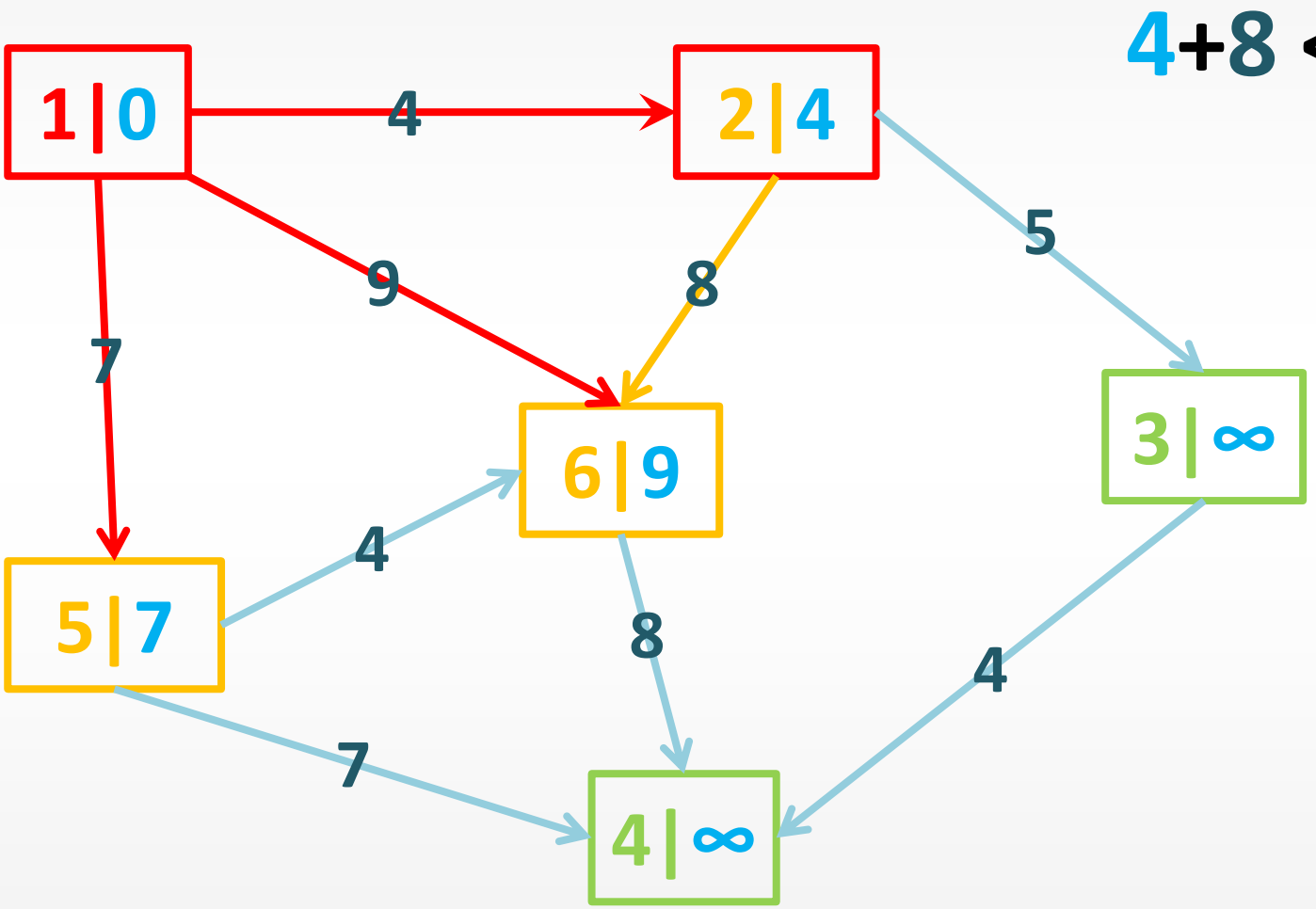
Алгоритм Дейкстры



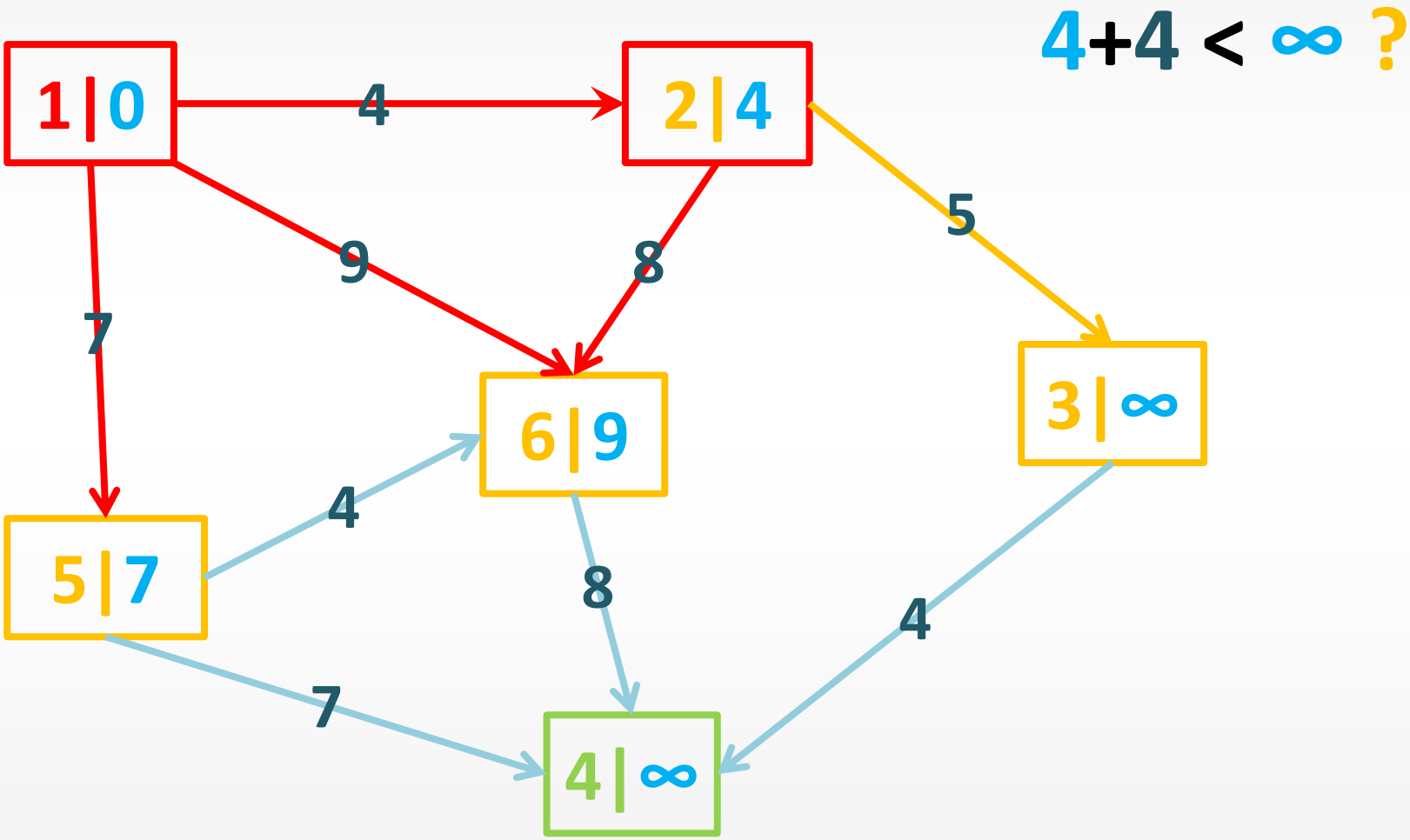
Алгоритм Дейкстры



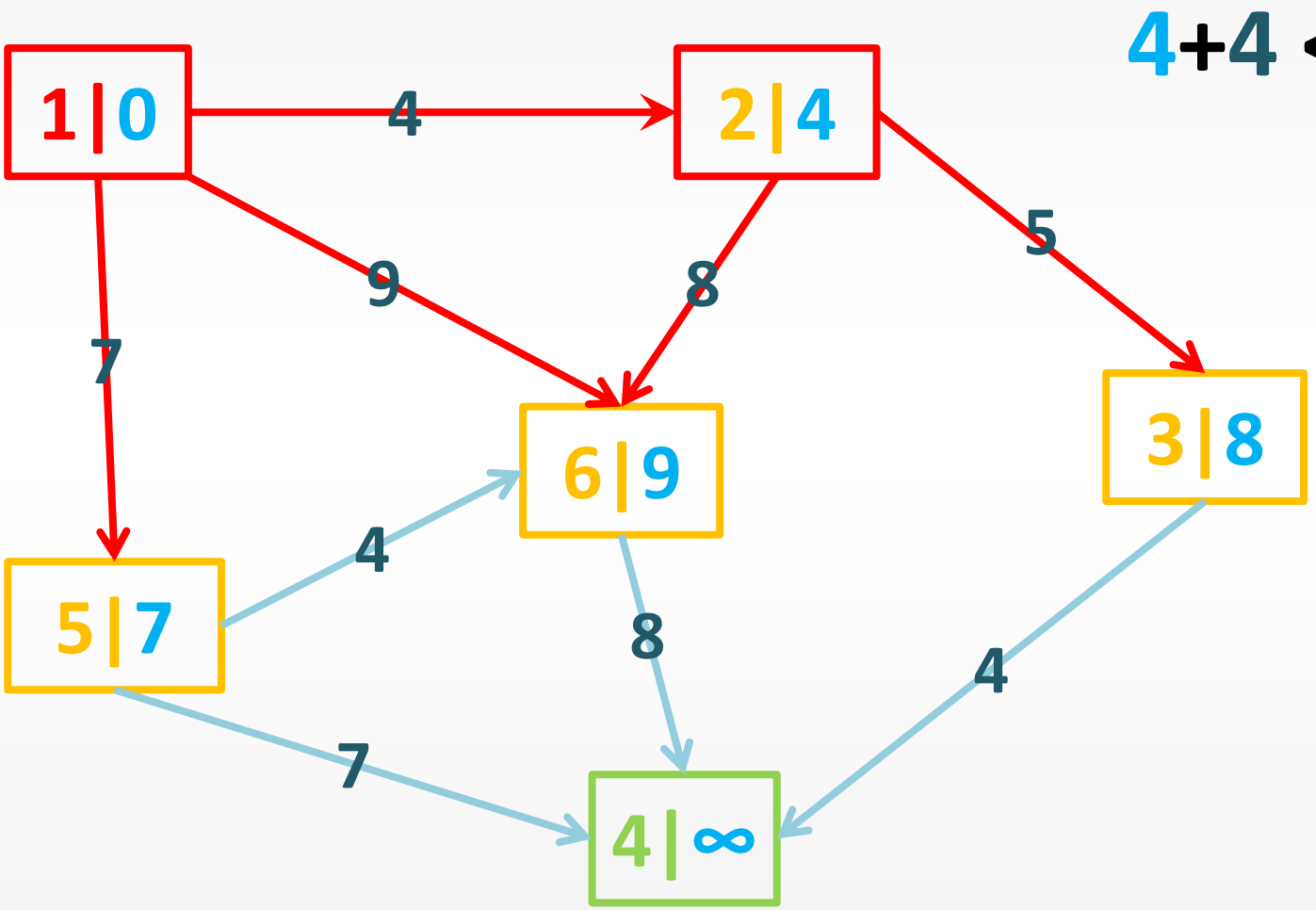
Алгоритм Дейкстры



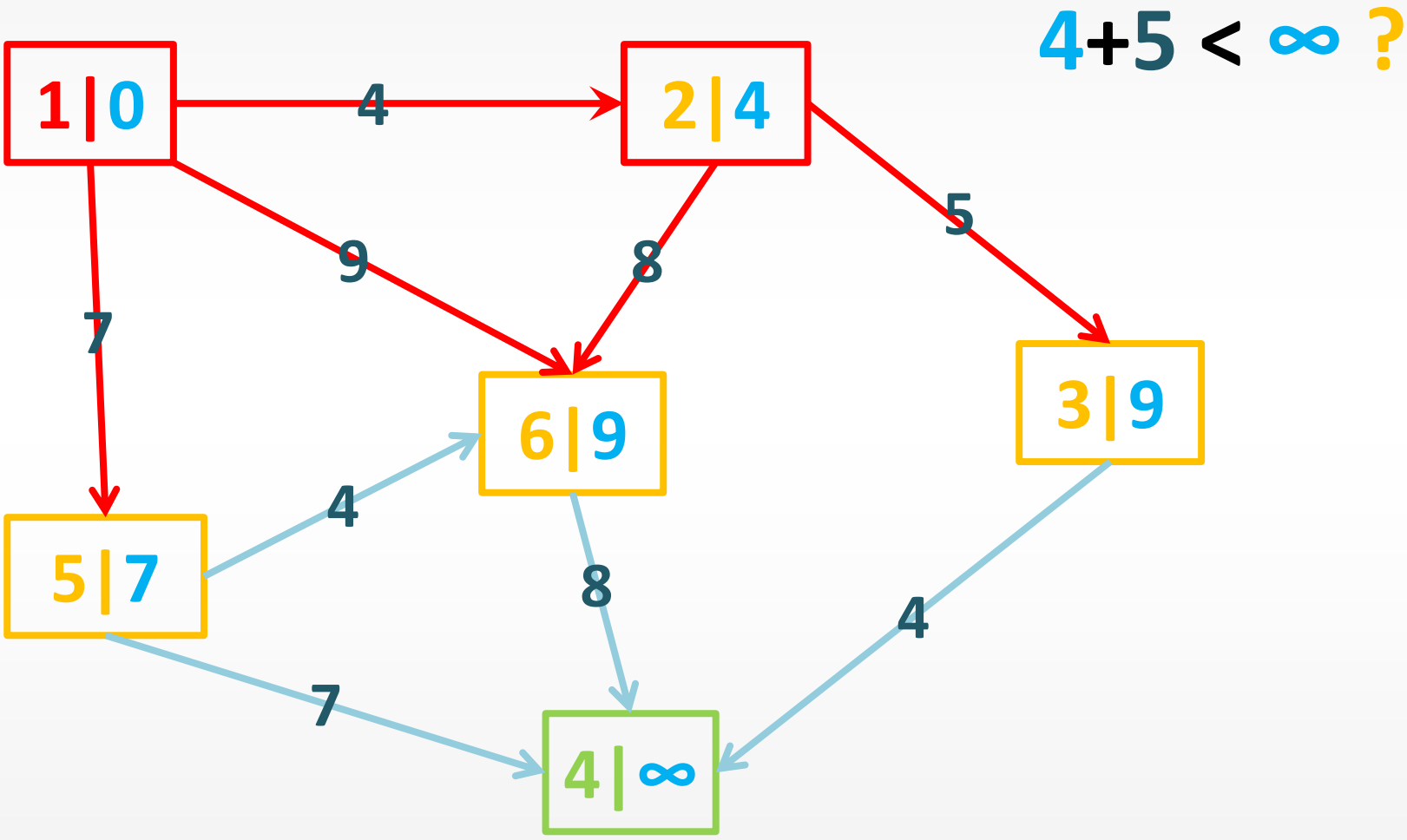
Алгоритм Дейкстры



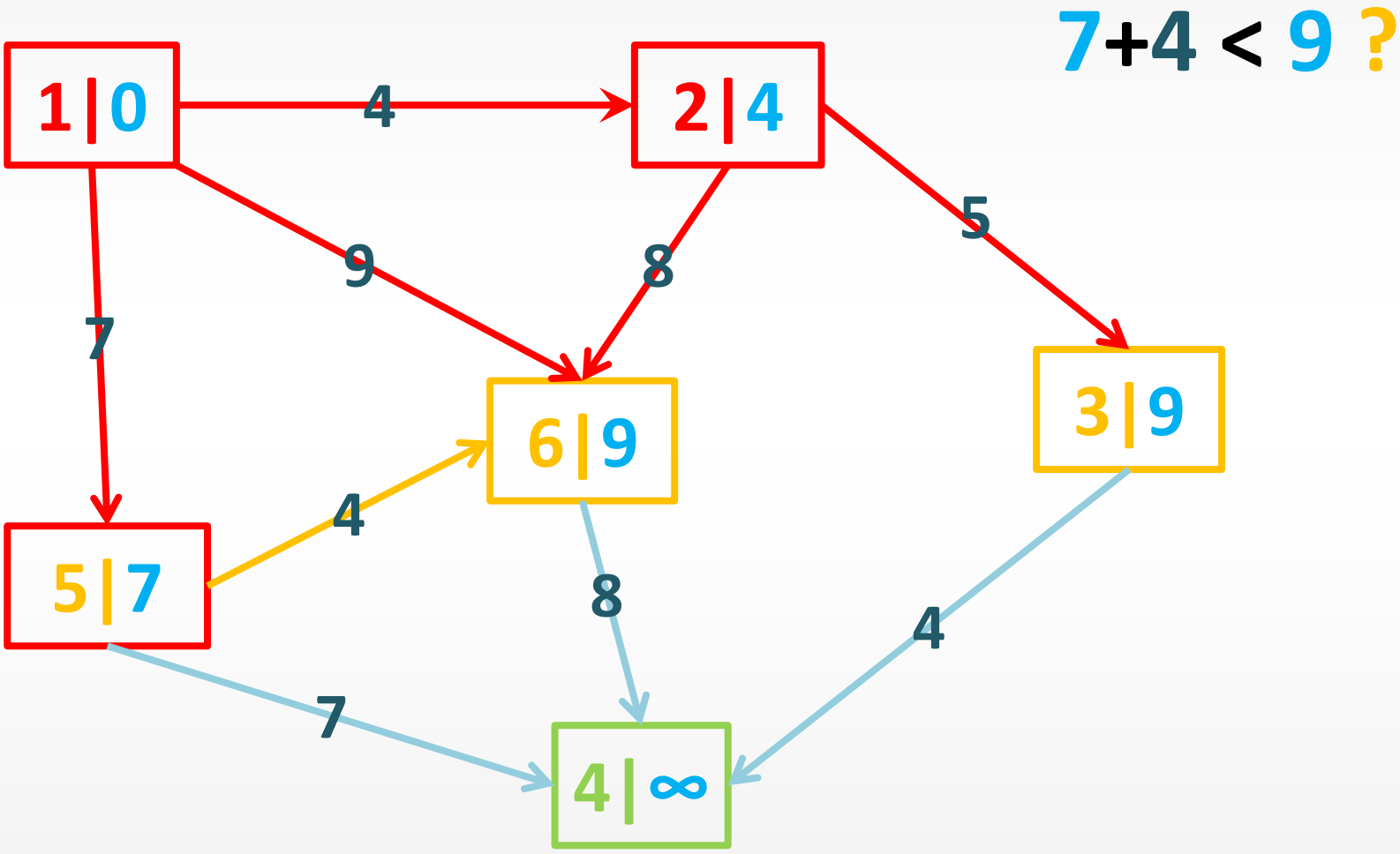
Алгоритм Дейкстры



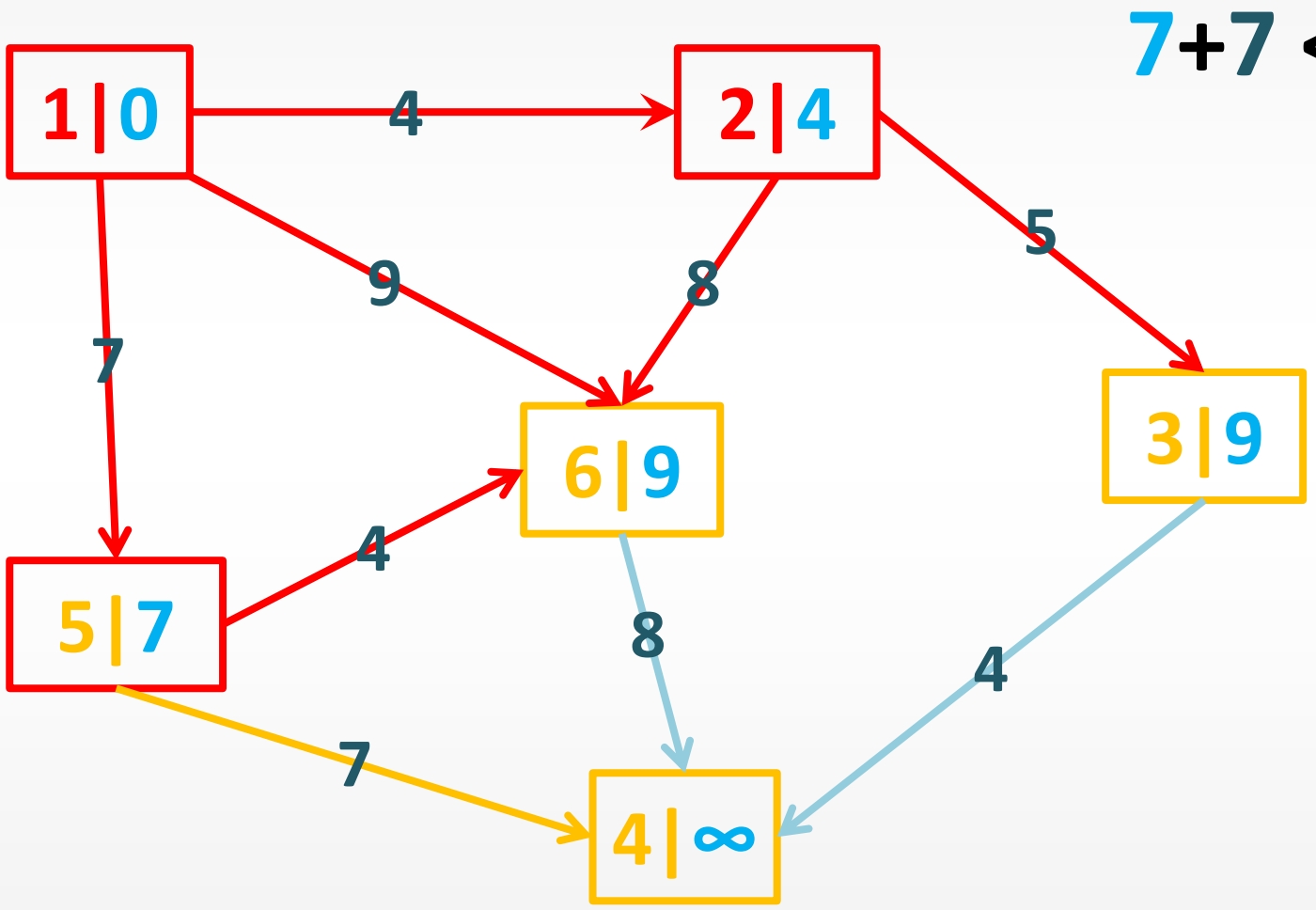
Алгоритм Дейкстры



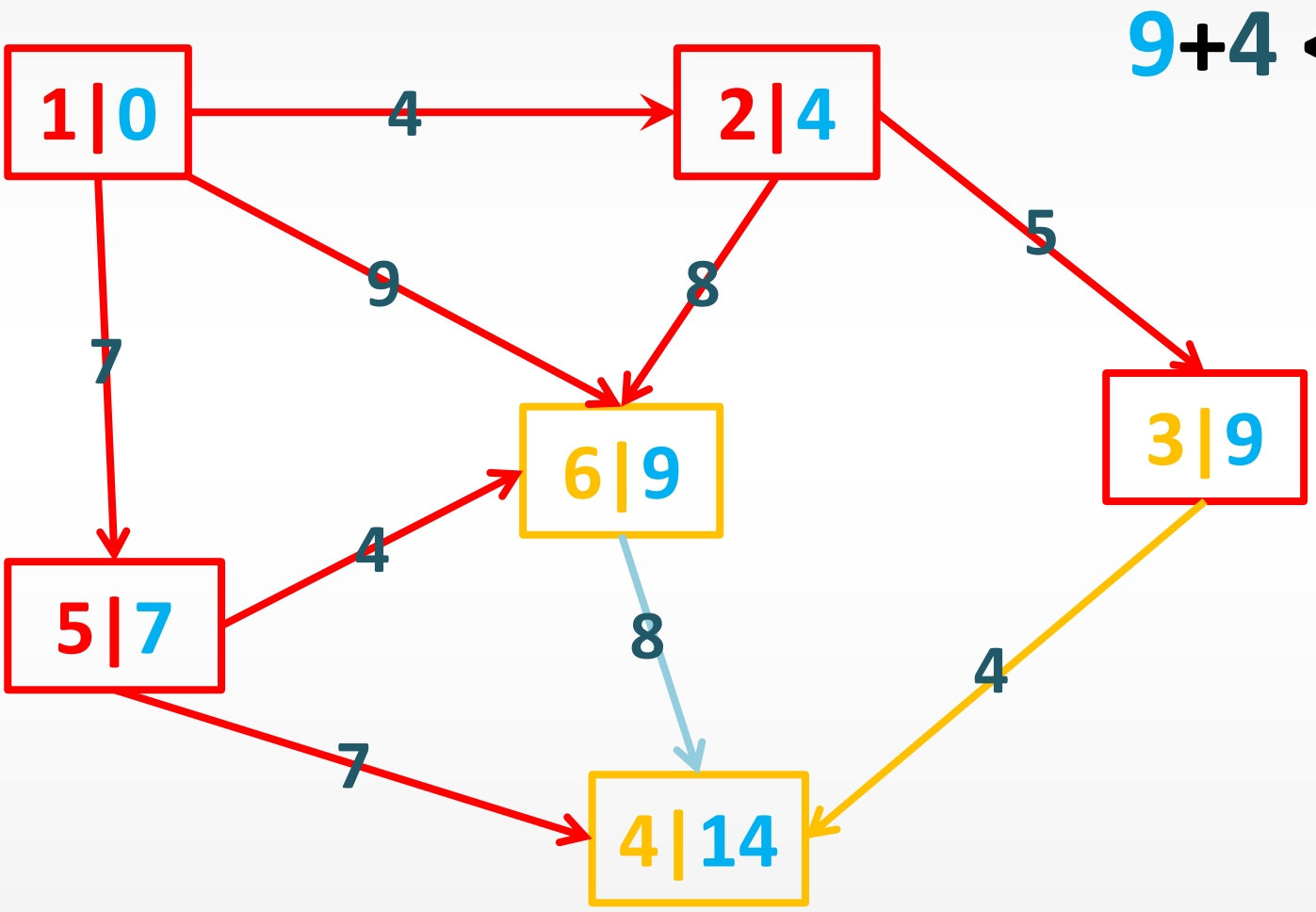
Алгоритм Дейкстры



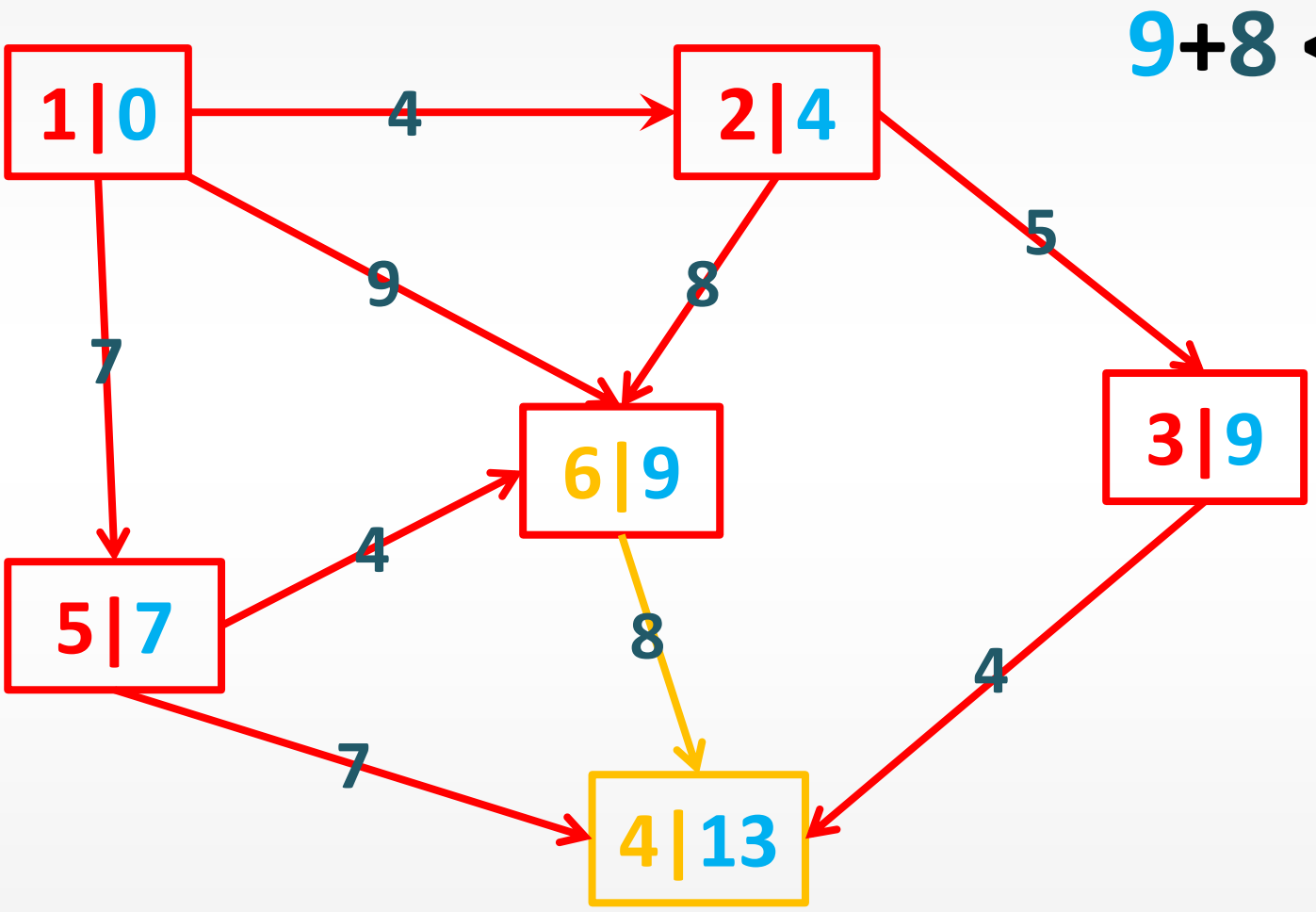
Алгоритм Дейкстры



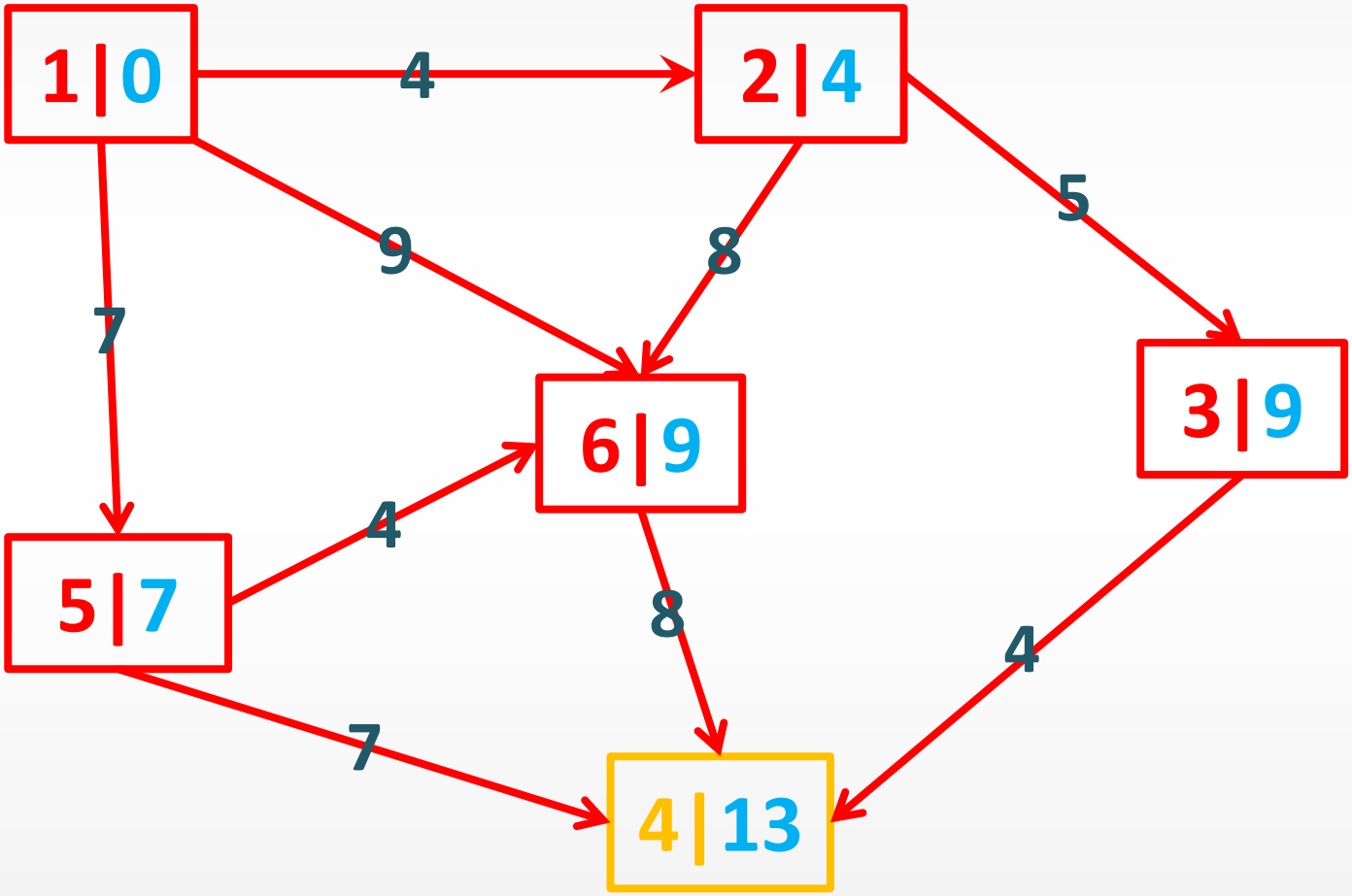
Алгоритм Дейкстры



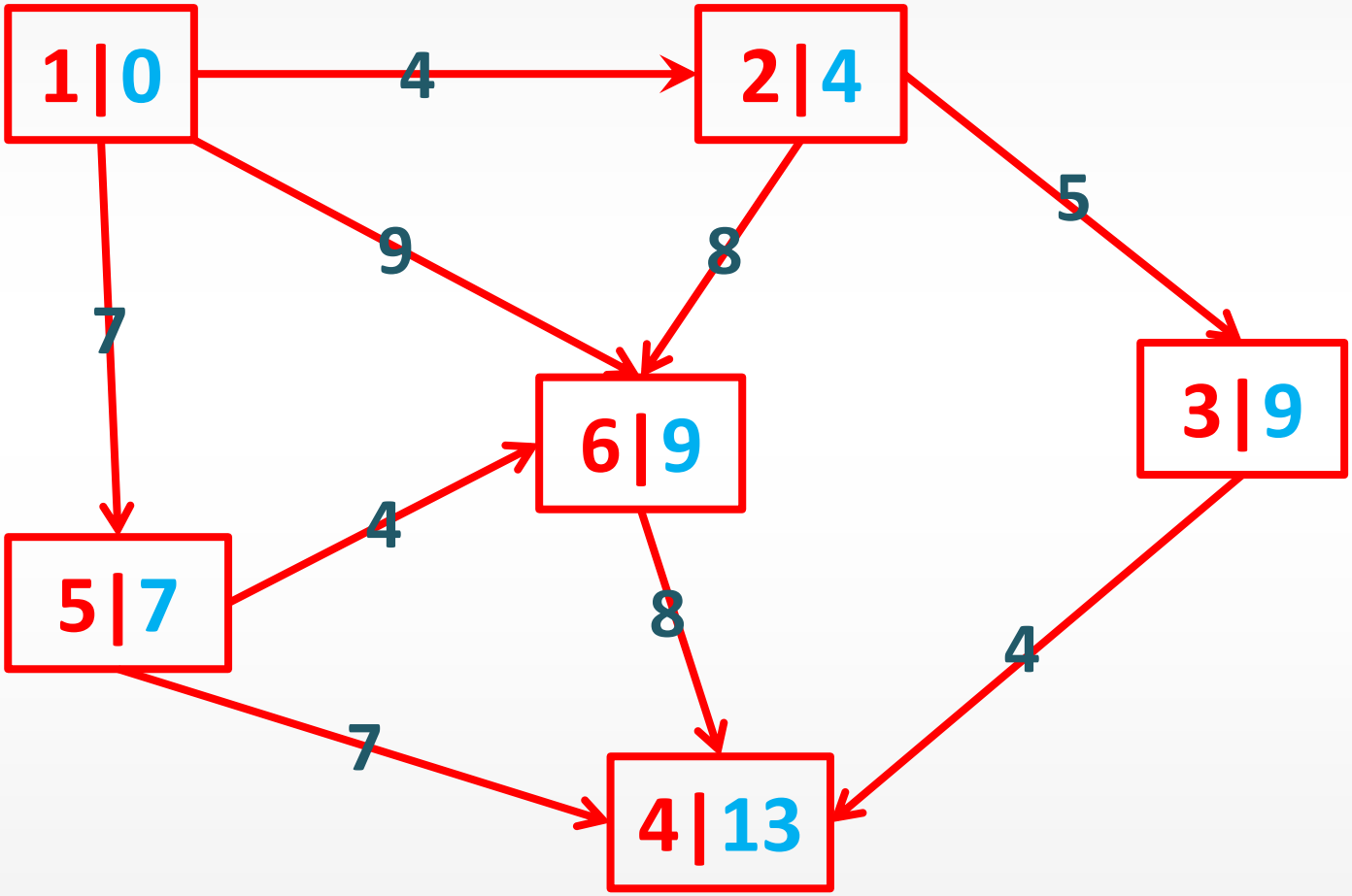
Алгоритм Дейкстры



Алгоритм Дейкстры



Алгоритм Дейкстры



Алгоритм Дейкстры

Этот алгоритм не работает в графах с рёбрами отрицательного веса!

Алгоритм Флойда(-Уоршелла)
(Floyd-Warshall algorythm)

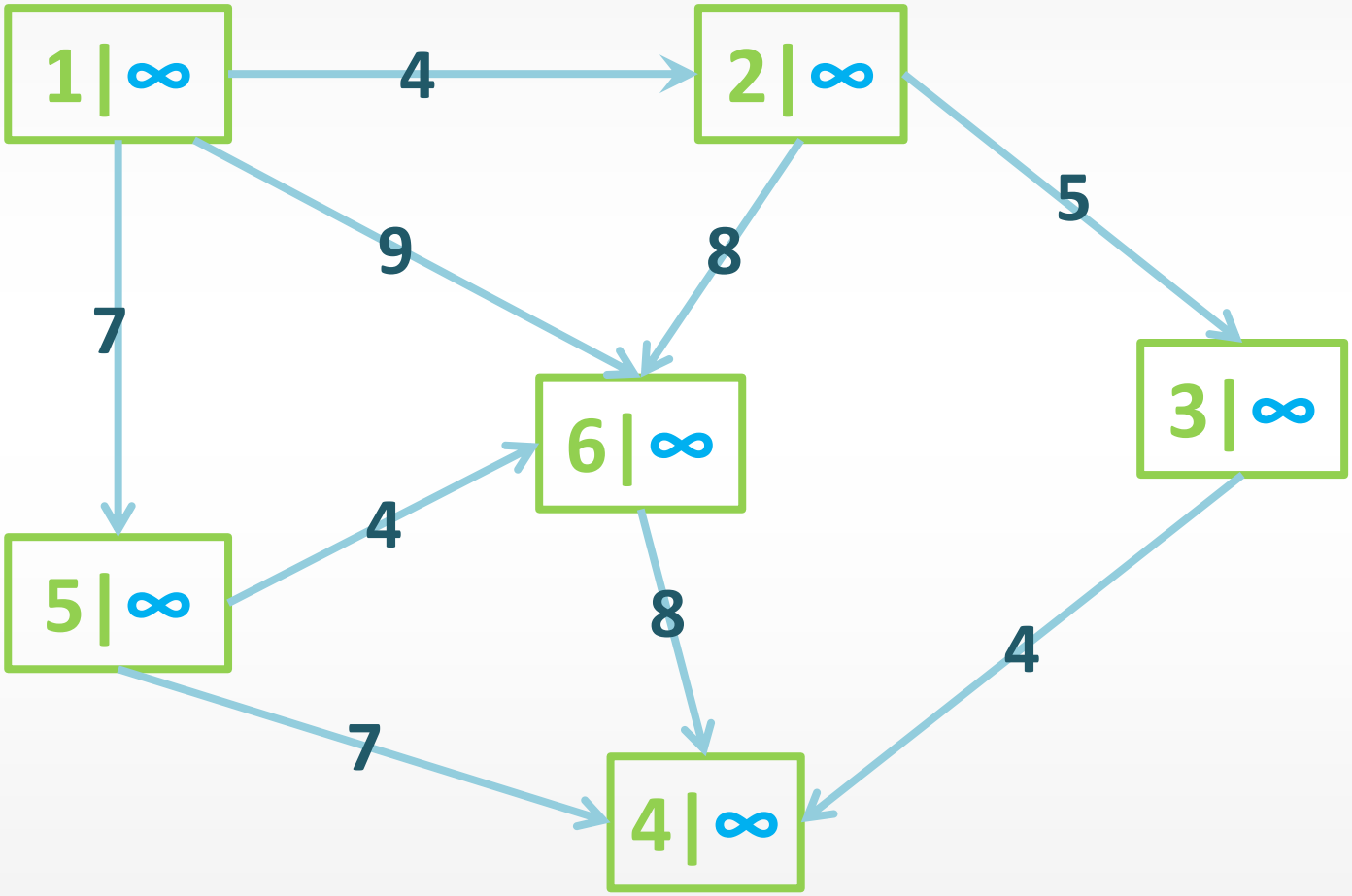
Для графов с отрицательными рёбрами существует решение ещё более сложное в понимании, но преступно простое в реализации. Этот алгоритм был разработан американскими учёными Робертом Флойдом и Стивеном Уоршеллом в 1962 году, и ищет расстояние между всеми возможными парами вершин графа.

Алгоритм Флойда(-Уоршелла)
(Floyd-Warshall algorythm)

Суть алгоритма:

- Изначально расстояния, известные алгоритму – это только длины рёбер, то есть матрица смежности графа.
- Пусть на каком-то шаге **i** нам точно известны все кратчайшие пути, использующие только вершины с номерами от **1** до **i-1**. Мы добавляем в это множество вершину с номером **i** и смотрим, стали ли какие-нибудь пути короче, и если стали – перезаписываем их.
- Для пары вершин **u** и **v** и множества вершин **[1,i]** кратчайшее расстояние от **u** до **v** считается так: **d[u][v] = min (d[u][v], d[u][i] + d[i][v])**

Алгоритм Флойда



Алгоритм Флойда

	1	2	3	4	5	6
1	∞	4	∞	∞	7	9
2	∞	∞	5	∞	∞	8
3	∞	∞	∞	4	∞	∞
4	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	7	∞	4
6	∞	∞	∞	8	∞	∞

Первая итерация: используем только вершину 1.

Алгоритм Флойда

	1	2	3	4	5	6
1	∞	4	∞	∞	7	9
2	∞	∞	5	∞	∞	8
3	∞	∞	∞	4	∞	∞
4	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	7	∞	4
6	∞	∞	∞	8	∞	∞

Вторая итерация: используем вершины 1 и 2. Новых путей нет, так как путь d[1][2] уже был кратчайшим, а пути d[2][1] не существует.

Алгоритм Флойда

	1	2	3	4	5	6
1	∞	4	9	∞	7	9
2	∞	∞	5	∞	∞	8
3	∞	∞	∞	4	∞	∞
4	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	7	∞	4
6	∞	∞	∞	8	∞	∞

Третья итерация: используем вершины с 1 по 3. Есть новый путь d[1][3], длина которого равна сумме d[1][2] и d[2][3].

Алгоритм Флойда

	1	2	3	4	5	6
1	∞	4	9	13	7	9
2	∞	∞	5	9	∞	8
3	∞	∞	∞	4	∞	∞
4	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	7	∞	4
6	∞	∞	∞	8	∞	∞

Четвёртая итерация:

используем вершины с 1 по 4. Есть новые пути:

d[1][4] = d[1][3] + d[3][4]

и

d[2][4] = d[2][3] + d[3][4]

Алгоритм Флойда

	1	2	3	4	5	6
1	∞	4	9	13	7	9
2	∞	∞	5	9	∞	8
3	∞	∞	∞	4	∞	∞
4	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	7	∞	4
6	∞	∞	∞	8	∞	∞

Больше новых кратчайших путей он не найдёт: пути через вершины 5 и 6 есть, но они слишком длинные.

Алгоритм Флойда

«Преступная простота» реализации алгоритма – три вложенных цикла и формула:

```
void floyd() {
    for(int i=0;i<n;i++)
        for(int u=0;u<n;u++)
            for(int v=0;v<n;v++)
                d[u][v] = min( d[u][v], d[u][i] + d[i][v] );
} ,
```

где d[u][v] – изначально матрица, идентичная матрице смежности графа.

Вопросы?