

2015/5/11 Mon.

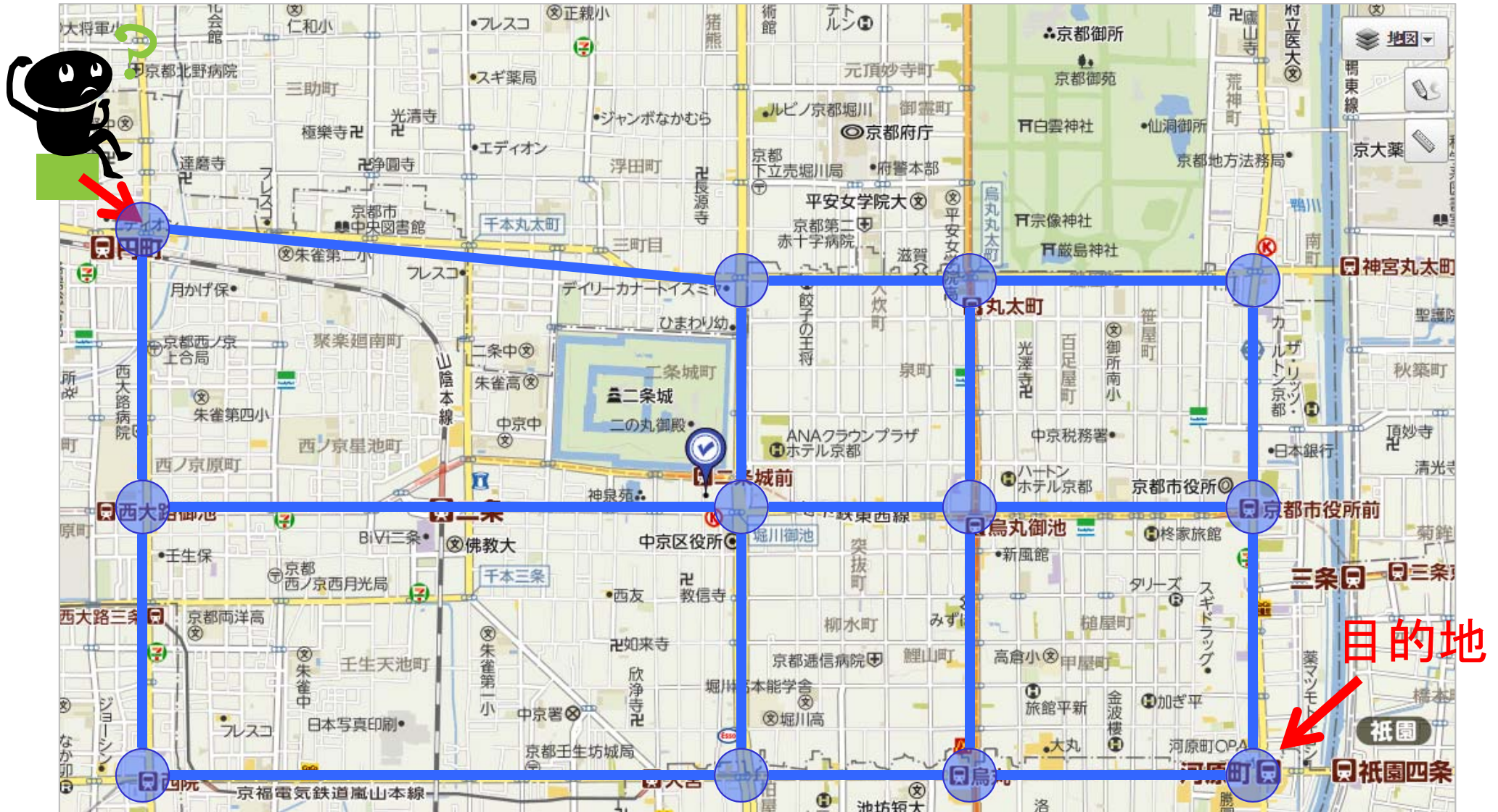
問題解決技法入門

最短経路探索

堀田 敬介

ルート探索

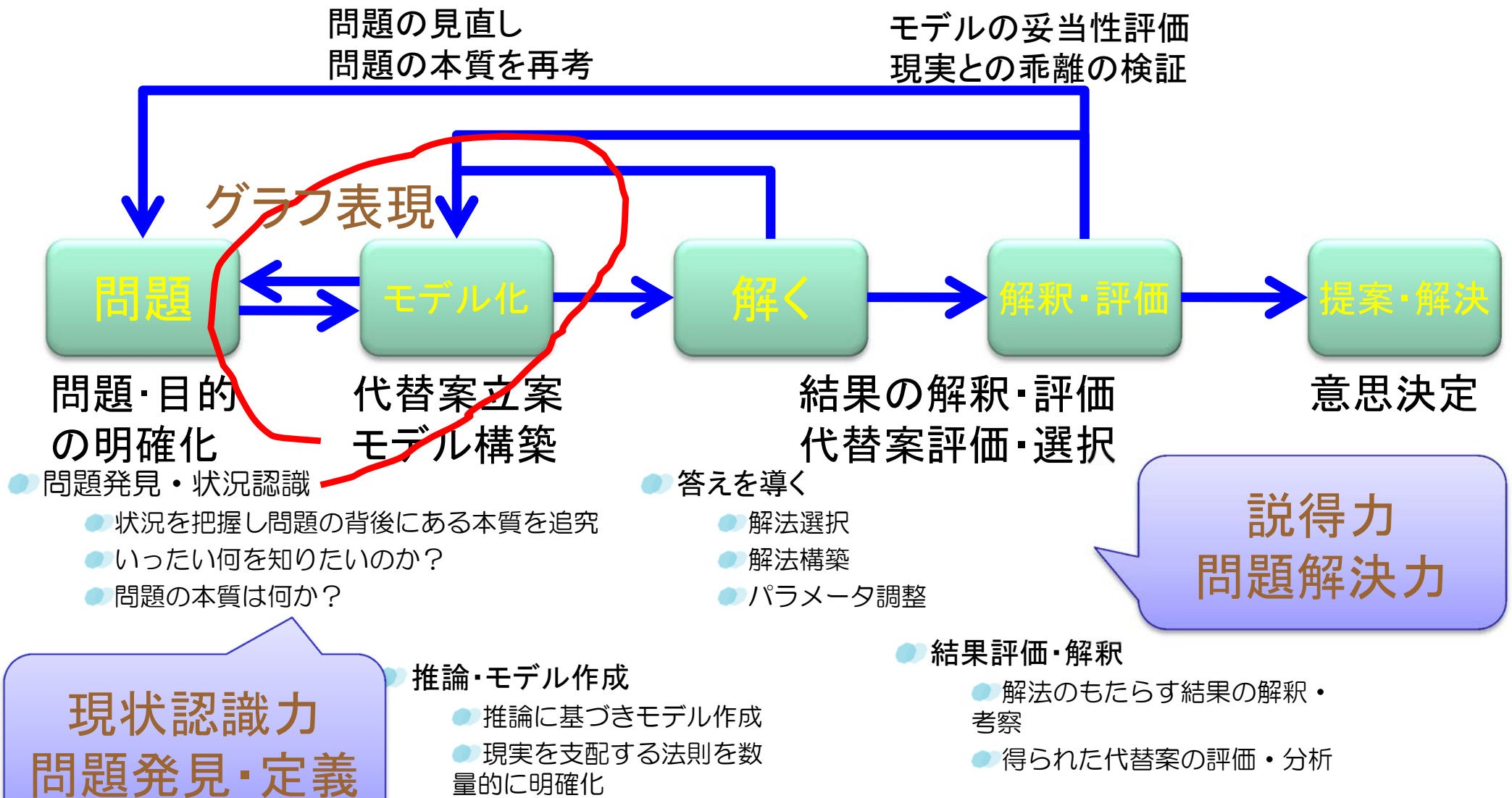
今、円町の交差点にいる 河原町の交差点まで、車で大通りのみを選んで通り、目的地までたどり着きたい どの経路(ルート)を通るのがよいか？



意思決定

論理的思考力
データ分析, 統計学
数理的アプローチ

- 「問題の把握」から「意思決定」までの流れ



参考：ナビゲーション・システム

航海, 航海術

- ナビゲーション・システムの2大機能

- 情報表示

- 現在地や渋滞情報, 周辺情報などを地図に重ねて表示

- ルート探索

- 目的地を指定すると現在地からの(最短)経路を表示

ex) カー・ナビゲーション・システム

automotive navigation system

PND・GPS携帯・
スマホなど多様化

どうやってるの？

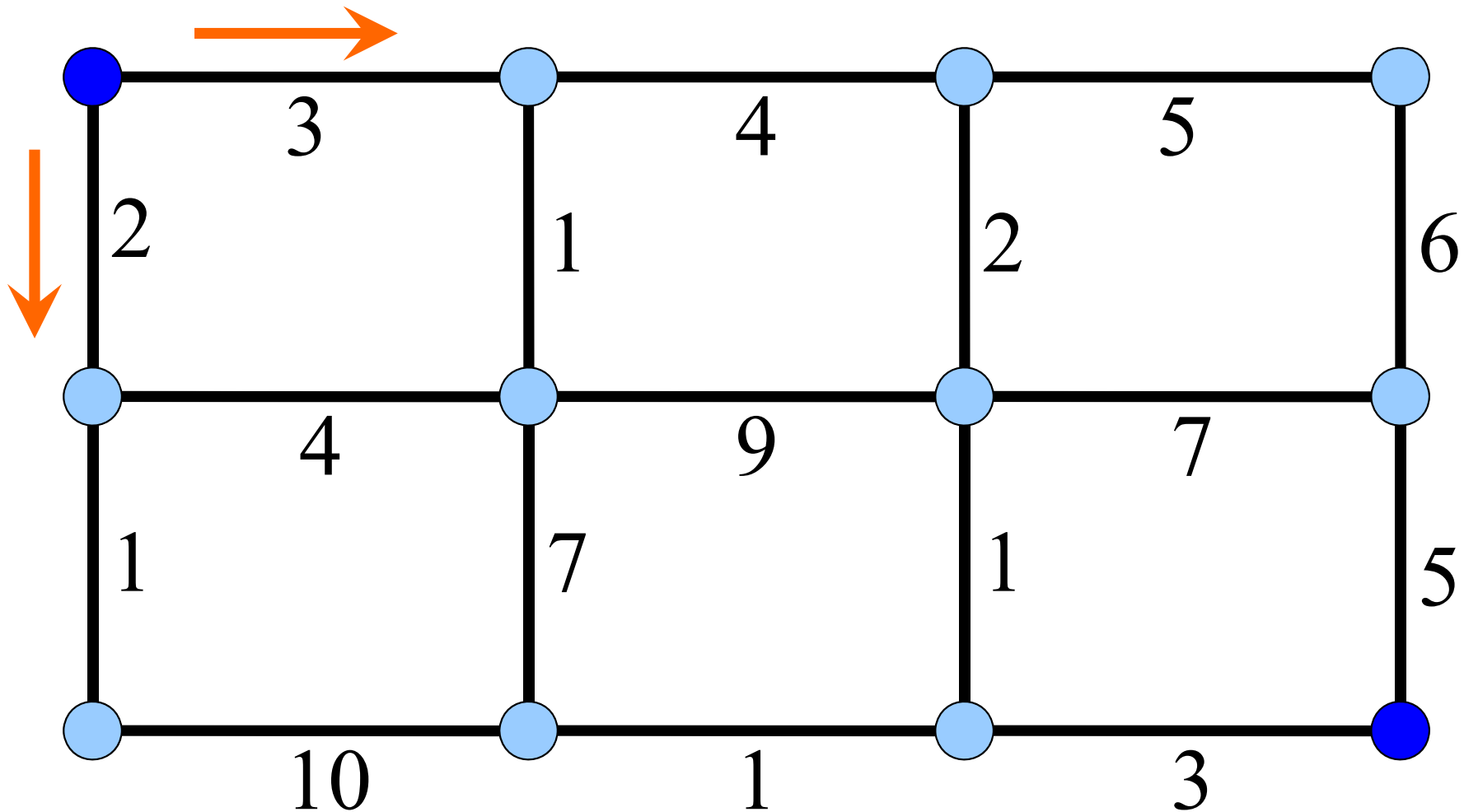


解く: どうやって解くか?

youtube
[erato お姉さん]で検索

全ての経路を調べ、
その中から最も短い経路を選べば良い!
[素朴で素直な方法 = 全列挙, しらみつぶし]

何通りあるか
お姉さんに聞いてみよう!

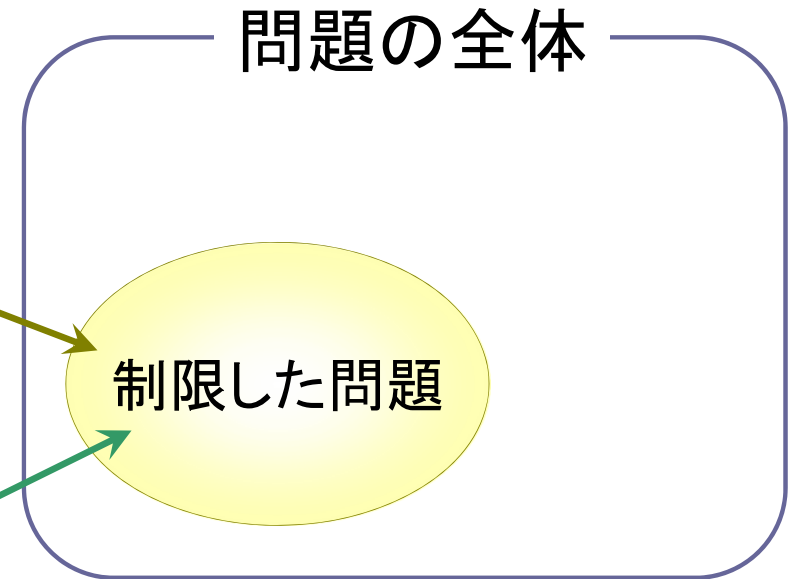


難しいなら易くすればいいのさ！

OR的問題解決のヒント
問題を**簡単**にする！

{ 問題の**一部**だけを考える
条件を**付加**して易くする

ここだけで考えて上手いけば、
全体に広げられるかも！



全てのネットワーク上の最短路問題

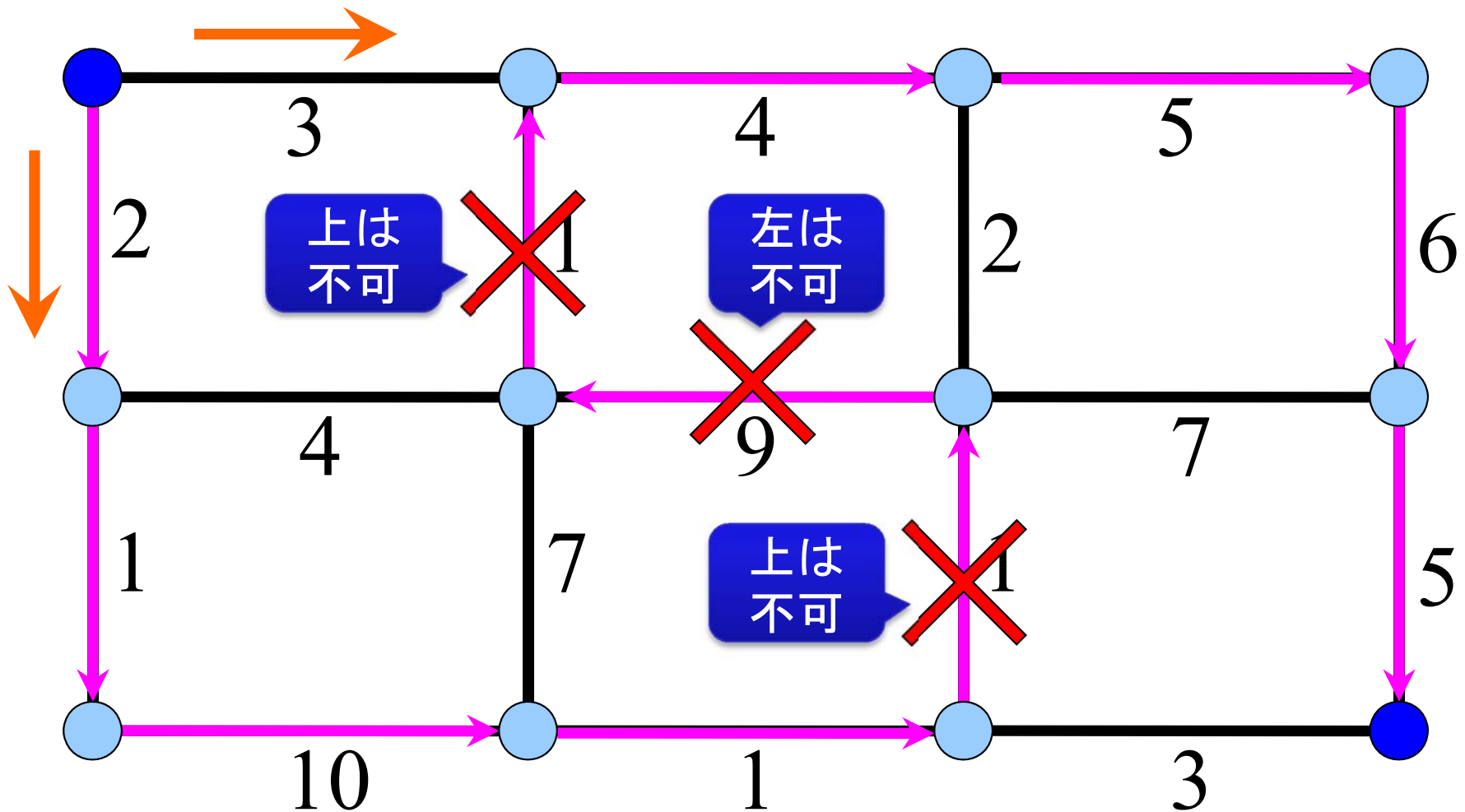
制限した問題

- 格子状のネットワーク
- 出発地: 左上点, 目的地: 右下点
- 移動は**右・下**方向へのみ

難しいなら易くすればいいのさ！

制限した問題

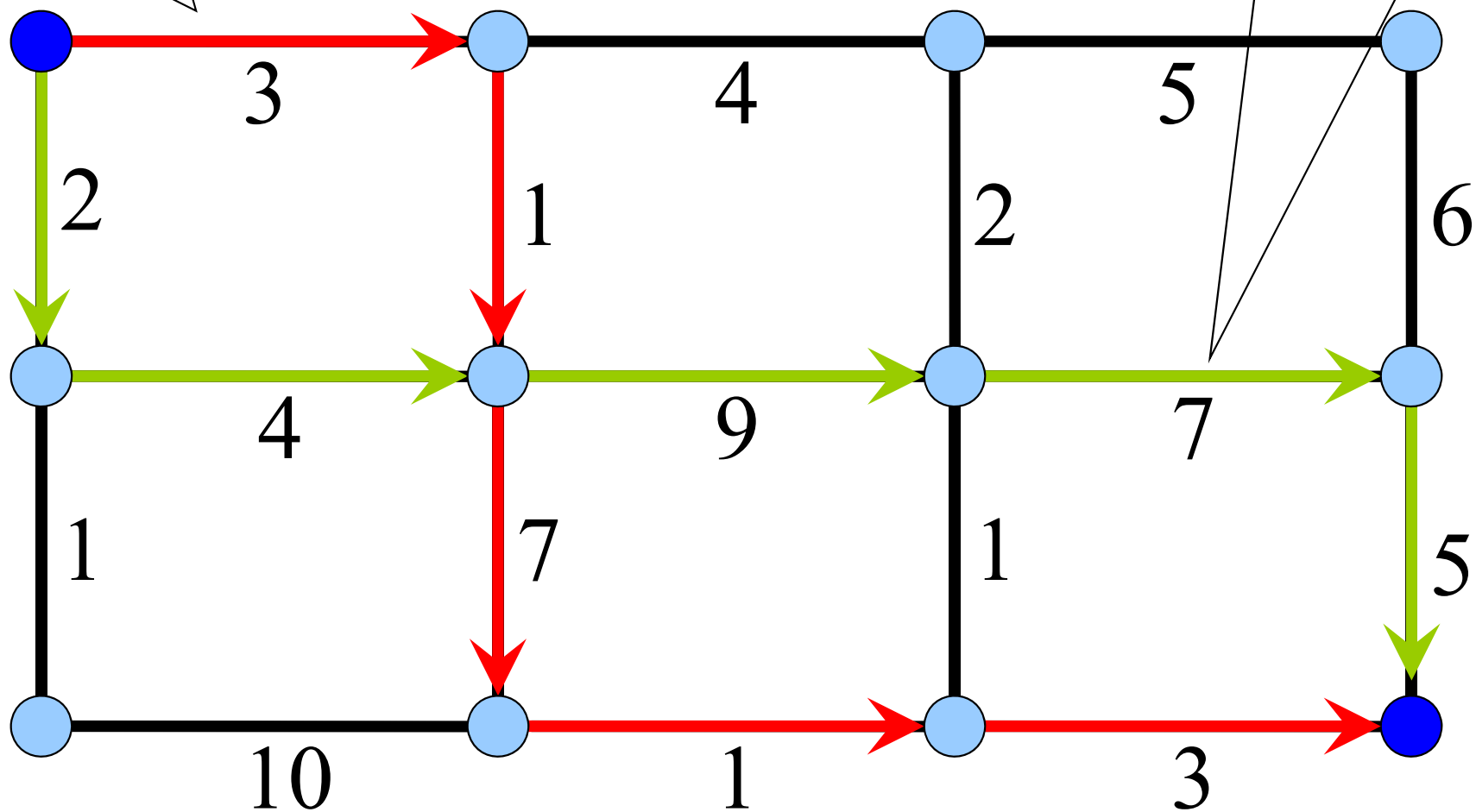
- 格子状のネットワーク
- 出発地: 左上点, 目的地: 右下点
- 移動は右・下方向へのみ



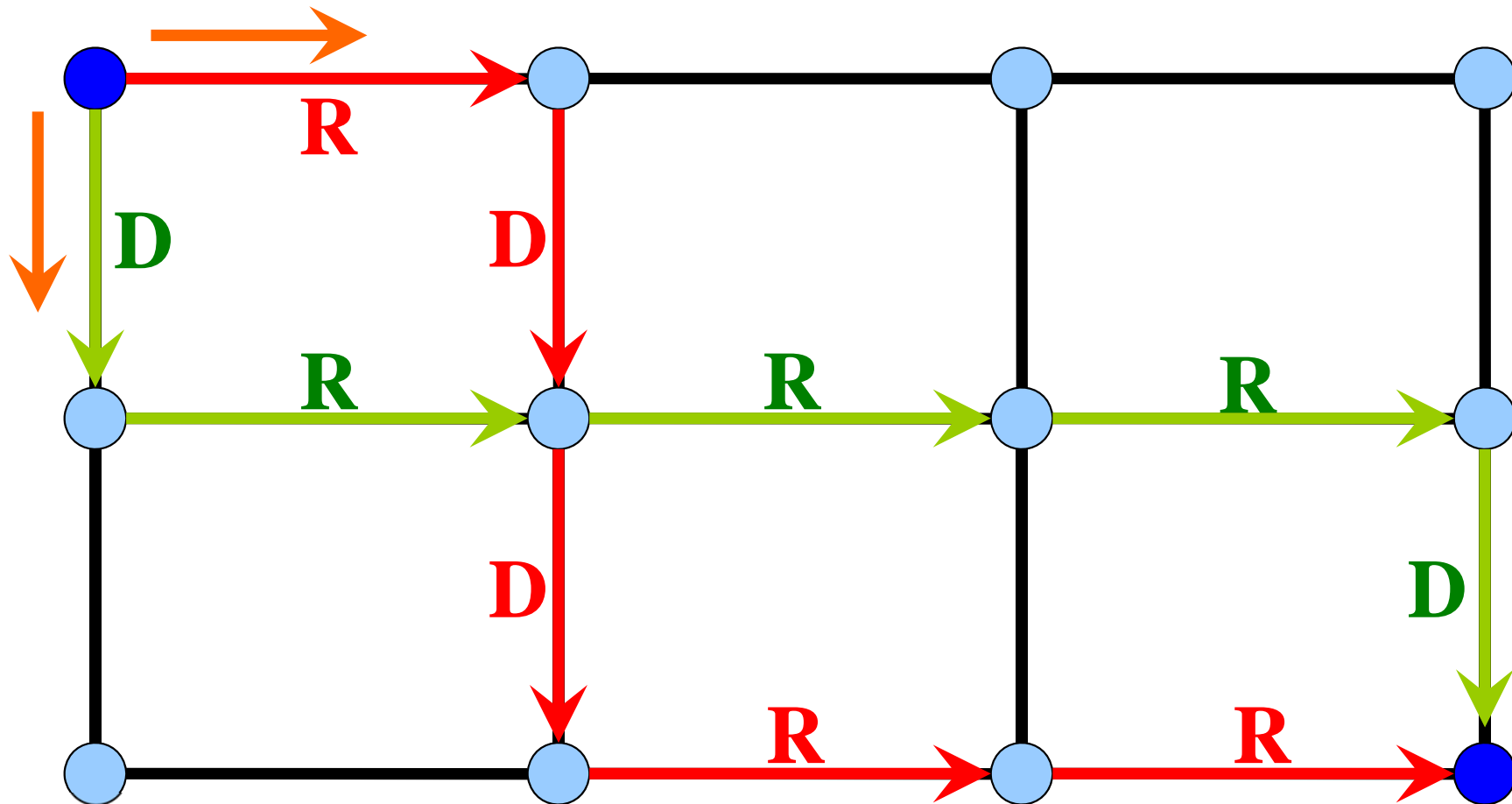
難しいなら易くすればいいのさ！

$$3+1+7+1+3 = 15$$

$$2+4+9+7+5 = 27$$



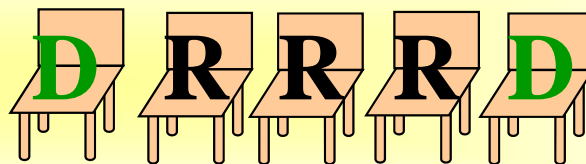
さて、経路は全部で幾つあるのか？



Point: どんな経路も、順番を無視すれば、R=3回、D=2回使う

緑の経路 = DRRRD

赤の経路 = RDDRR



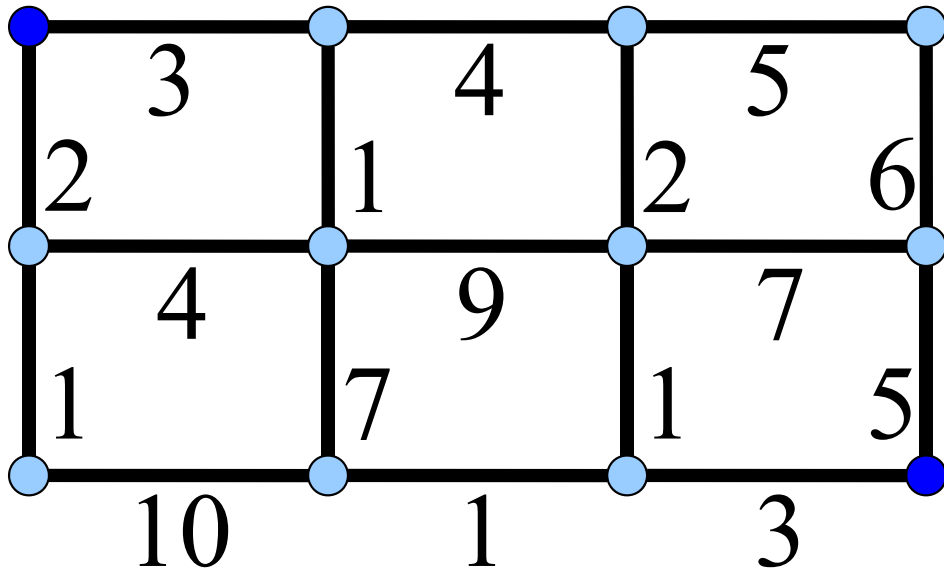
$$= \frac{(R+D)!}{R!D!} \text{通り}$$

i.e., (R+D)の椅子へのDの座らせ方を決めれば良い $\rightarrow {}_{R+D}C_D$

例では ${}_{3+2}C_2 = 10$ 通り

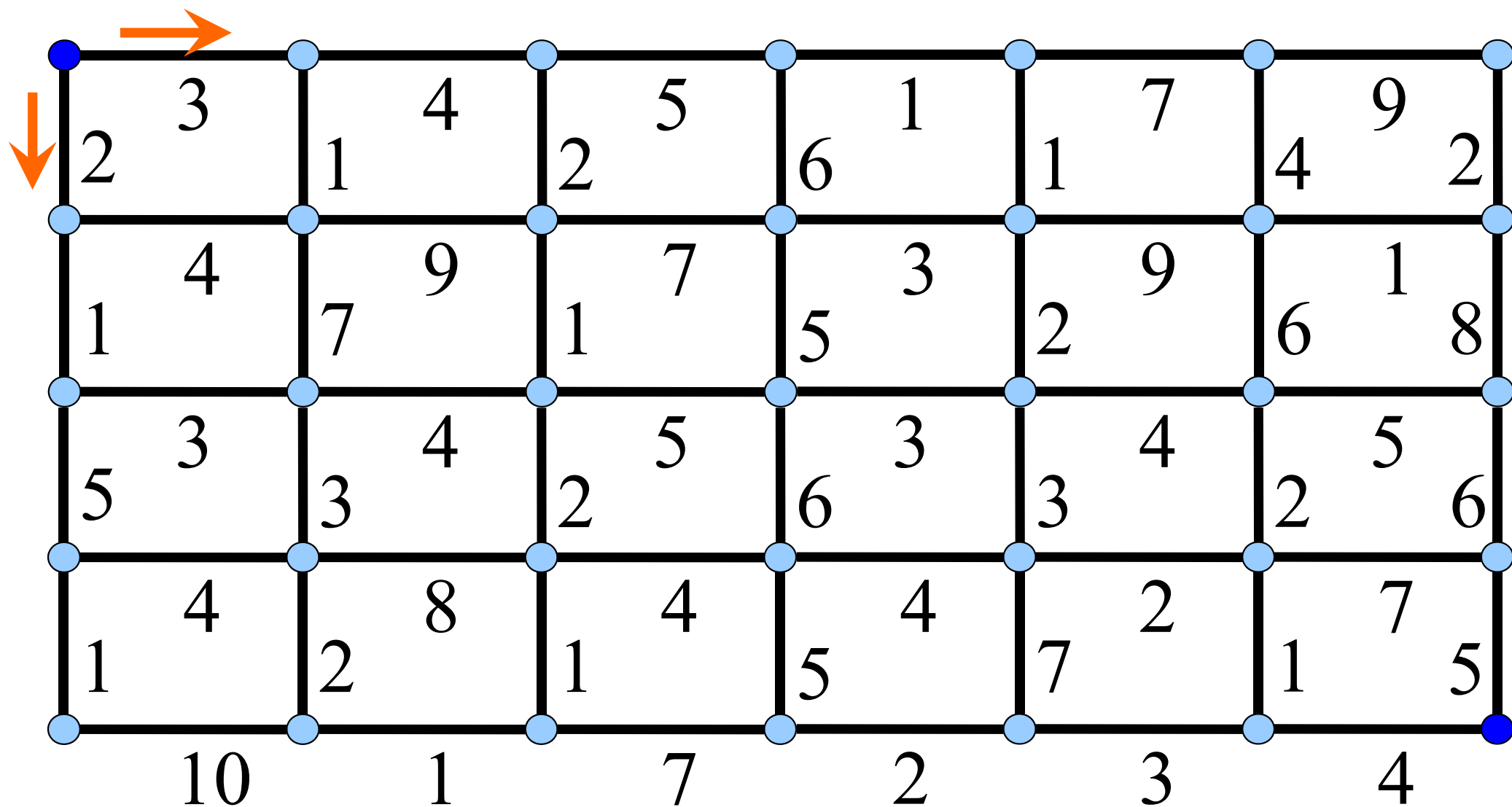
演習：やってみよう！全列挙

- **Q:** スタート(左上)からゴール(右下)へと至る**最短経路**を求めなさい. そして**それが最短だと示しなさい**



- ① **DDRRR**: $2+1+10+1+3=17$
 - ② **DRDRR**: $2+4+7+1+3=17$
 - ③ **DRRDR**: $2+4+9+1+3=19$
 - ④ **DRRRD**: $2+4+9+7+5=27$
 - ⑤ **RDDRR**: $3+1+7+1+3=15$
 - ⑥ **RDRDR**: $3+1+9+1+3=17$
 - ⑦ **RDRRD**: $3+1+9+7+5=25$
 - ⑧ **RRDDR**: $3+4+2+1+3=13$
 - ⑨ **RRDRD**: $3+4+2+7+5=21$
 - ⑩ **RRRDD**: $3+4+5+6+5=23$
- **A:** 全列挙したよ ①～⑩
の**10通り**計算し**⑧が最短だ!**

経路は全部で幾つ？

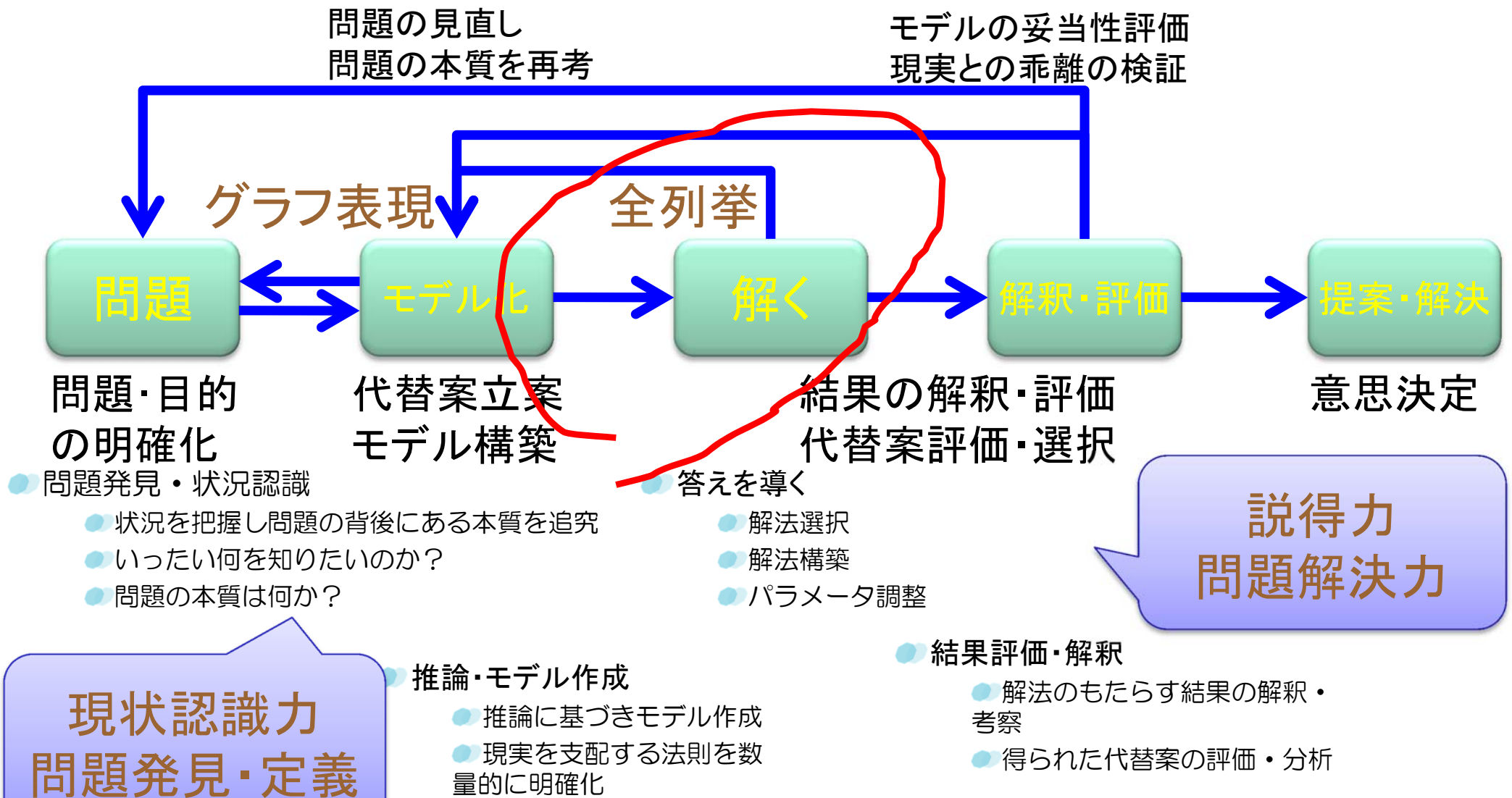


R=6, D=4なので, ${}_{6+4}C_4 = \frac{10 \cdot 9 \cdot 8 \cdot 7}{4 \cdot 3 \cdot 2 \cdot 1} = 210$ 通り

意思決定

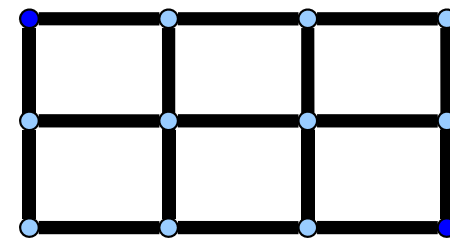
論理的思考力
データ分析, 統計学
数理的アプローチ

- 「問題の把握」から「意思決定」までの流れ



経路は全部で幾つ？【全列挙】

| R(横) | D(縦) | 全経路 |
|------|------|-----------------------|
| 3 | 2 | 10 |
| 6 | 4 | 210 |
| 10 | 5 | 3,003 |
| 20 | 10 | 30,045,015 |
| 50 | 50 | 1.0×10^{29} |
| 100 | 100 | 9.1×10^{58} |
| 500 | 500 | 2.7×10^{299} |
| 1000 | 1000 | #NUM! |



【格子道路の街】
cf.京都市,札幌市
R, D幾つぐらい？

経路は全部で幾つ？【全列挙】

経路がとてもたくさんあるとは言っても、今のコンピュータは
かなりの速さで計算できるんでしょ？ だから大丈夫だよな！

- 代表的なCPU, Game機, super computer の 浮動小数点演算回数
 - Intel Core i7(3.2GHz) : **51.2GFLOPS** ...1秒間に約**512億**回
 - PS3 : **218GFLOPS** ...1秒間に約**2180億**回
 - PS4 : **1.84TFLOPS** ...1秒間に約**1兆8400億**回
 - 京 : **10.51PFLOPS** ...1秒間に約**1京510兆**回

(※2011年6月, 11月 **世界最速!** by Top500.org)
(※2012年6月=2位, 11月=3位, 2013年6月=4位, 11月=4位)

※**FLOPS** = *FL*oating-*point* *O*perations *P*er *S*econd

[Wikipedia「FLOPS」より]
2013/5/1の情報

1つの経路を見つけ、その総コストを計算するの
に、たどる経路枝数の浮動小数点演算でできると仮定しよう

例えば、R=10, D=5の経路なら、10+5回の演算で計算可と仮定するということ

K(キロ) $\approx \times 10^3 =$ 千倍
M(メガ) $\approx \times 10^6 =$ 百万倍
G(ギガ) $\approx \times 10^9 =$ 10億倍
T(テラ) $\approx \times 10^{12} =$ 1兆倍
P(ペタ) $\approx \times 10^{15} =$ 千兆倍
E(エクサ) $\approx \times 10^{18} =$ 百京倍

経路は全部で幾つ？【全列挙】

1.84TFLOPS

10.51PFLOPS

| R(横) D(縦) | 全経路 | PS4 | 京 |
|-----------|-----------------------|--------------------------|--------------------------|
| 3 2 | 10 | 0.000000000 秒 | 0.000000000 秒 |
| 6 4 | 210 | 0.000000001 秒 | 0.000000000 秒 |
| 10 5 | 3,003 | 0.000000024 秒 | 0.000000000 秒 |
| 20 10 | 30,045,015 | 0.000489864 秒 | 0.000000086 秒 |
| 25 25 | 1.3×10^{14} | 57 分 | 0.601382523 秒 |
| 30 30 | 1.2×10^{17} | 45 日 | 11 分 |
| 40 40 | 1.1×10^{23} | 148,219 年 | 26 年 |
| 50 50 | 1.0×10^{29} | 1.7×10^{11} 年 | 30,439,996 年 |
| 100 100 | 9.1×10^{58} | 2.3×10^{31} 宙齡 | 4.0×10^{27} 宙齡 |
| 500 500 | 2.7×10^{299} | 3.4×10^{272} 宙齡 | 5.9×10^{268} 宙齡 |

圧倒的な計算力をもつコンピュータですら、**全列挙(しらみつぶし)**では答えを求めることが出来ない！

1宙齡 = 138億年



参考：大きい数を表す接頭辞

- 万(まん) $\times 10^4$
- 億(おく) $\times 10^8$
- 兆(ちょう) $\times 10^{12}$
- 京(けい) $\times 10^{16}$
- 垓(がい) $\times 10^{20}$
- 杼(じょ) $\times 10^{24}$
- 穰(じょう) $\times 10^{28}$
- 溝(こう) $\times 10^{32}$
- 澗(かん) $\times 10^{36}$
- 正(せい) $\times 10^{40}$
- 載(さい) $\times 10^{44}$
- 極(ごく) $\times 10^{48}$
- 恒河沙(ごうがしゃ) $\times 10^{52}$
- 阿僧祇(あそうぎ) $\times 10^{56}$
- 那由他(なゆた) $\times 10^{60}$
- 不可思議(ふかしぎ) $\times 10^{64}$
- 無量大数(むりょうたいすう) $\times 10^{68}$

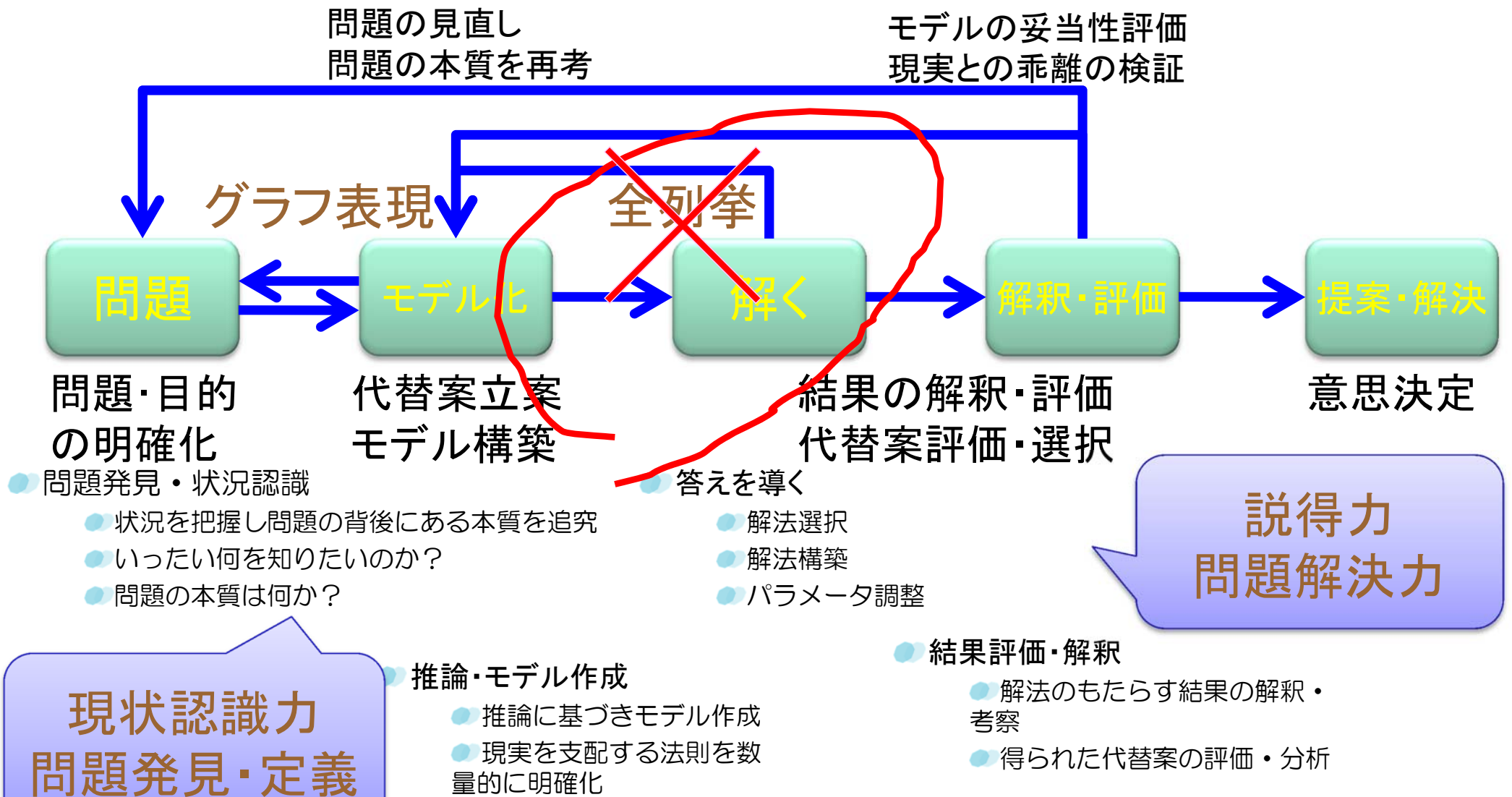
【注】「杼」は正しくは「のぎへん」(らしい)

【注】「無量大数」は「無限大 ∞ 」とは違う

意思決定

論理的思考力
データ分析, 統計学
数理的アプローチ

- 「問題の把握」から「意思決定」までの流れ



ではどうする？

- 素朴で素直な方法〔列挙法〕
 - 全経路をしらみつぶしに調べて、最も短い経路を見つける方法

時間が掛かり過ぎる！



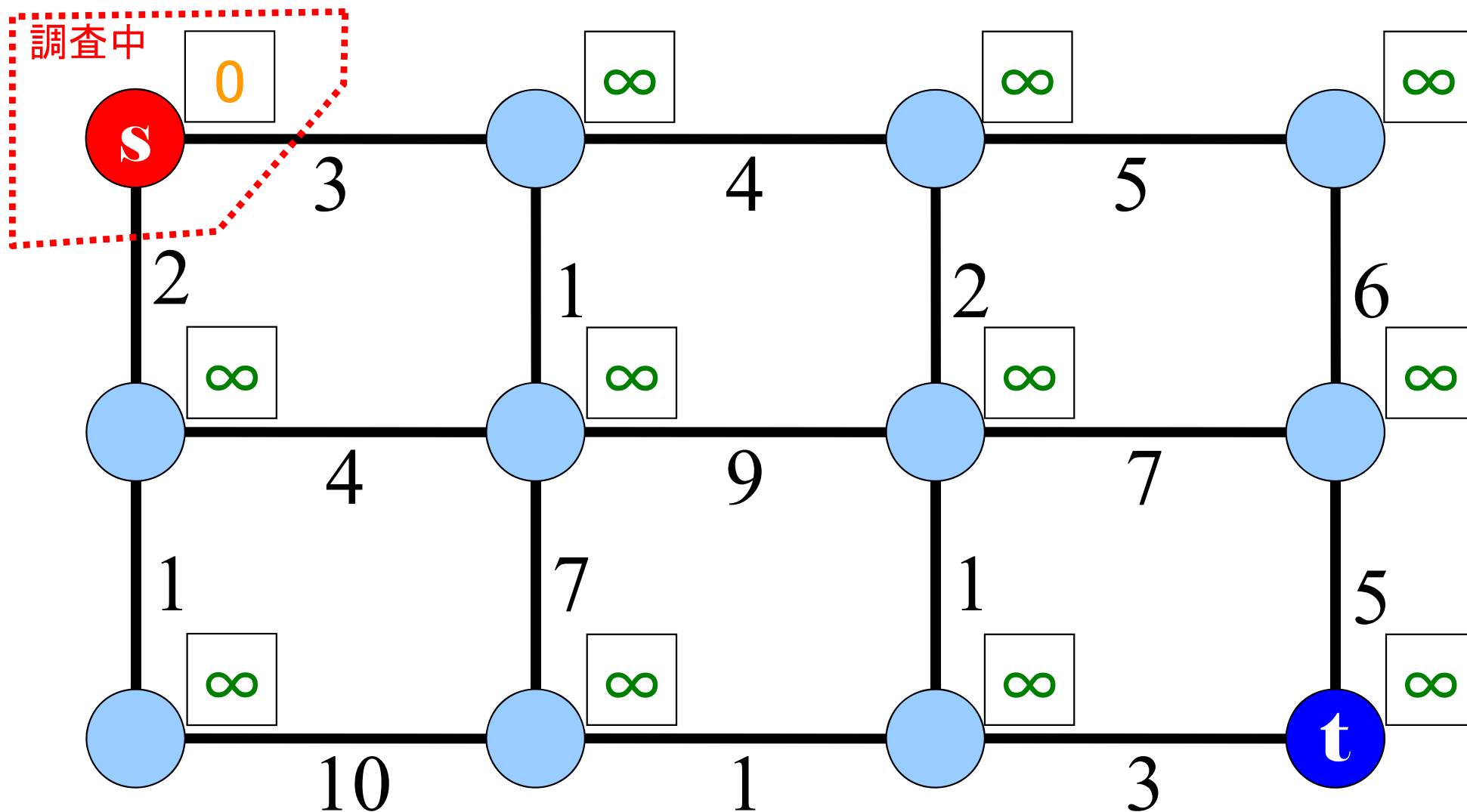
全経路をしらみつぶしに調べずに、最も短い経路を、現実的時間で見つける方法があるか？

Dijkstra法
(ダイクストラ法)

人間の創造的な仕事！

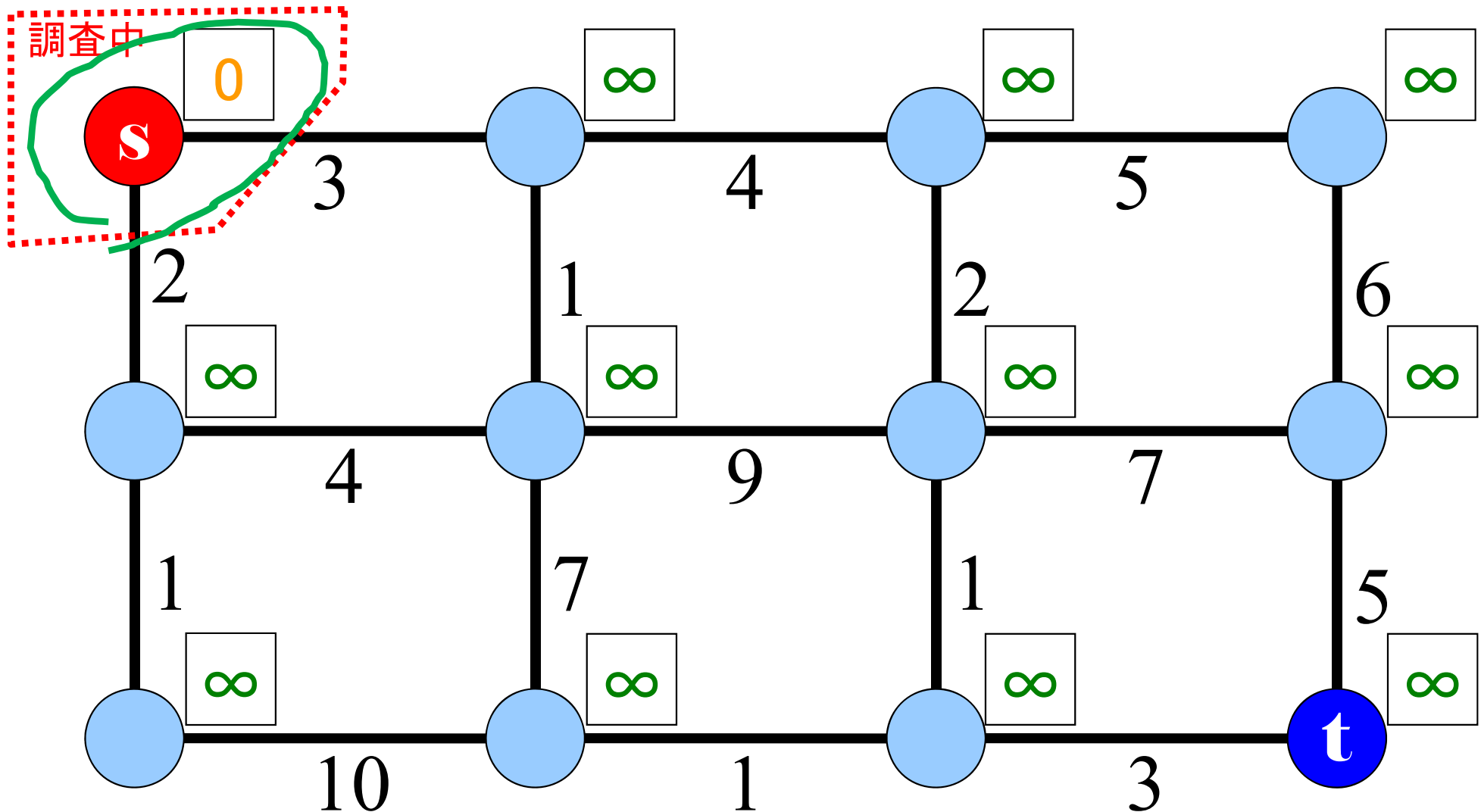
Dijkstra法 (初期設定)

step0: start点のラベルを0にし, その他のラベルを ∞ に設定する
start点 (**S**) を 調査中 の点集合とする



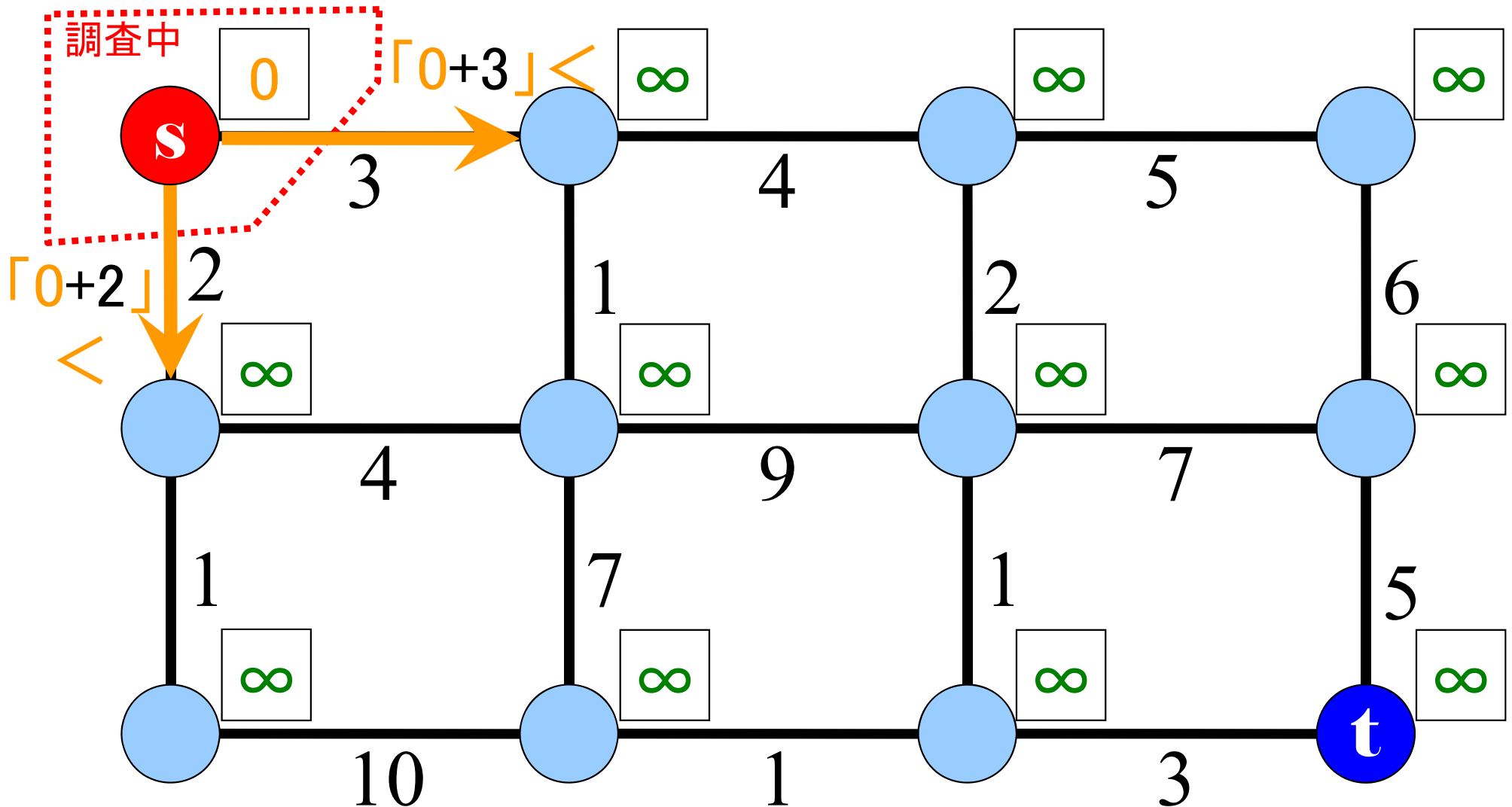
Dijkstra法 (更新法)

step1-1: 調査中点のうちで、ラベル値が最小の点を見つける



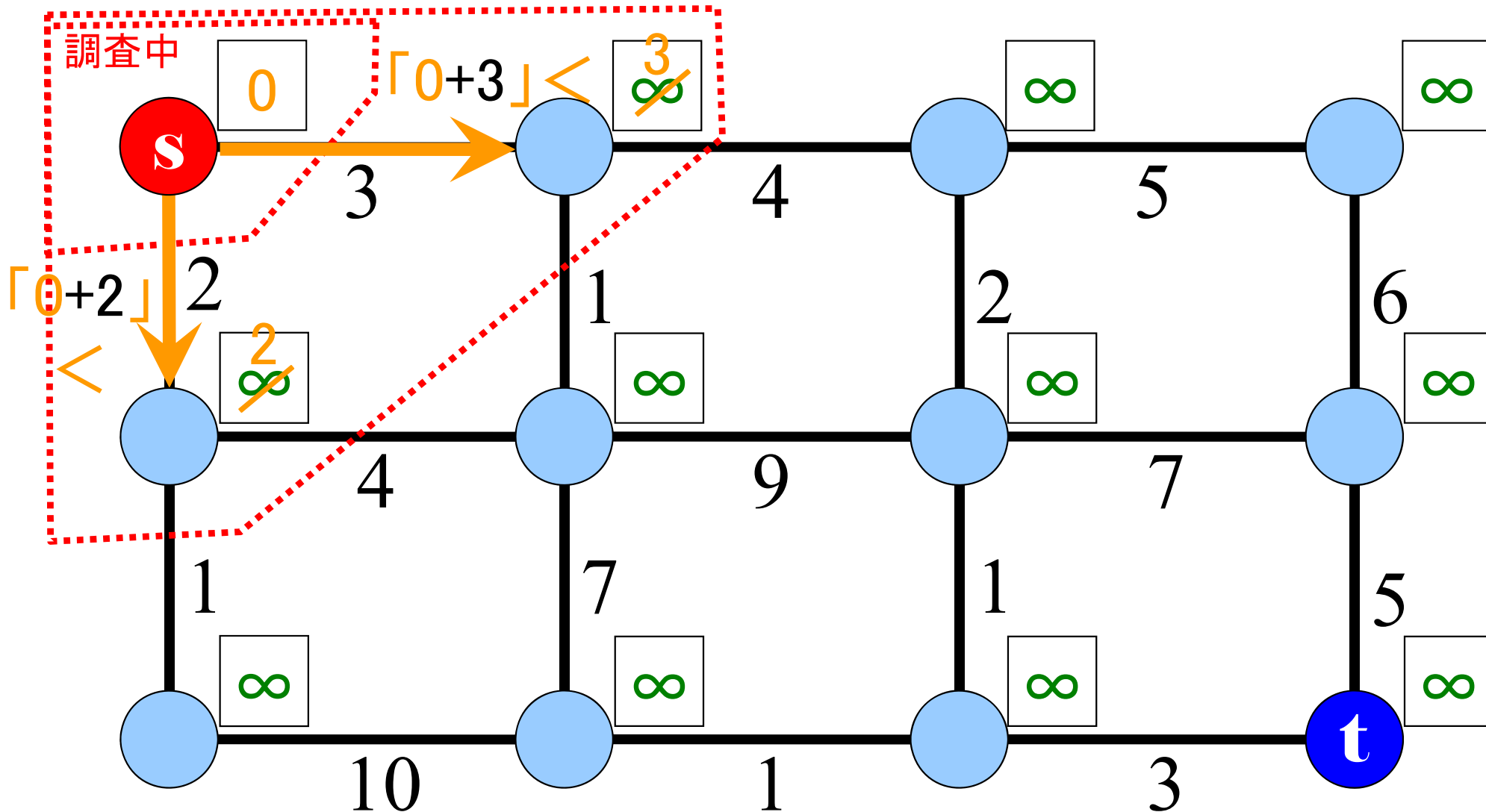
Dijkstra法 (更新法)

step1-2: その点から出る全枝について「ラベル+枝コスト」を計算し、枝先点のラベル値と比較し、もし、小さい($<$)→枝先点のラベル更新(暫定最短路)し、枝先点を調査中に追加, そうでないなら何もしない



Dijkstra法 (更新法)

step1-2: その点から出る全枝について「ラベル+枝コスト」を計算し、枝先点のラベル値と比較し、もし、小さい($<$)→枝先点のラベル更新(暫定最短路)し、枝先点を調査中に追加, そうでないなら何もしない



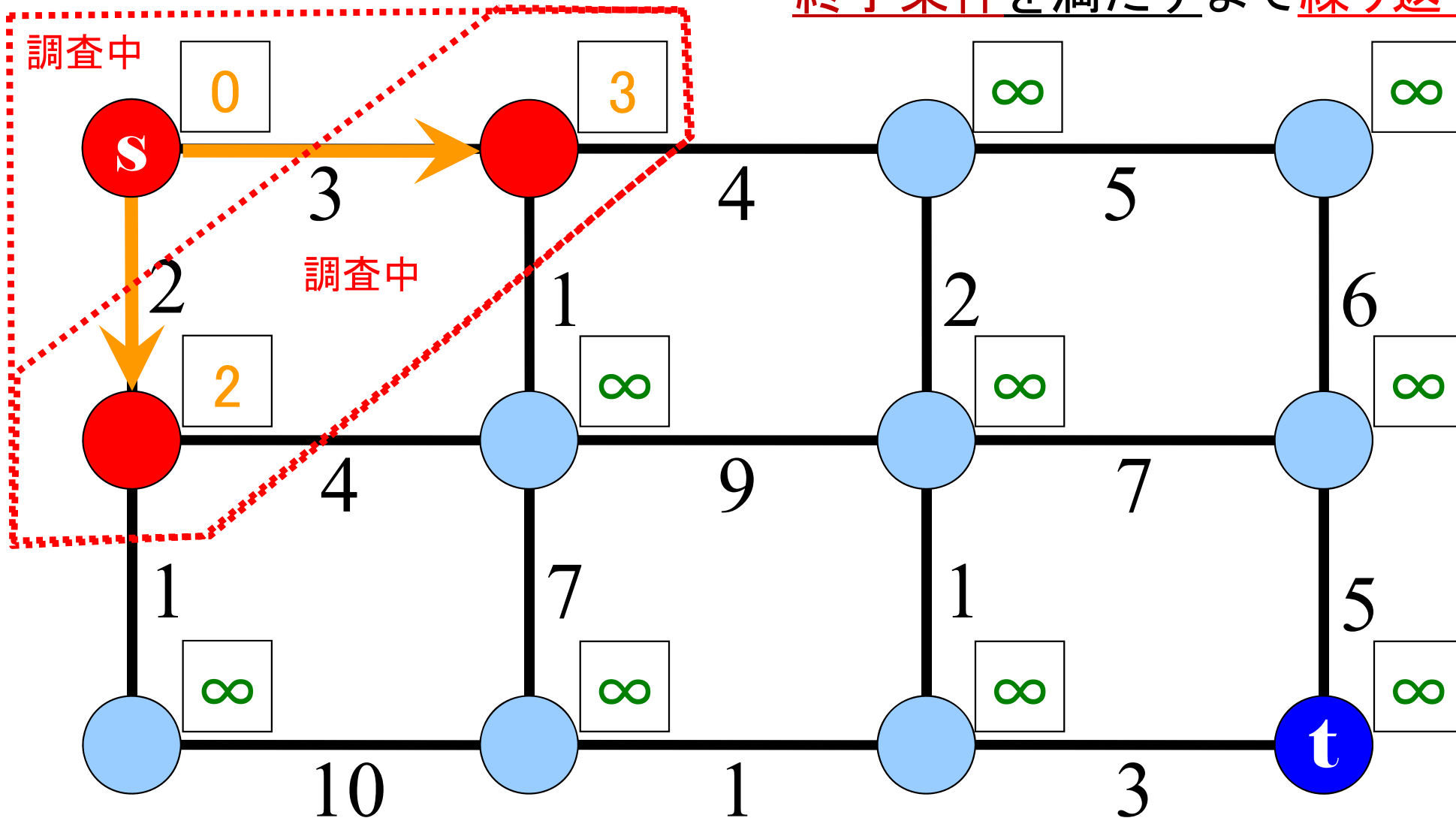
Dijkstra法

(更新法)

step1-3: 全枝終了したら, 調査中から外しラベル値確定
(その点までの暫定最短路が最短路として確定)

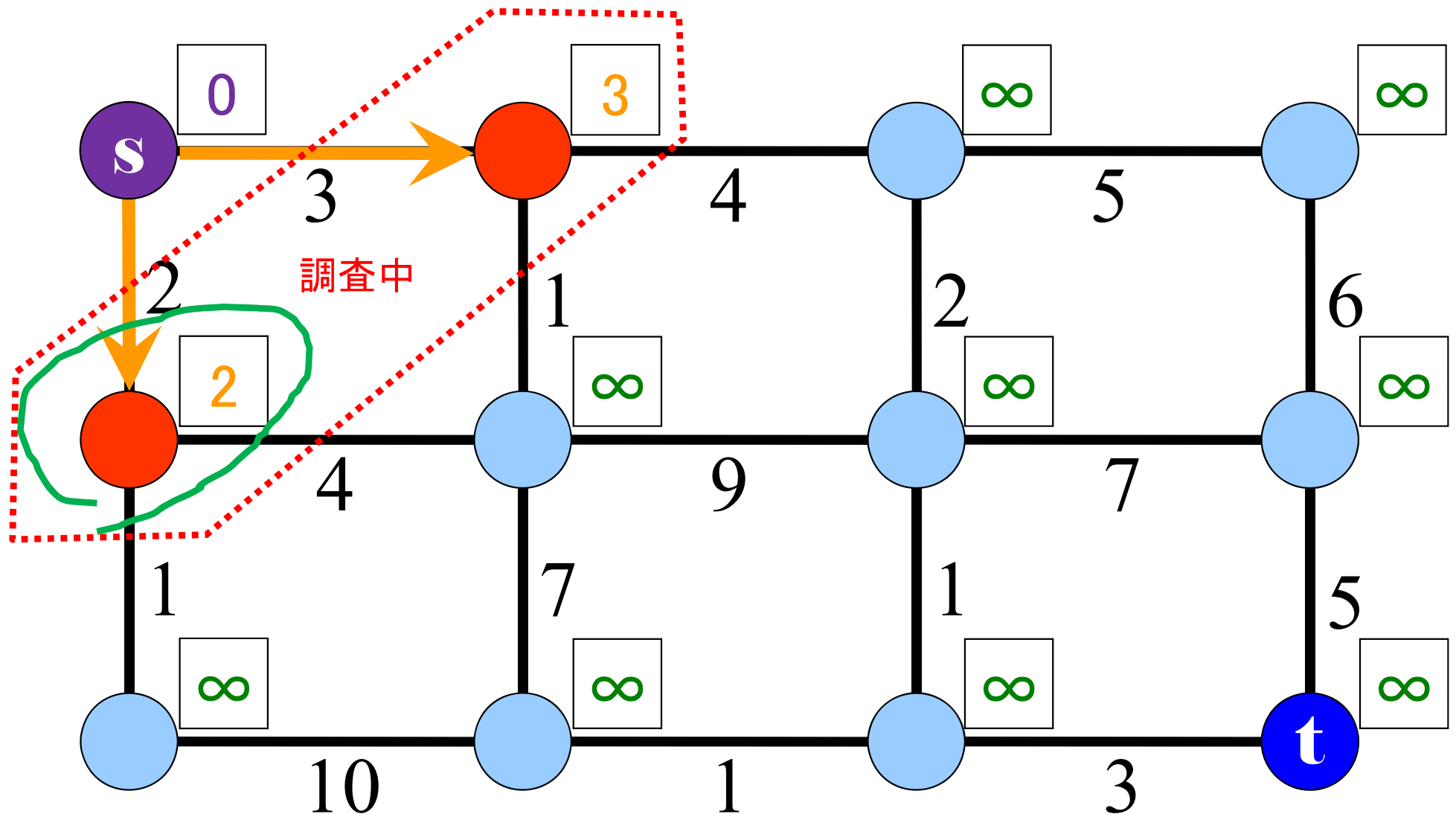
以降step1-1~step1-3を

終了条件を満たすまで繰り返す



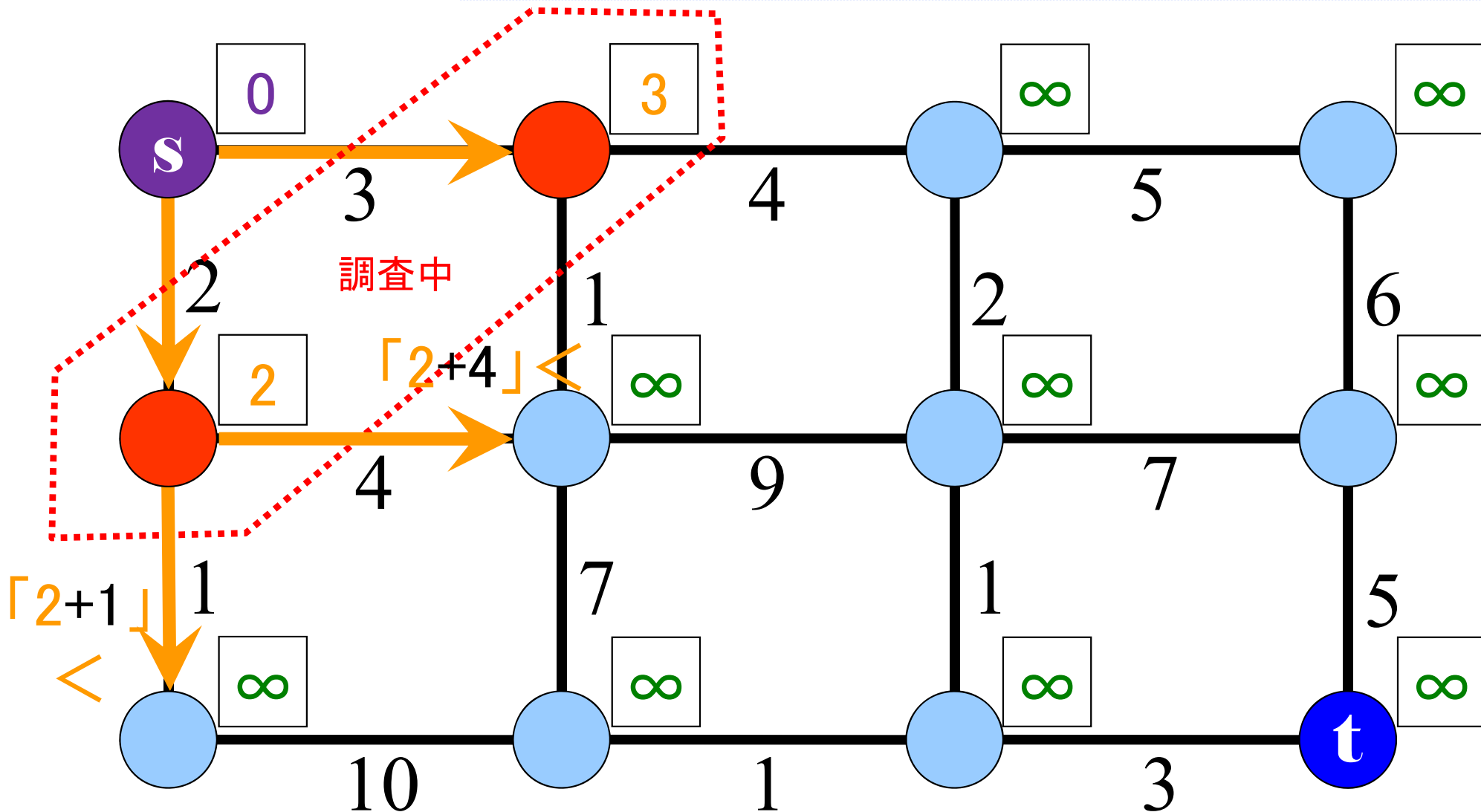
Dijkstra法 (更新法)

step1-1: 調査中点のうちで、ラベル値が最小の点を見つける



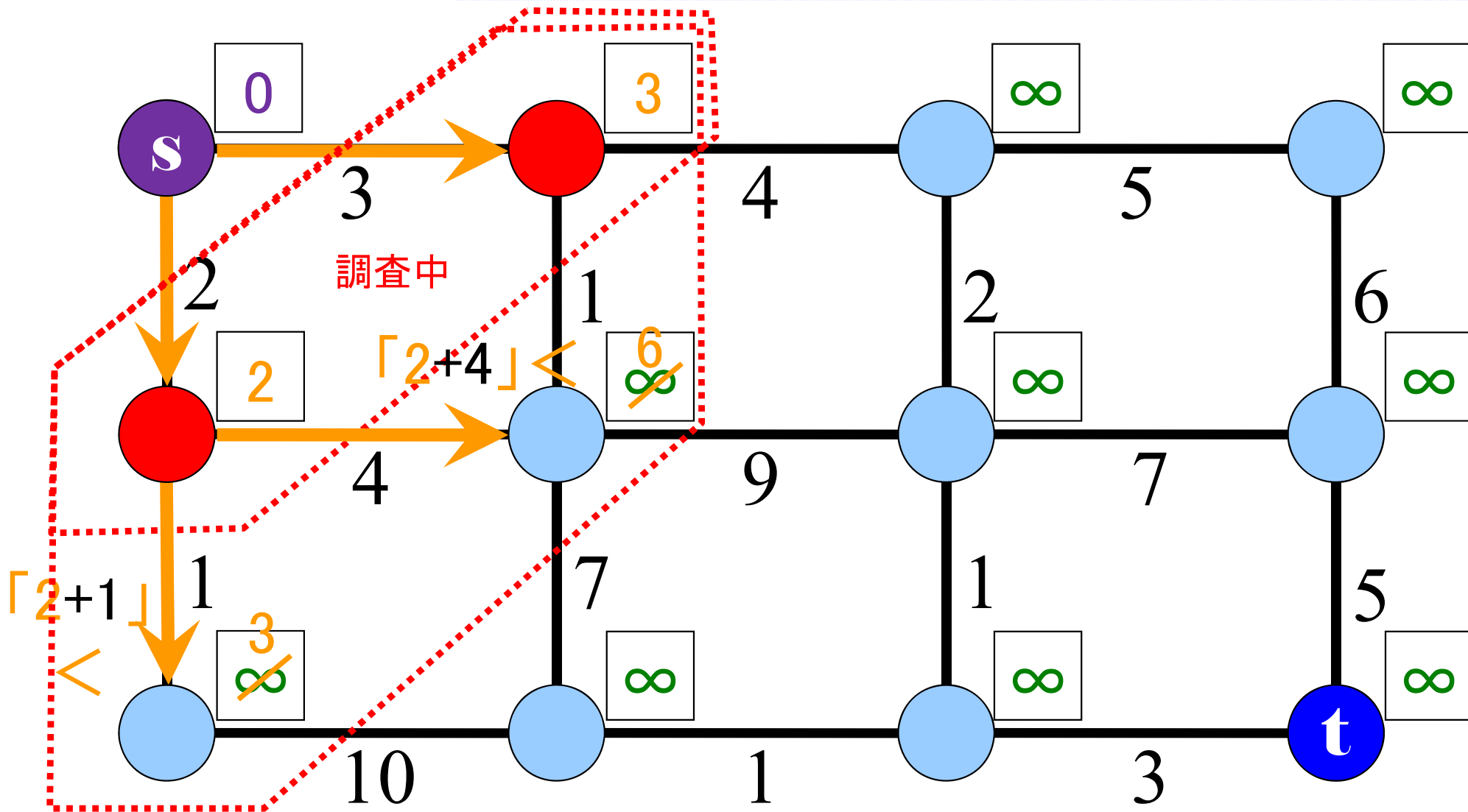
Dijkstra法 (更新法)

step1-2: その点から出る全枝について「ラベル+枝コスト」を計算し、枝先点のラベル値と比較し、もし、小さい($<$)→枝先点のラベル更新(暫定最短路)し、枝先点を調査中に追加, そうでないなら何もしない



Dijkstra法 (更新法)

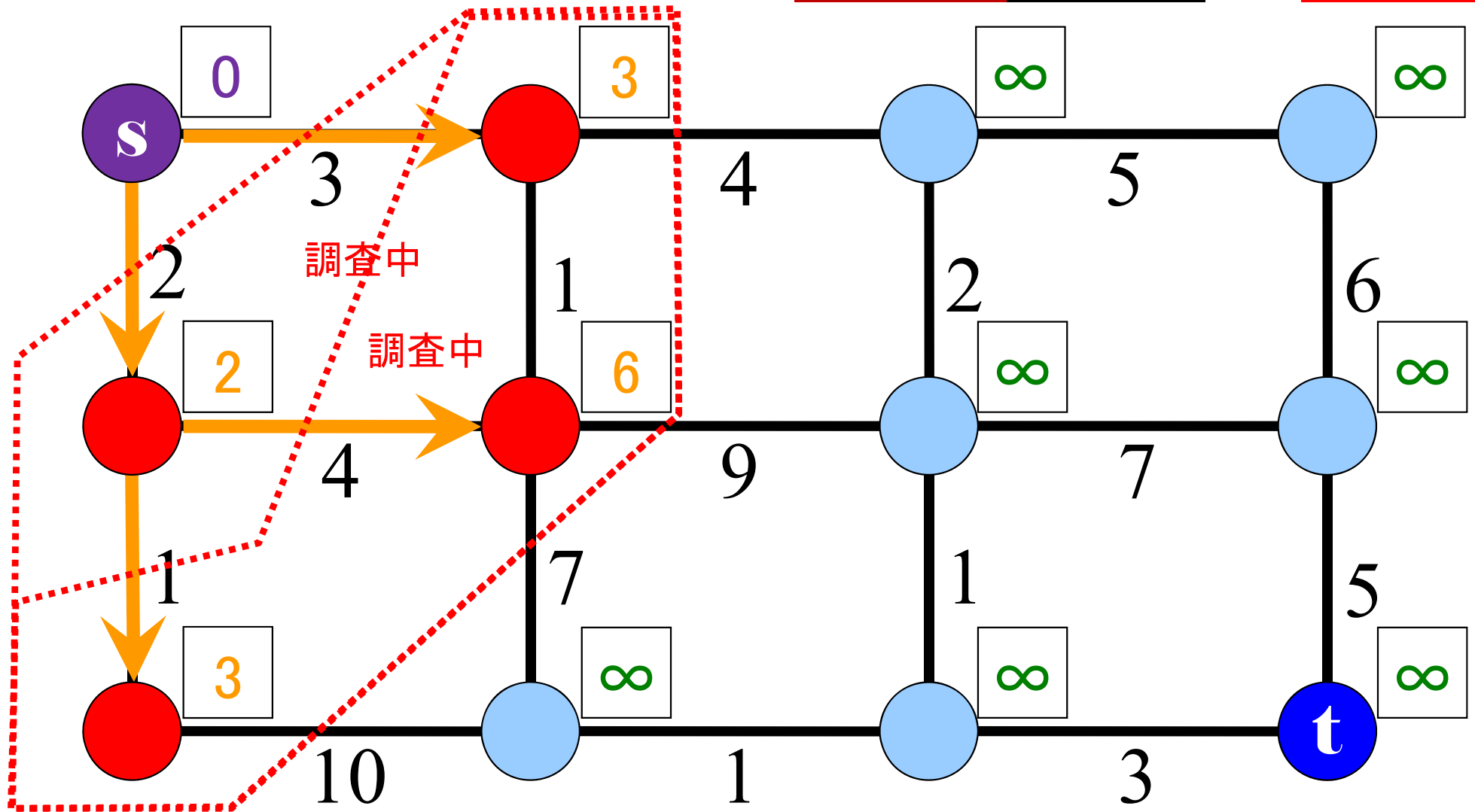
step1-2: その点から出る全枝について「ラベル+枝コスト」を計算し、枝先点のラベル値と比較し、もし、小さい($<$)→枝先点のラベル更新(暫定最短路)し、枝先点を調査中に追加, そうでないなら何もしない



Dijkstra法 (更新法)

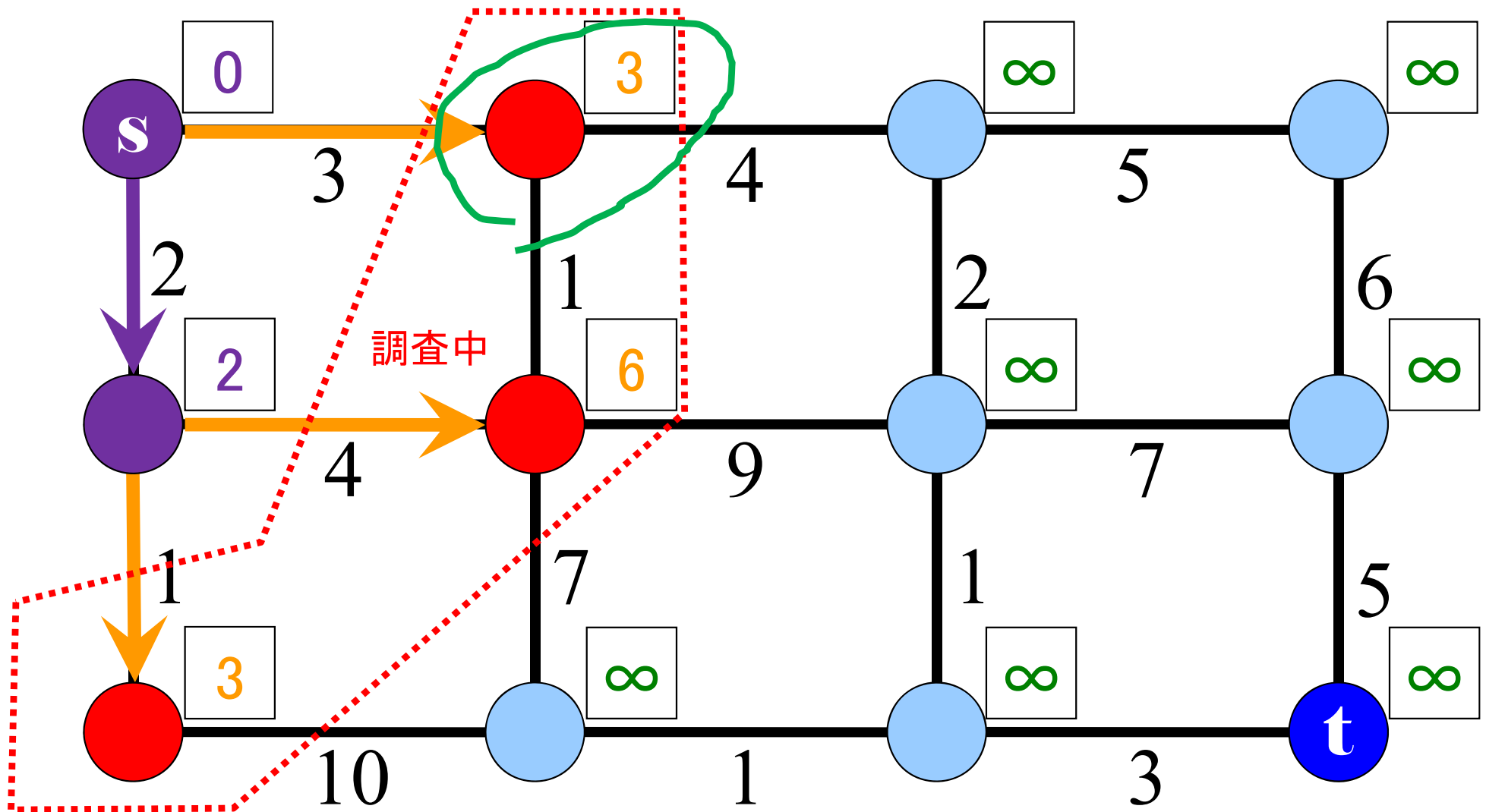
step1-3: 全枝終了したら, 調査中から外しラベル値確定
(その点までの暫定最短路が最短路として確定)

以降step1-1~step1-3を
終了条件を満たすまで繰り返す



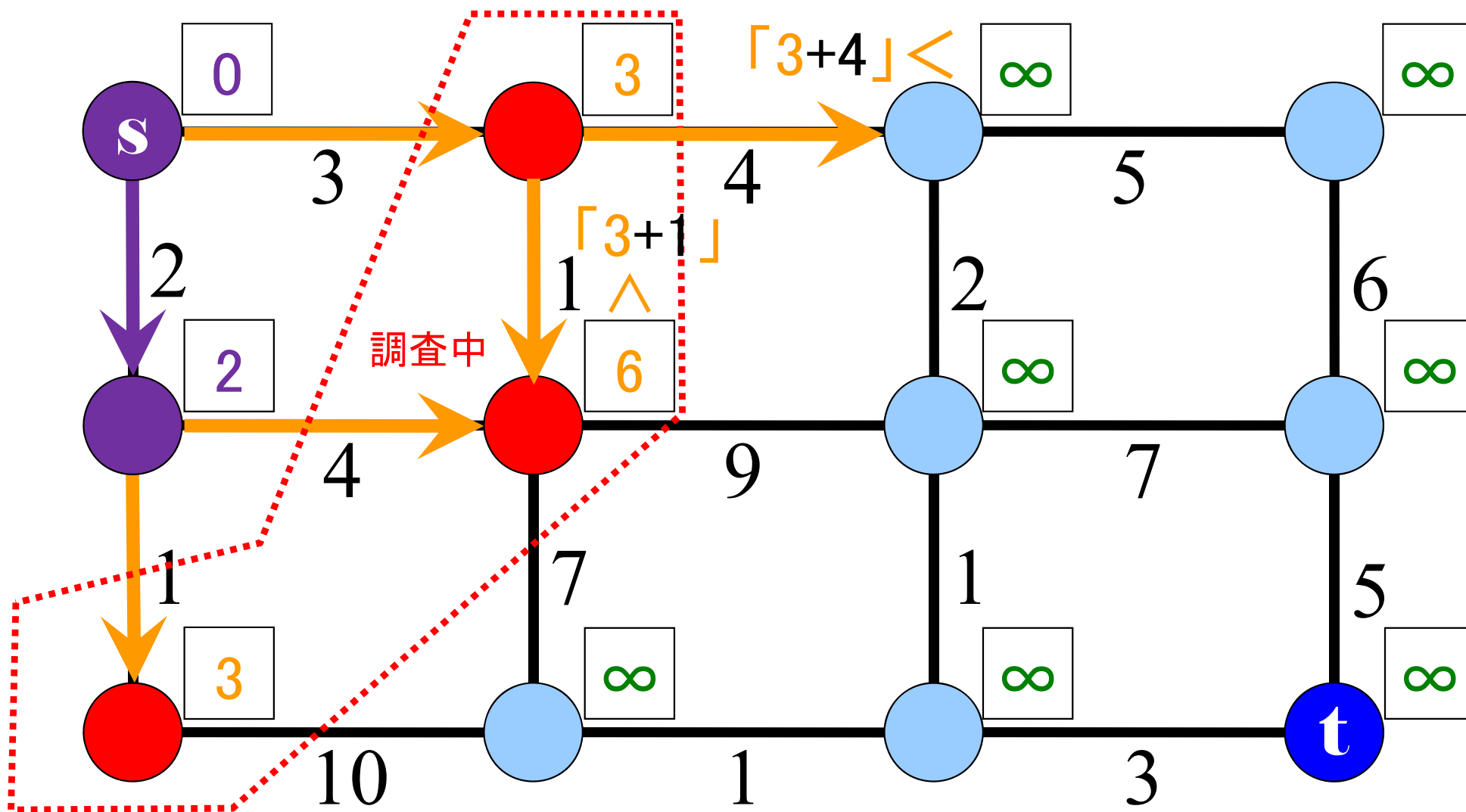
Dijkstra法 (更新法)

step1-1: 調査中点のうちで、ラベル値が最小の点を見つける



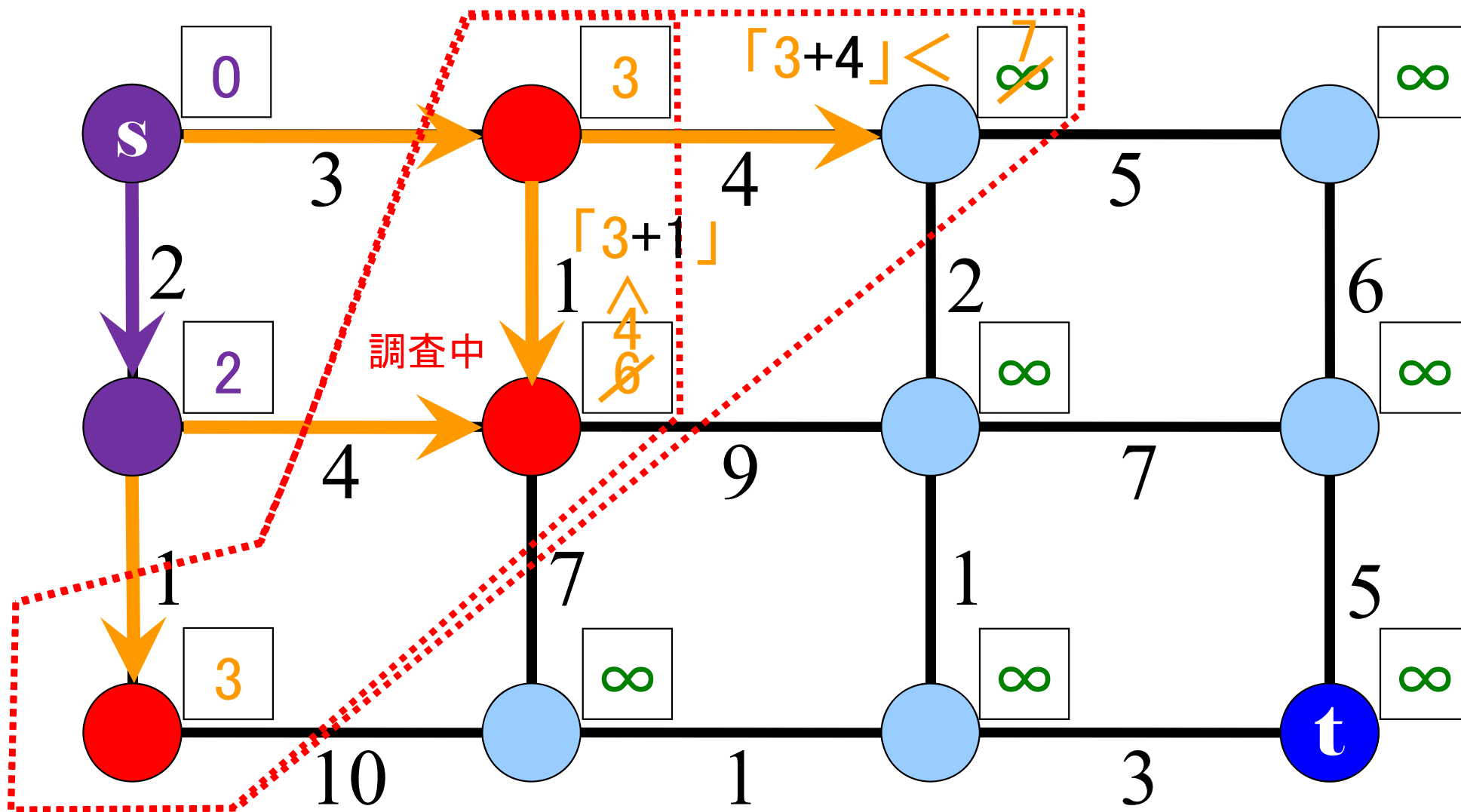
Dijkstra法 (更新法)

step1-2: その点から出る全枝について「ラベル+枝コスト」を計算し、枝先点のラベル値と比較し、もし、小さい($<$)→枝先点のラベル更新(暫定最短路)し、枝先点を調査中に追加, そうでないなら何もしない



Dijkstra法 (更新法)

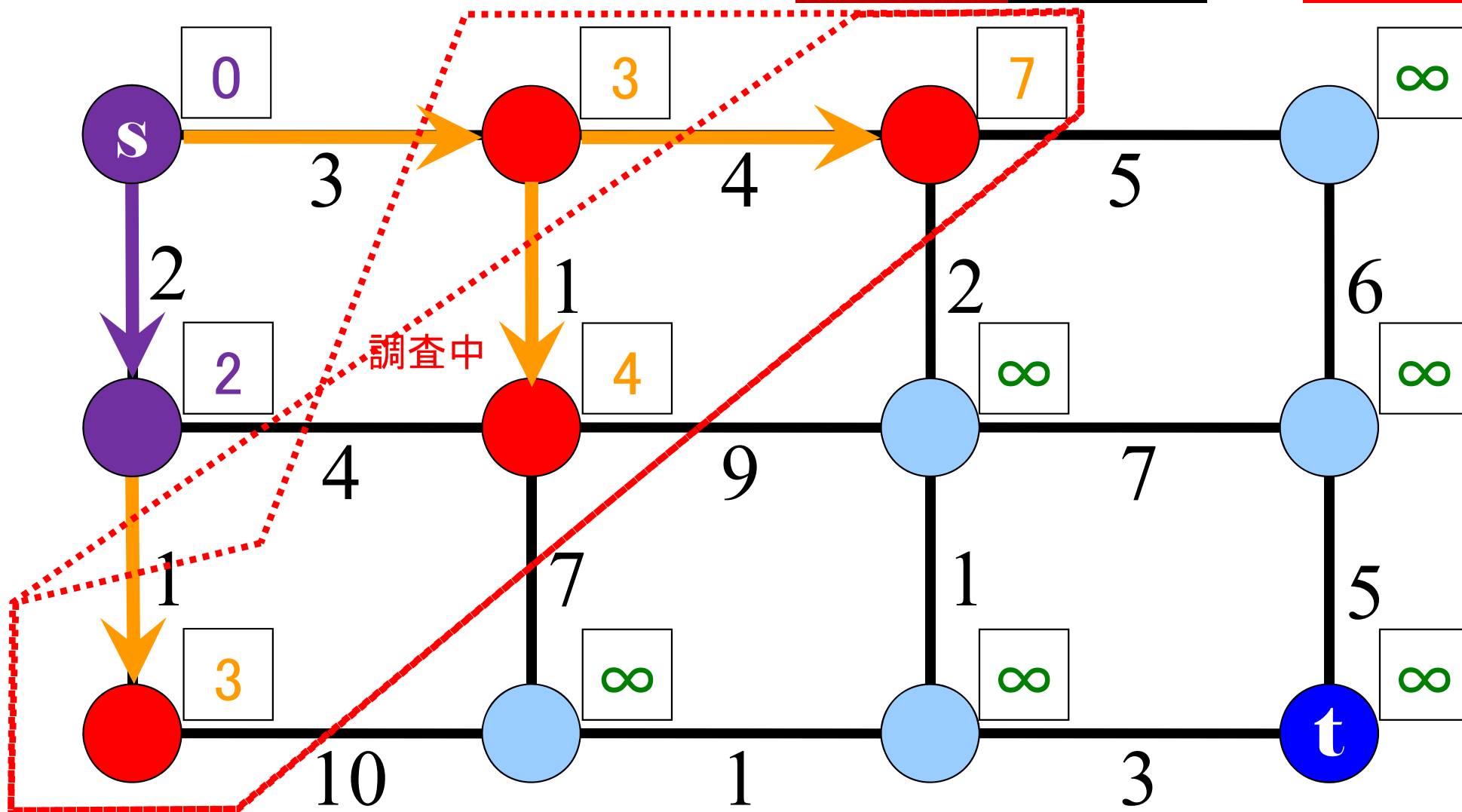
step1-2: その点から出る全枝について「ラベル+枝コスト」を計算し、枝先点のラベル値と比較し、もし、小さい($<$)→枝先点のラベル更新(暫定最短路)し、枝先点を調査中に追加, そうでないなら何もしない



Dijkstra法 (更新法)

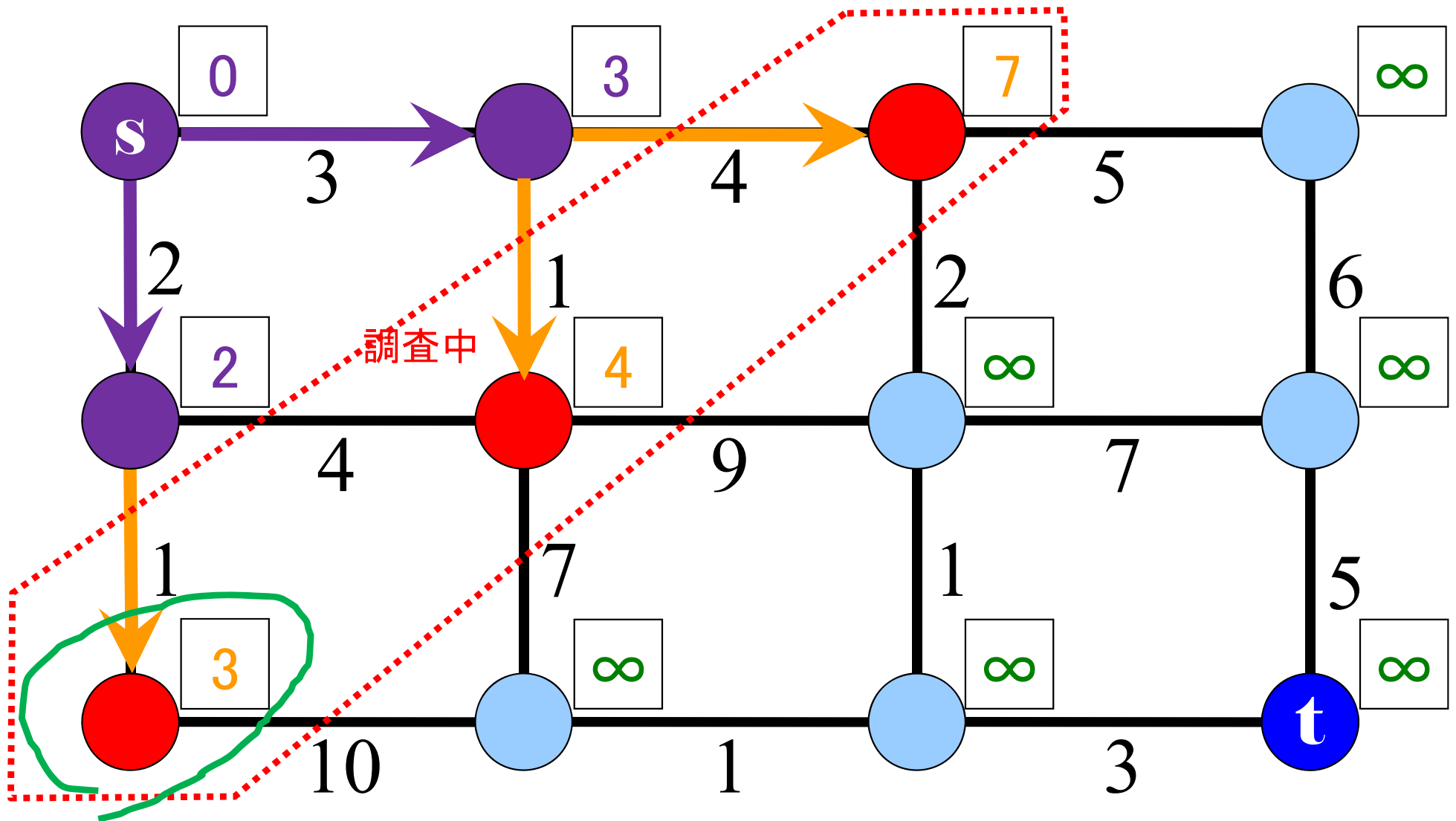
step1-3: 全枝終了したら, 調査中から外しラベル値確定
(その点までの暫定最短路が最短路として確定)

以降step1-1~step1-3を
終了条件を満たすまで繰り返す



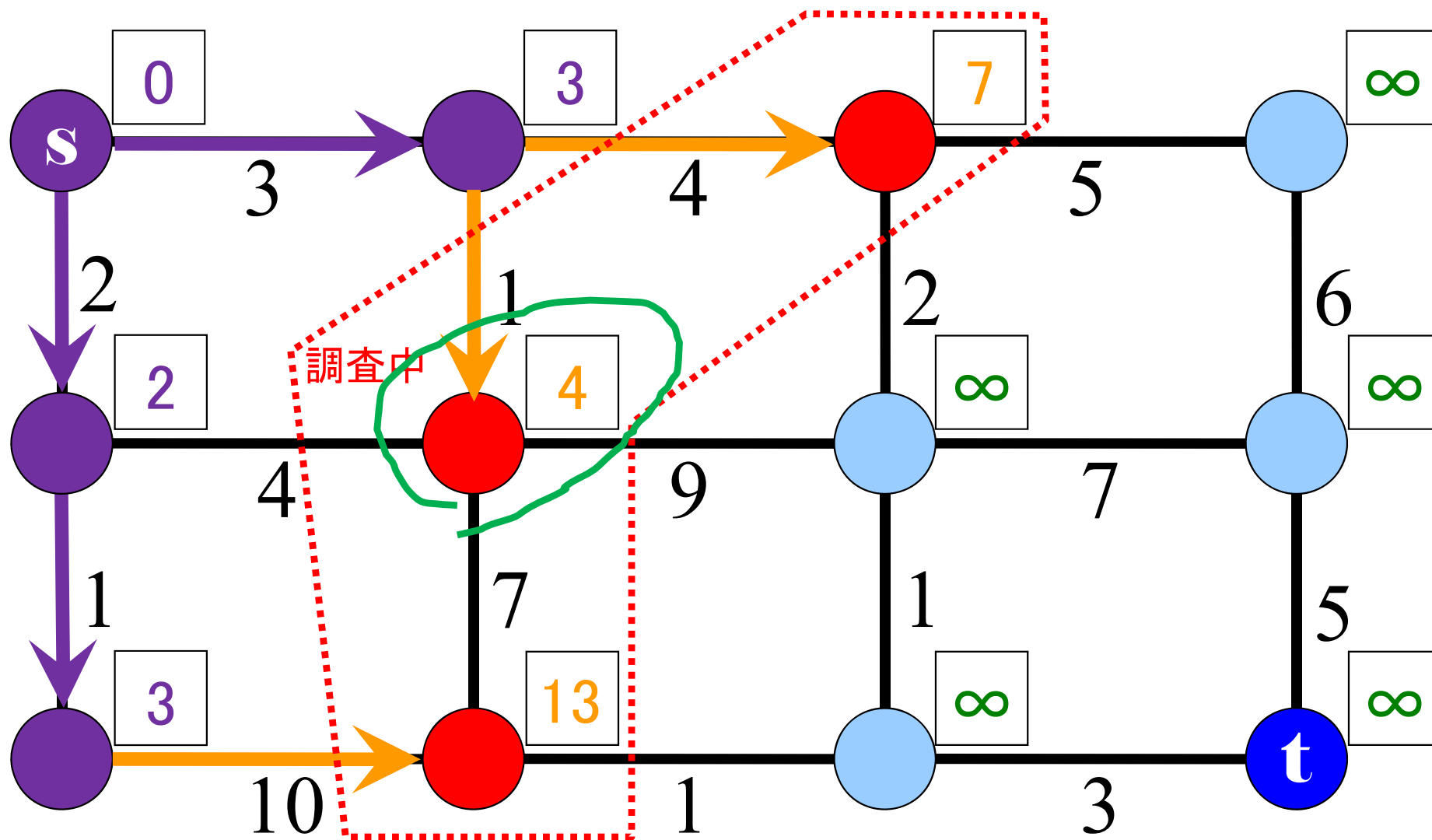
Dijkstra法 (更新法)

step1-1: 調査中点のうちで、ラベル値が最小の点を見つける



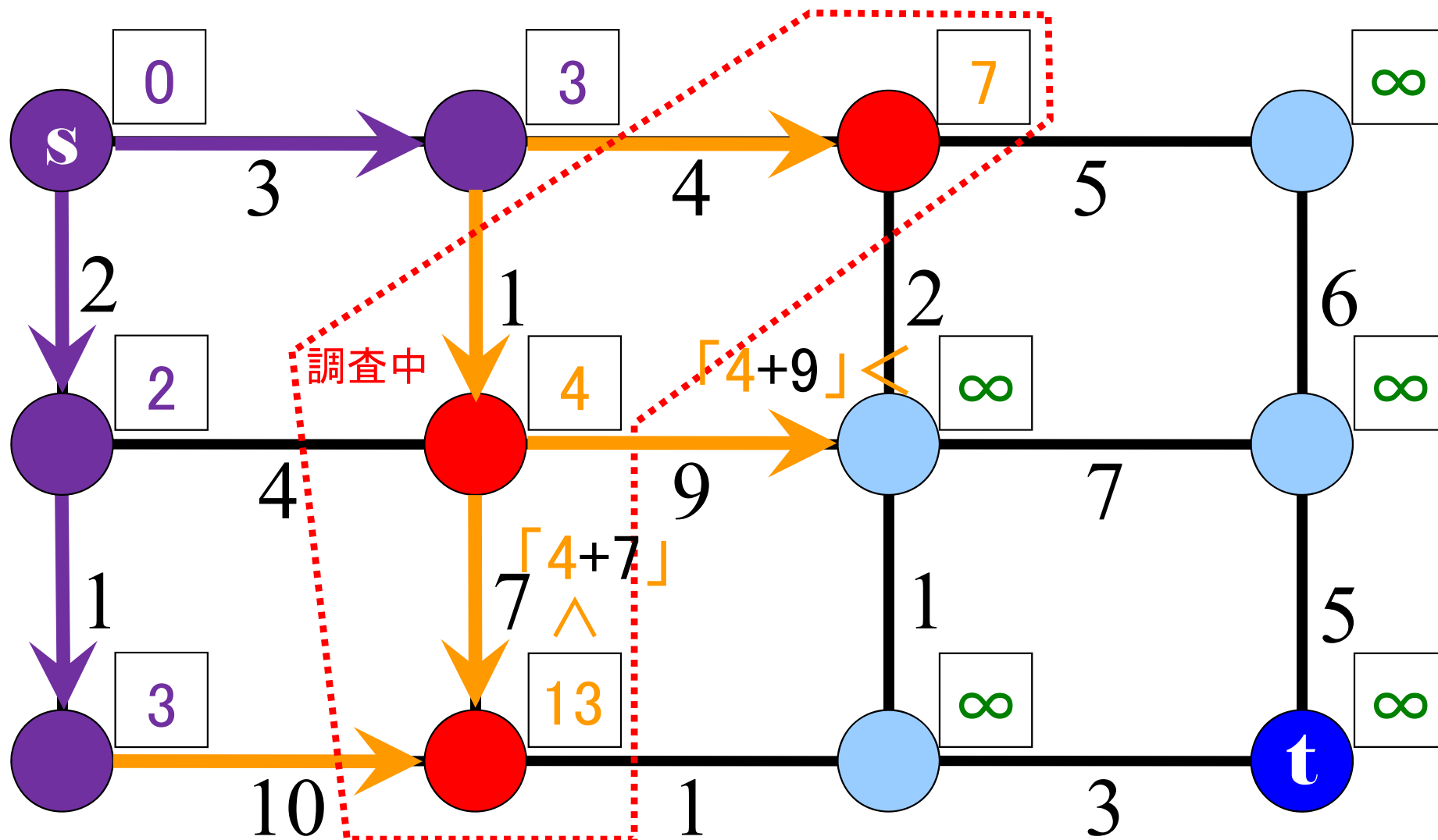
Dijkstra法 (更新法)

step1-1: 調査中点のうちで、ラベル値が最小の点を見つける



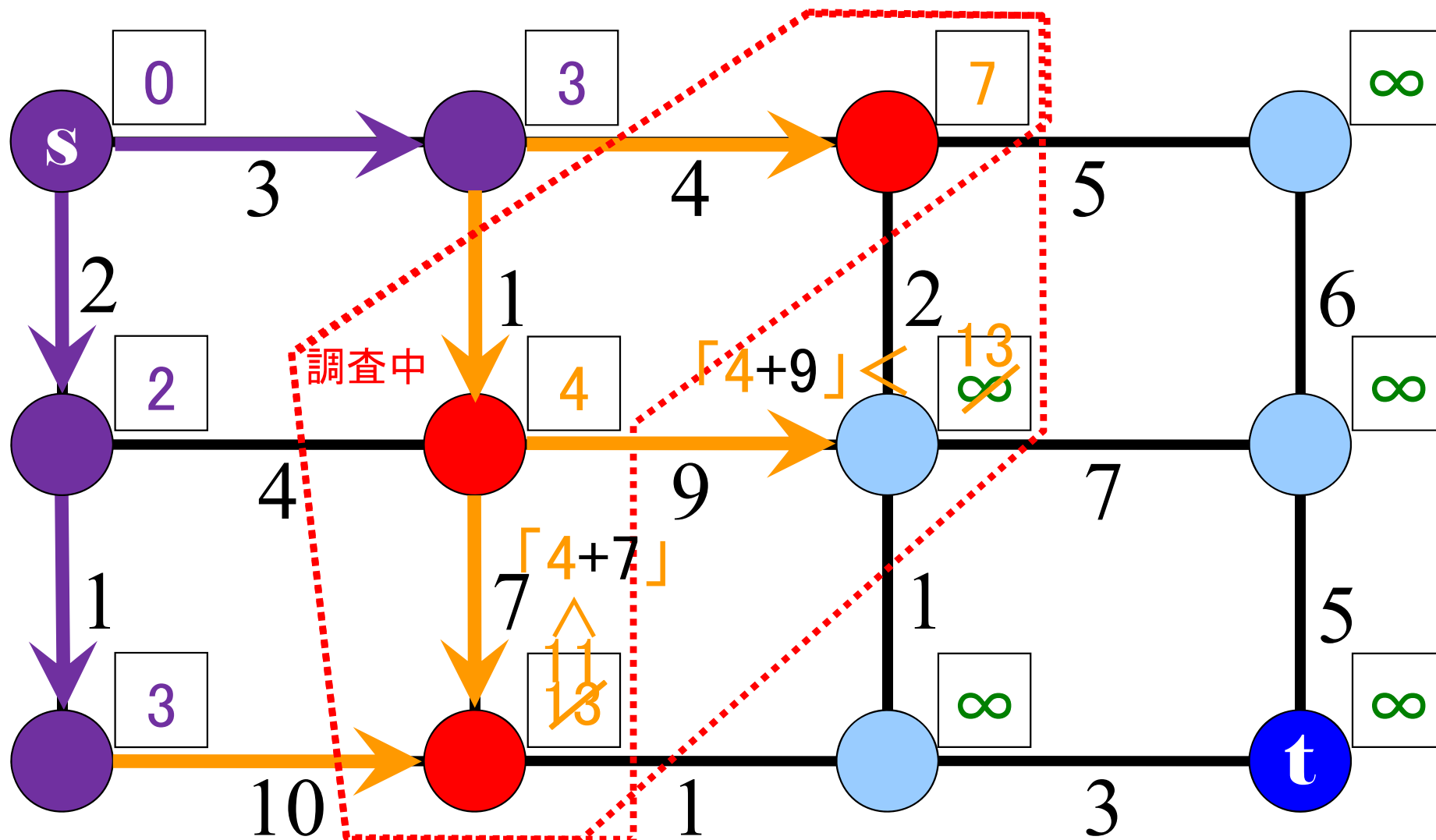
Dijkstra法 (更新法)

step1-2: その点から出る全枝について「ラベル+枝コスト」を計算し、枝先点のラベル値と比較し、もし、小さい($<$)→枝先点のラベル更新(暫定最短路)し、枝先点を調査中に追加, そうでないなら何もしない



Dijkstra法 (更新法)

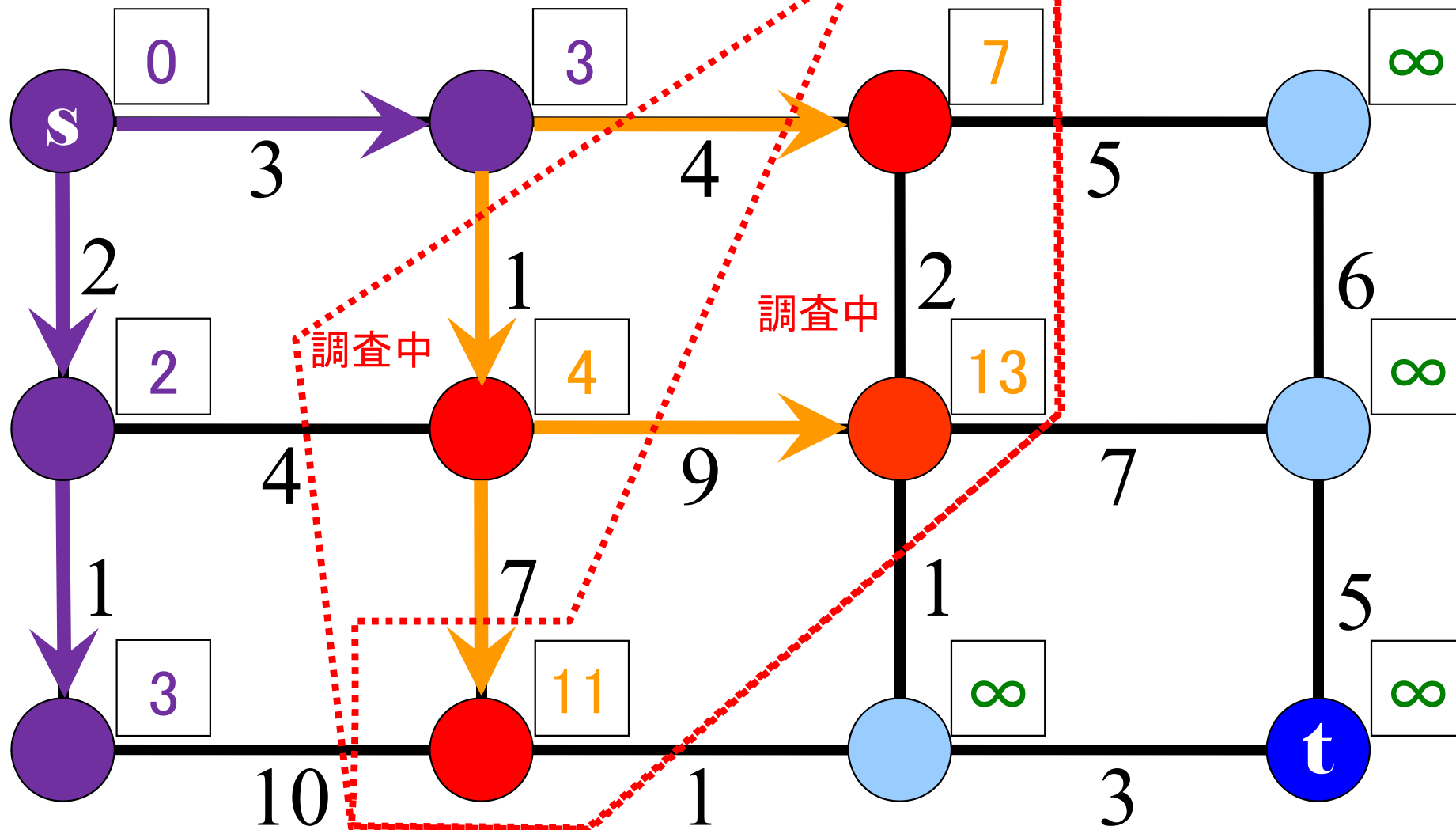
step1-2: その点から出る全枝について「ラベル+枝コスト」を計算し、枝先点のラベル値と比較し、もし、小さい($<$)→枝先点のラベル更新(暫定最短路)し、枝先点を調査中に追加, そうでないなら何もしない



Dijkstra法 (更新法)

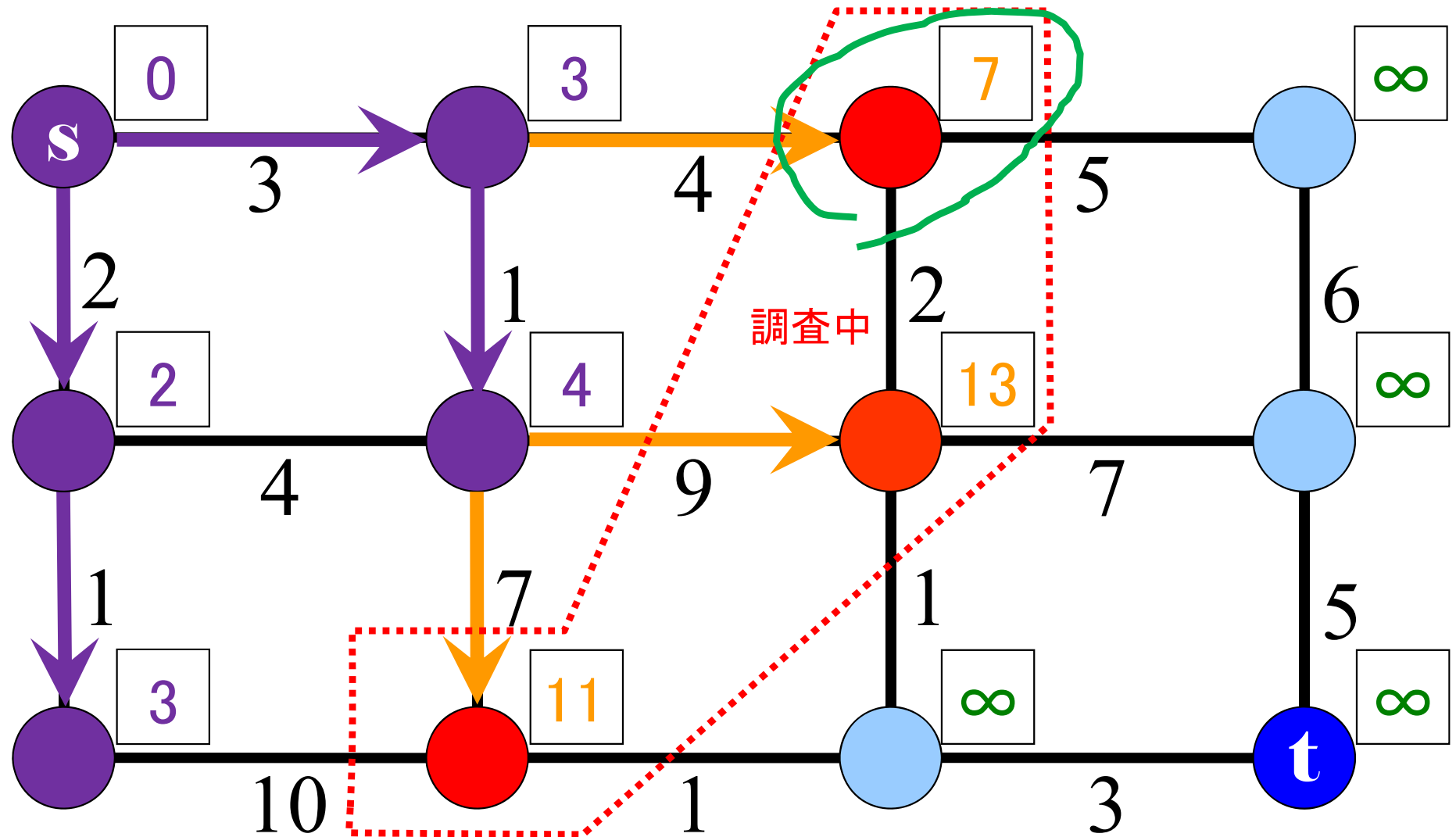
step1-3: 全枝終了したら, 調査中から外しラベル値確定
(その点までの暫定最短路が最短路として確定)

以降step1-1~step1-3を
終了条件を満たすまで繰り返す



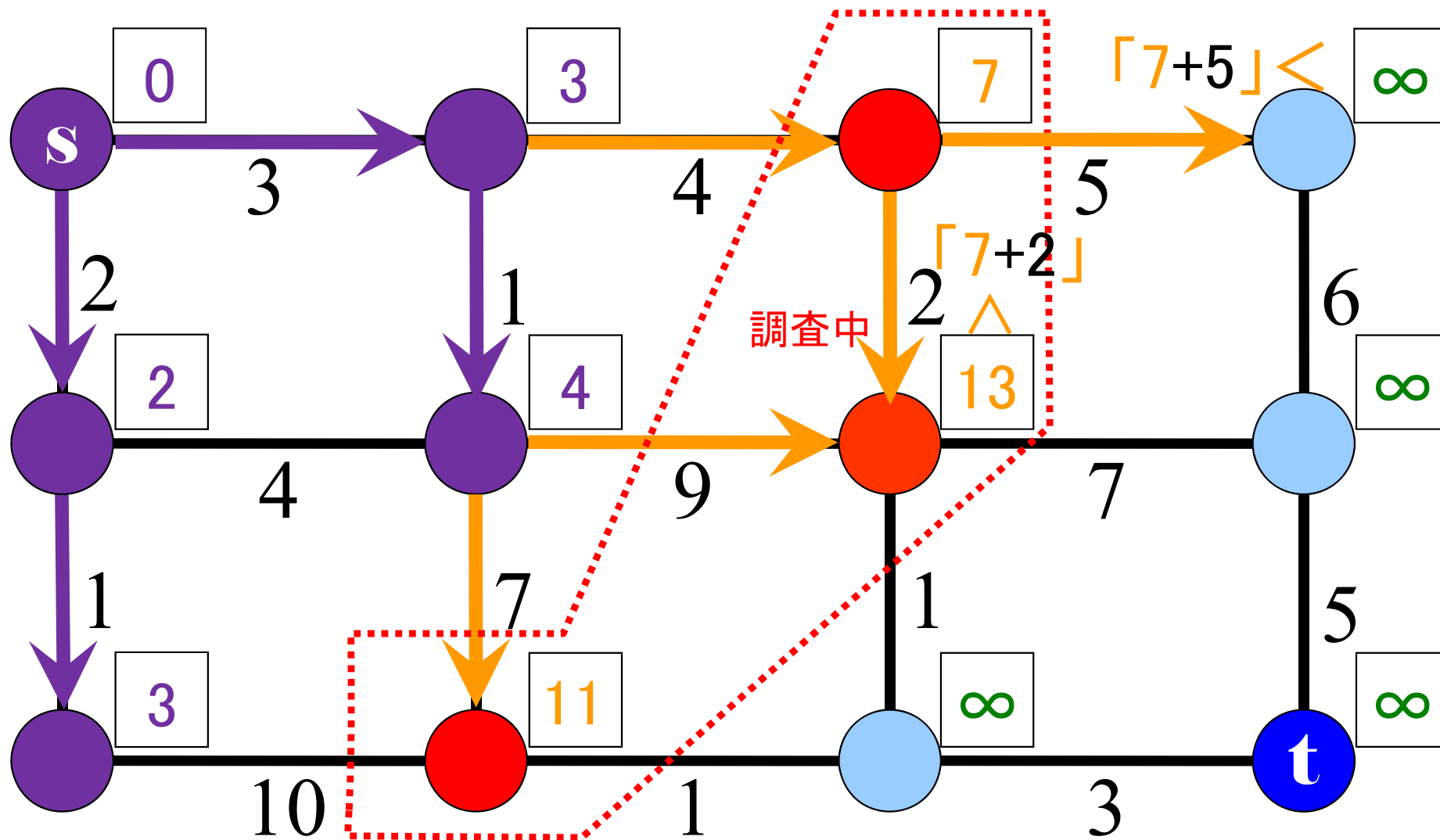
Dijkstra法 (更新法)

step1-1: 調査中点のうちで、ラベル値が最小の点を見つける



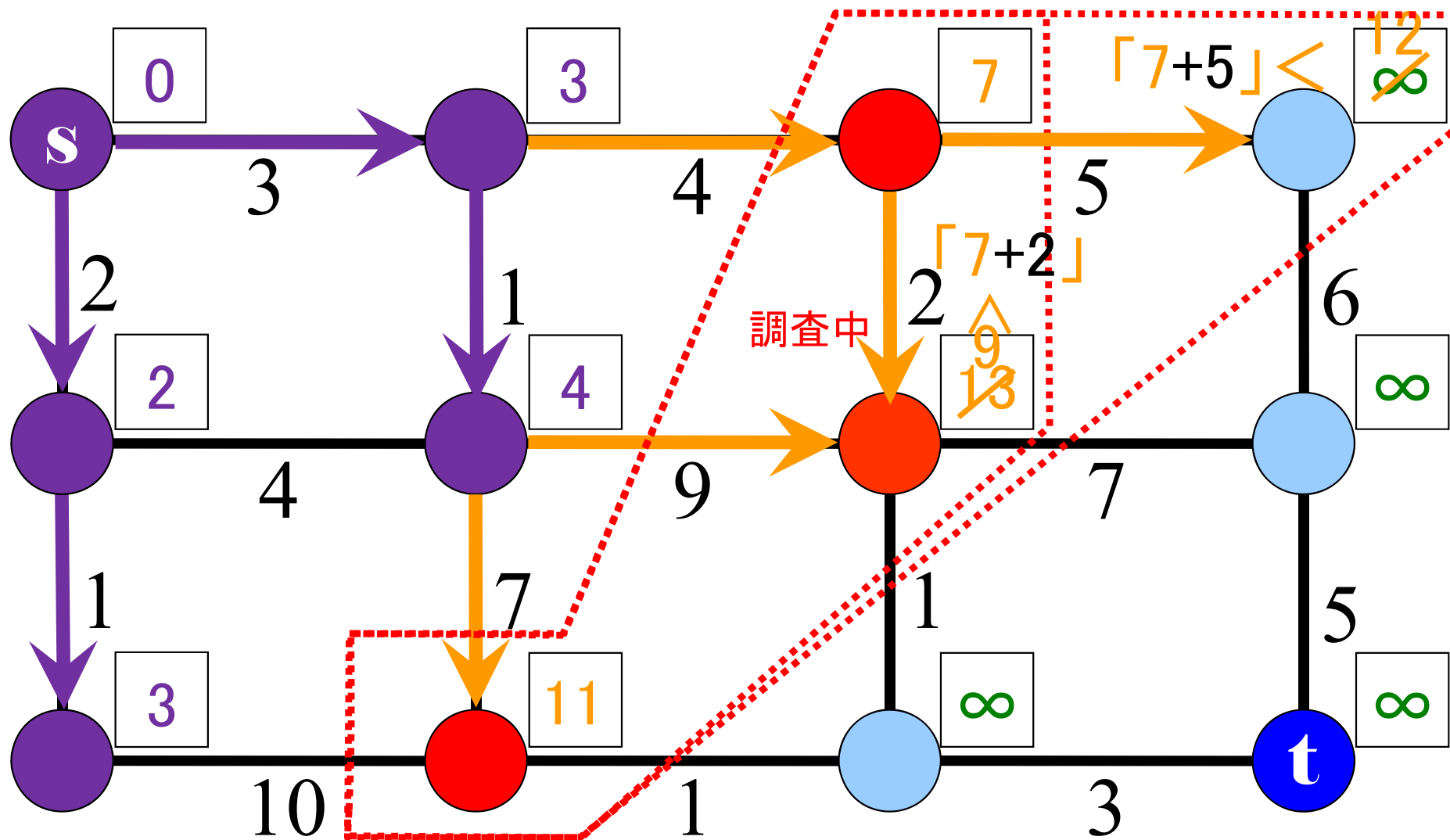
Dijkstra法 (更新法)

step1-2: その点から出る全枝について「ラベル+枝コスト」を計算し、枝先点のラベル値と比較し、もし、小さい($<$)→枝先点のラベル更新(暫定最短路)し、枝先点を調査中に追加, そうでないなら何もしない



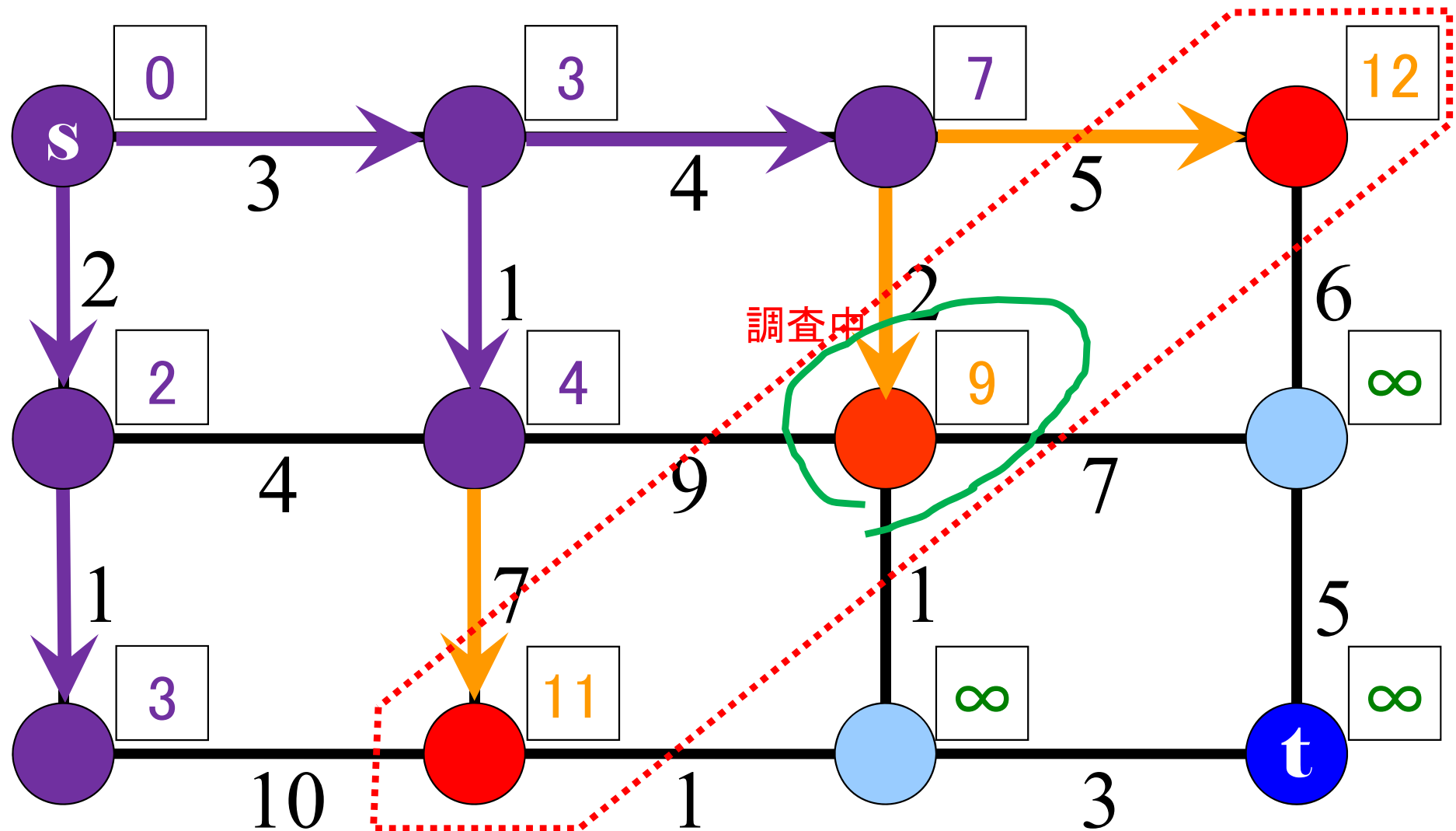
Dijkstra法 (更新法)

step1-2: その点から出る全枝について「ラベル+枝コスト」を計算し、枝先点のラベル値と比較し、もし、小さい($<$)→枝先点のラベル更新(暫定最短路)し、枝先点を調査中に追加, そうでないなら何もしない



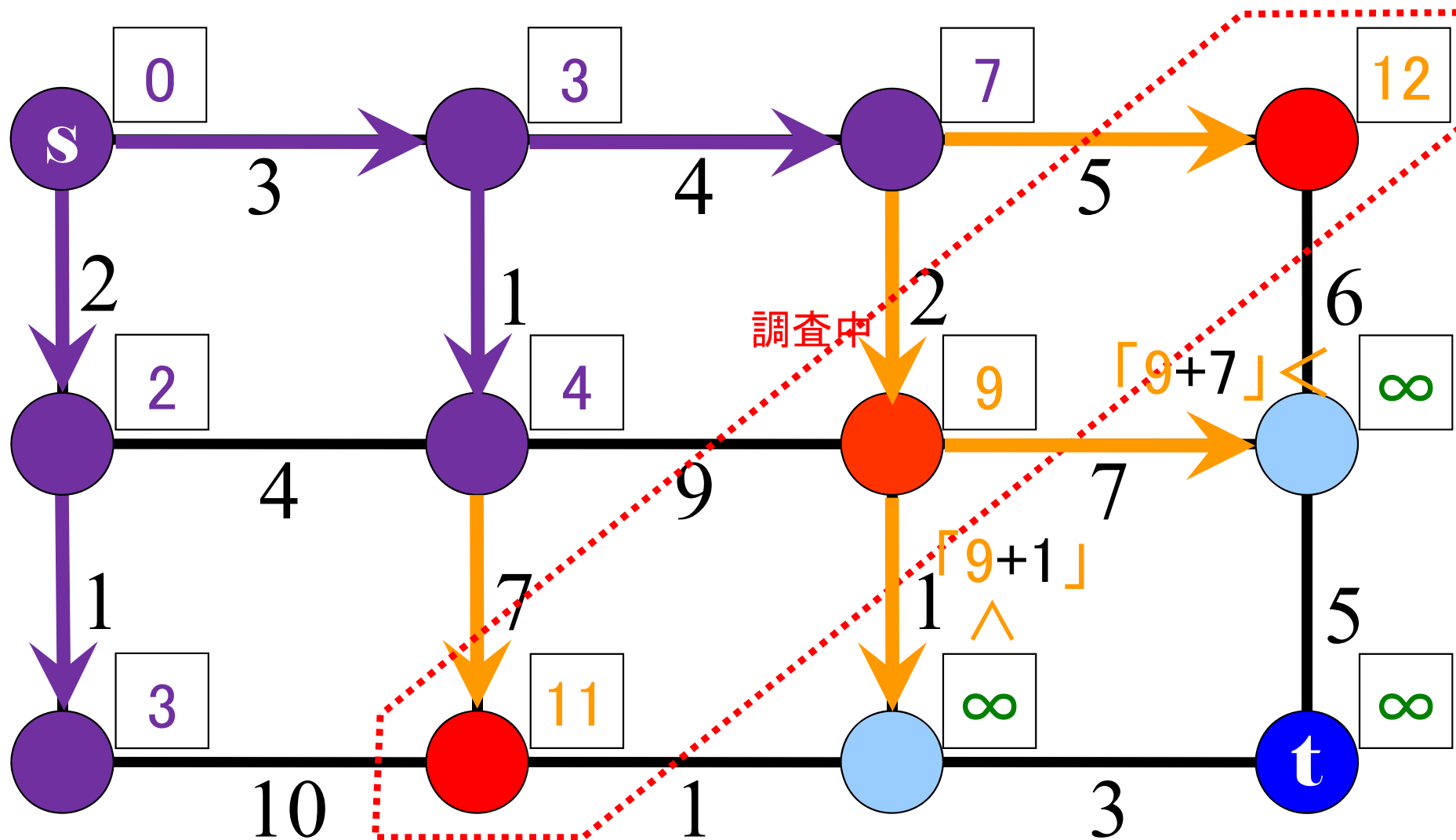
Dijkstra法 (更新法)

step1-1: 調査中点のうちで、ラベル値が最小の点を見つける



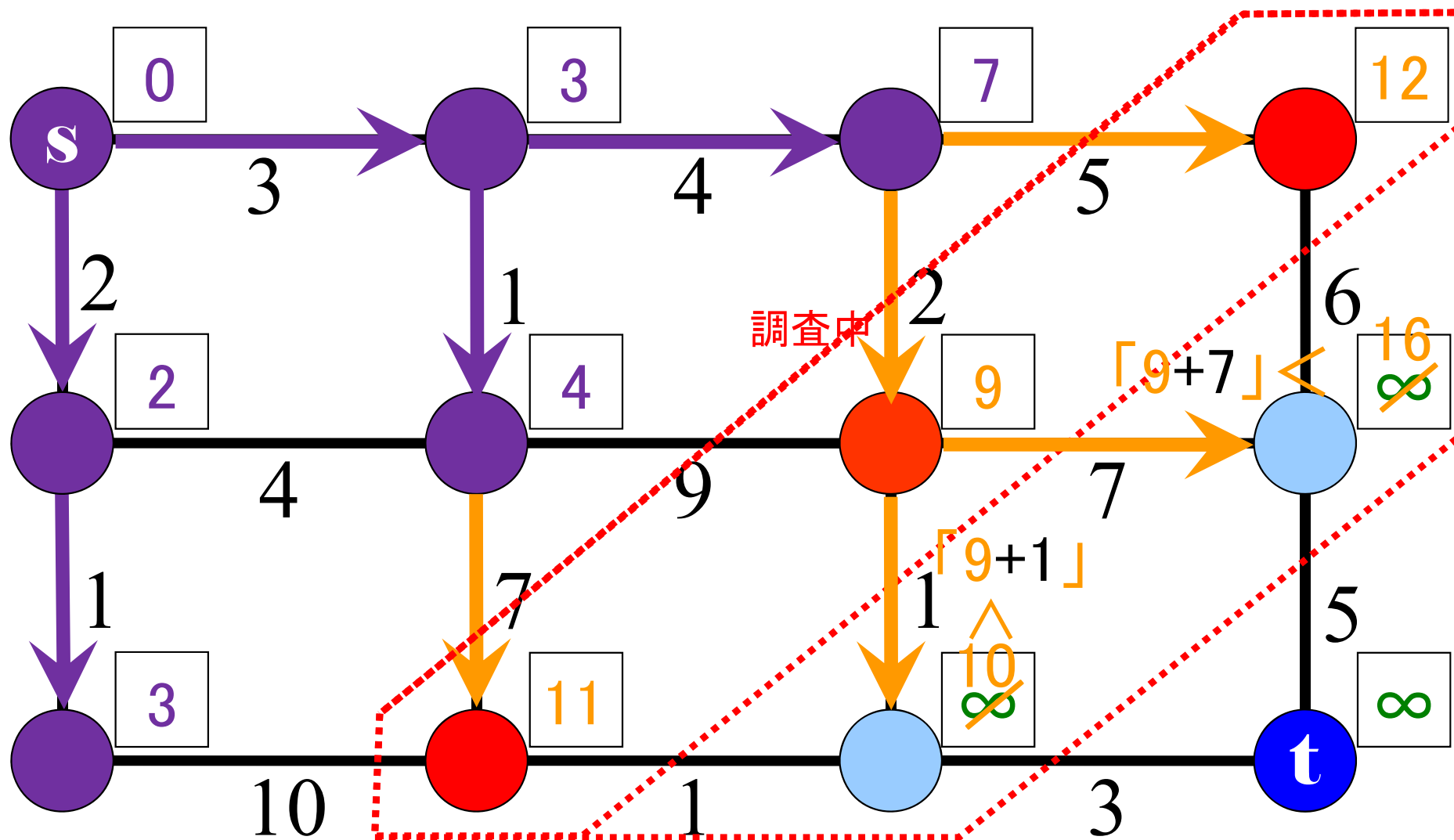
Dijkstra法 (更新法)

step1-2: その点から出る全枝について「ラベル+枝コスト」を計算し、枝先点のラベル値と比較し、もし、小さい($<$)→枝先点のラベル更新(暫定最短路)し、枝先点を調査中に追加, そうでないなら何もしない



Dijkstra法 (更新法)

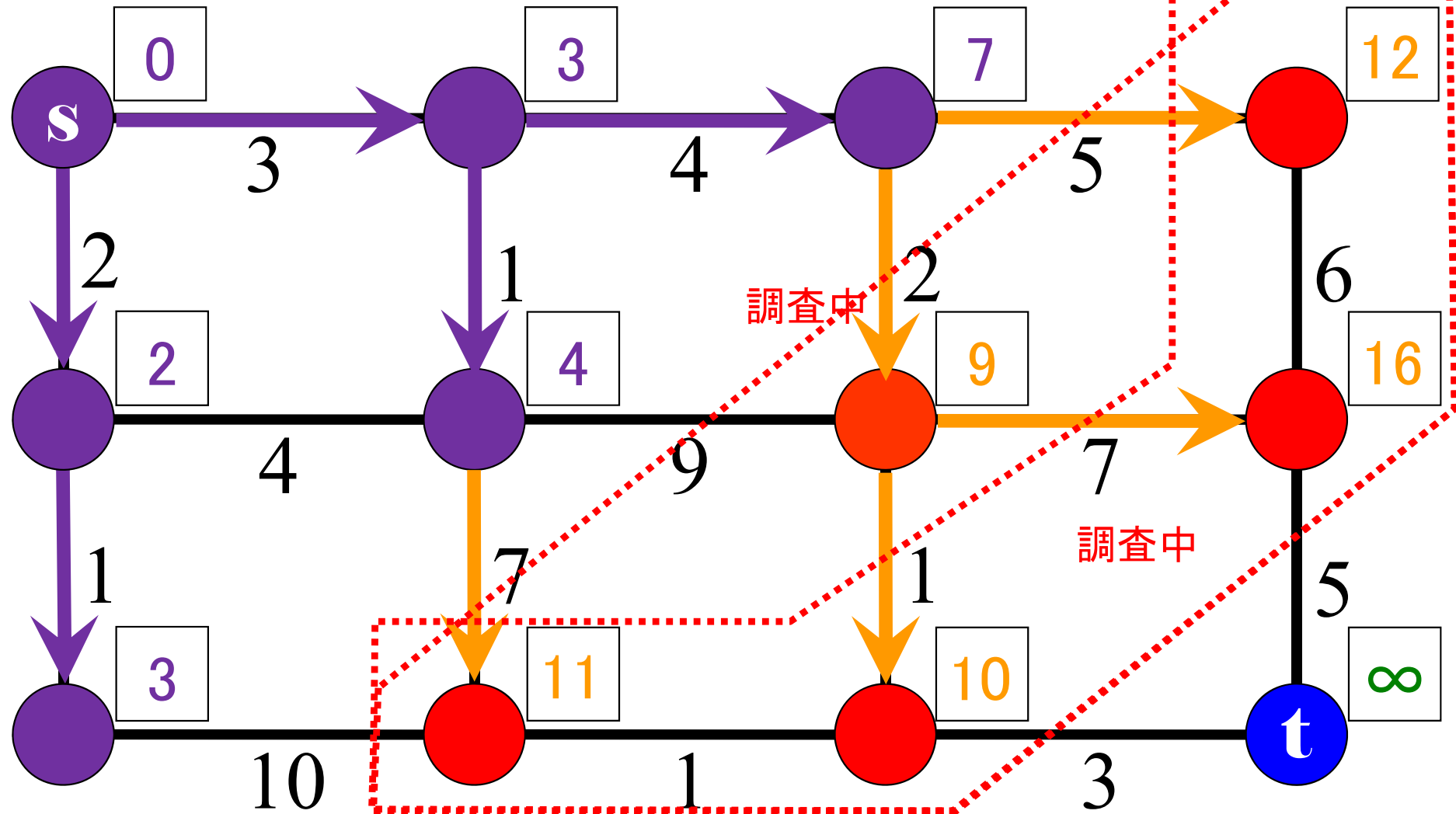
step1-2: その点から出る全枝について「ラベル+枝コスト」を計算し、枝先点のラベル値と比較し、もし、小さい($<$)→枝先点のラベル更新(暫定最短路)し、枝先点を調査中に追加, そうでないなら何もしない



Dijkstra法 (更新法)

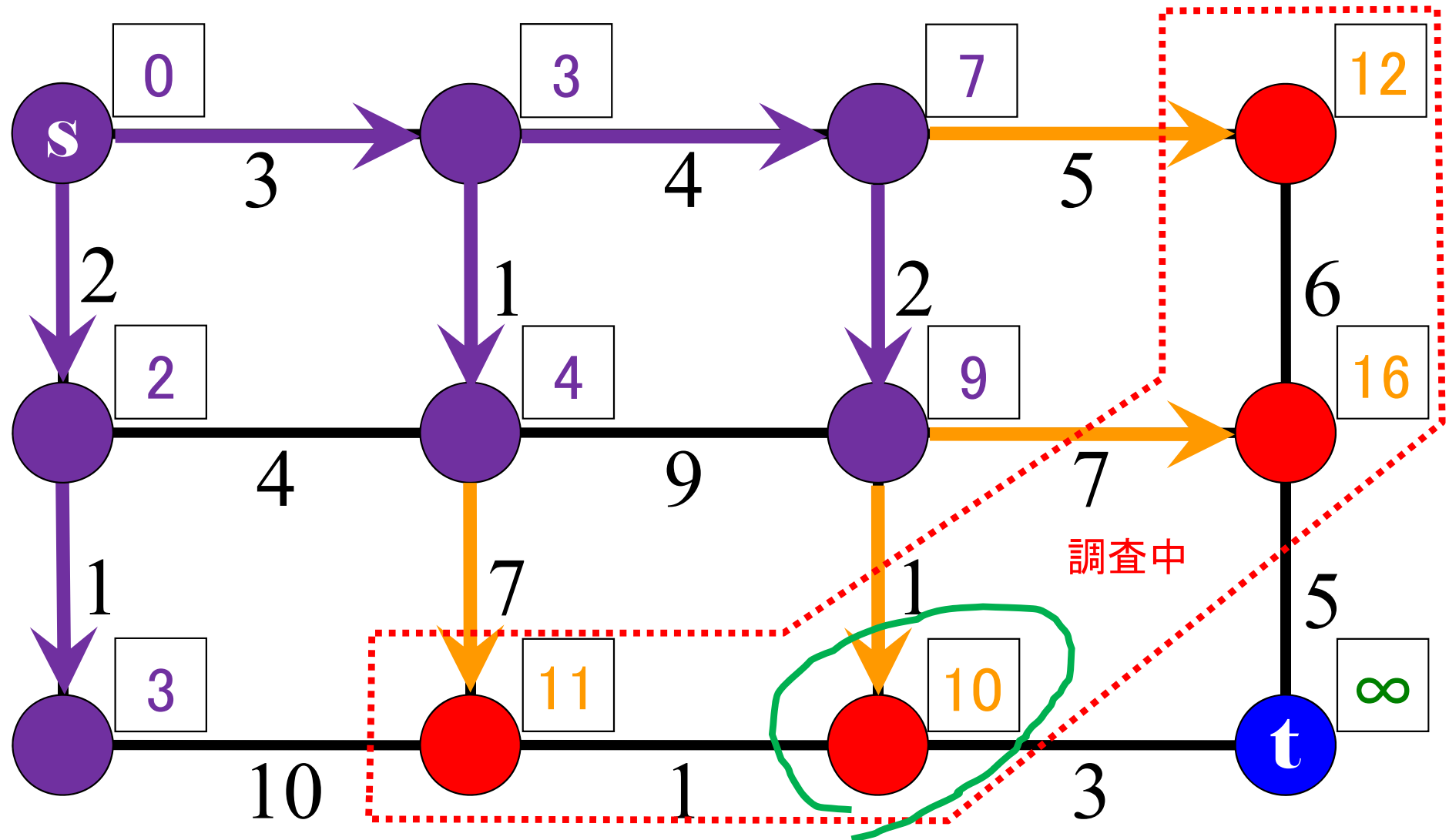
step1-3: 全枝終了したら, 調査中から外しラベル値確定
(その点までの暫定最短路が最短路として確定)

以降step1-1~step1-3を
終了条件を満たすまで繰り返す



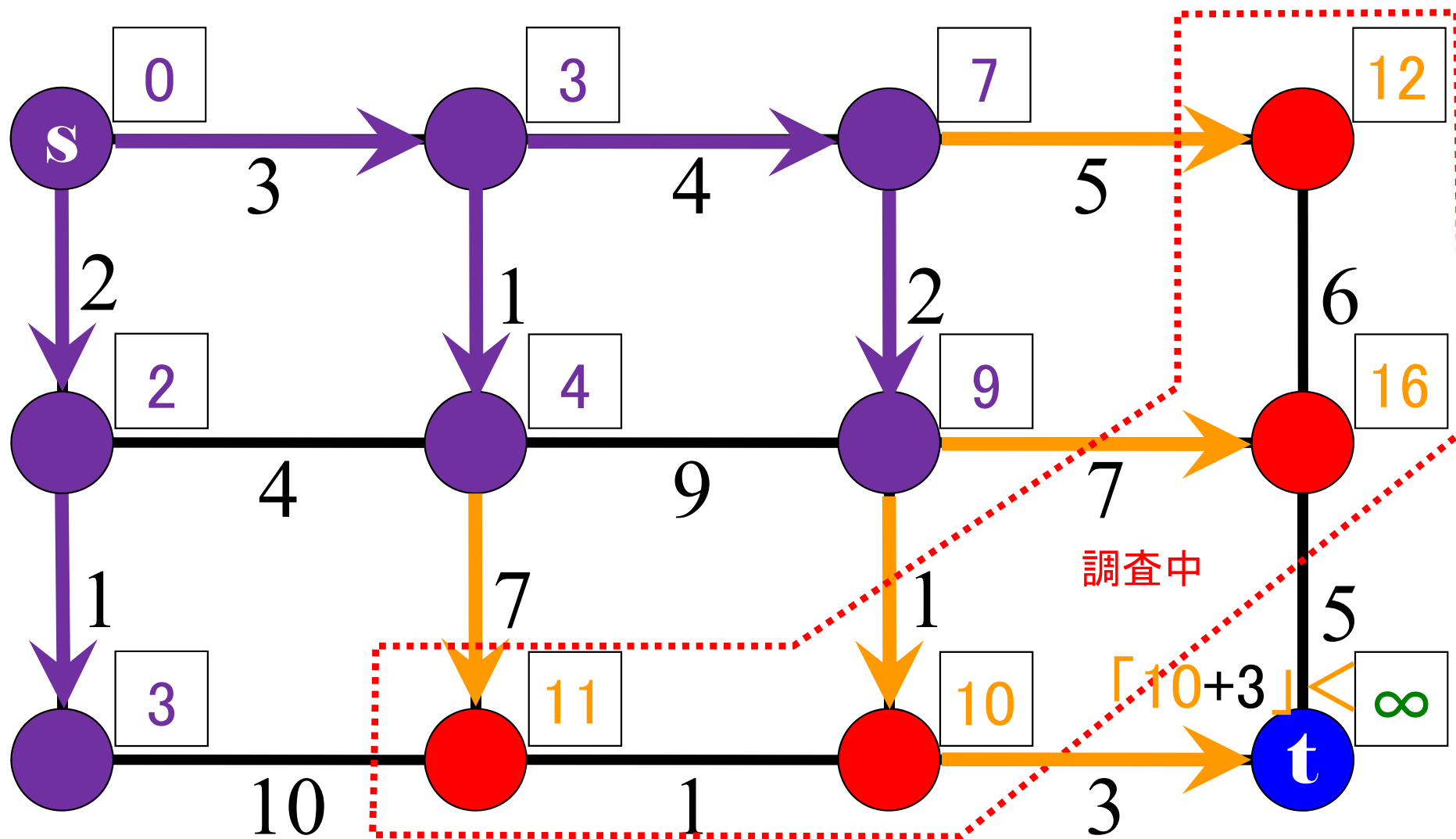
Dijkstra法 (更新法)

step1-1: 調査中点のうちで、ラベル値が最小の点を見つける



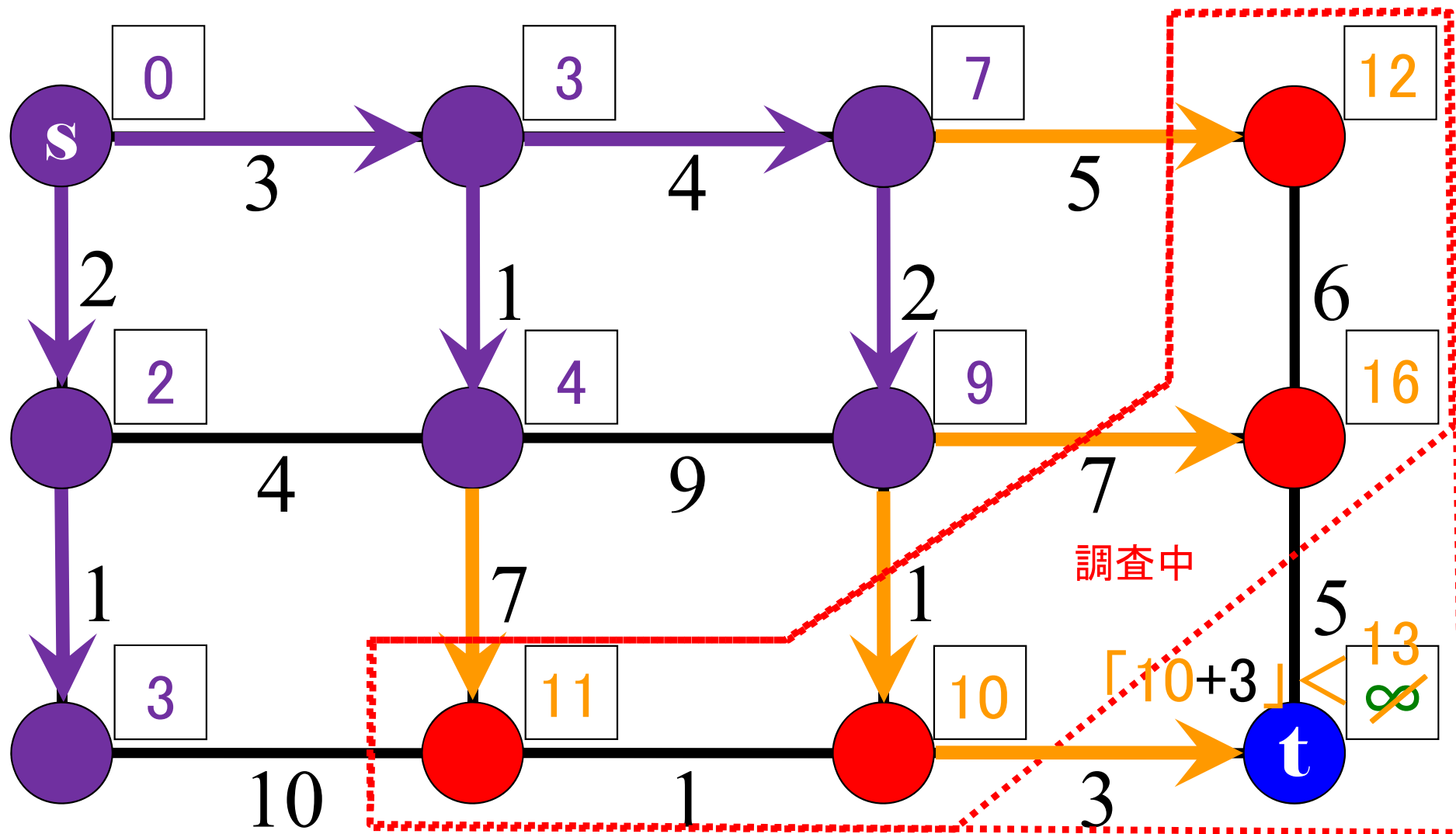
Dijkstra法 (更新法)

step1-2: その点から出る全枝について「ラベル+枝コスト」を計算し、枝先点のラベル値と比較し、もし、小さい($<$)→枝先点のラベル更新(暫定最短路)し、枝先点を調査中に追加, そうでないなら何もしない



Dijkstra法 (更新法)

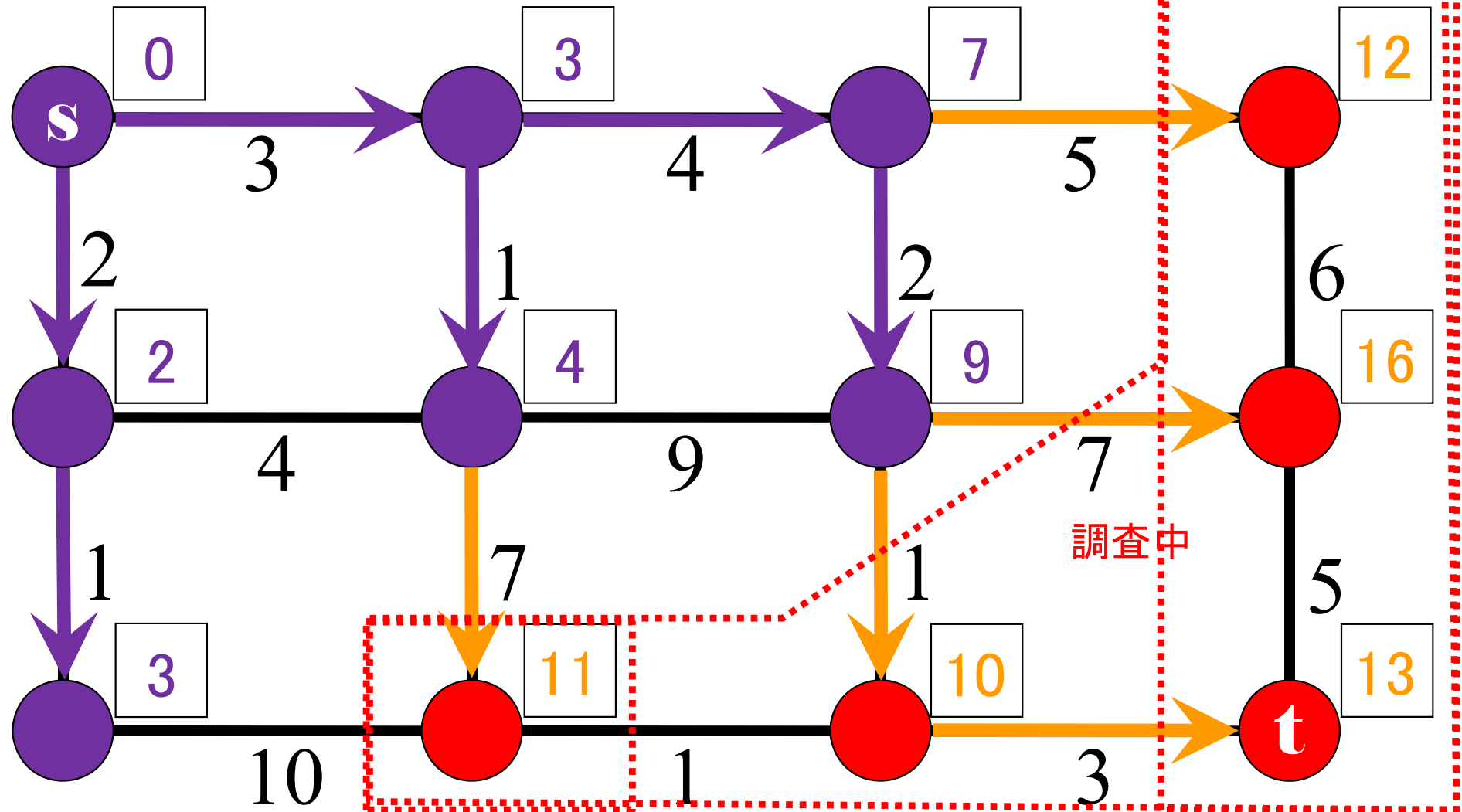
step1-2: その点から出る全枝について「ラベル+枝コスト」を計算し、枝先点のラベル値と比較し、もし、小さい($<$)→枝先点のラベル更新(暫定最短路)し、枝先点を調査中に追加, そうでないなら何もしない



Dijkstra法 (更新法)

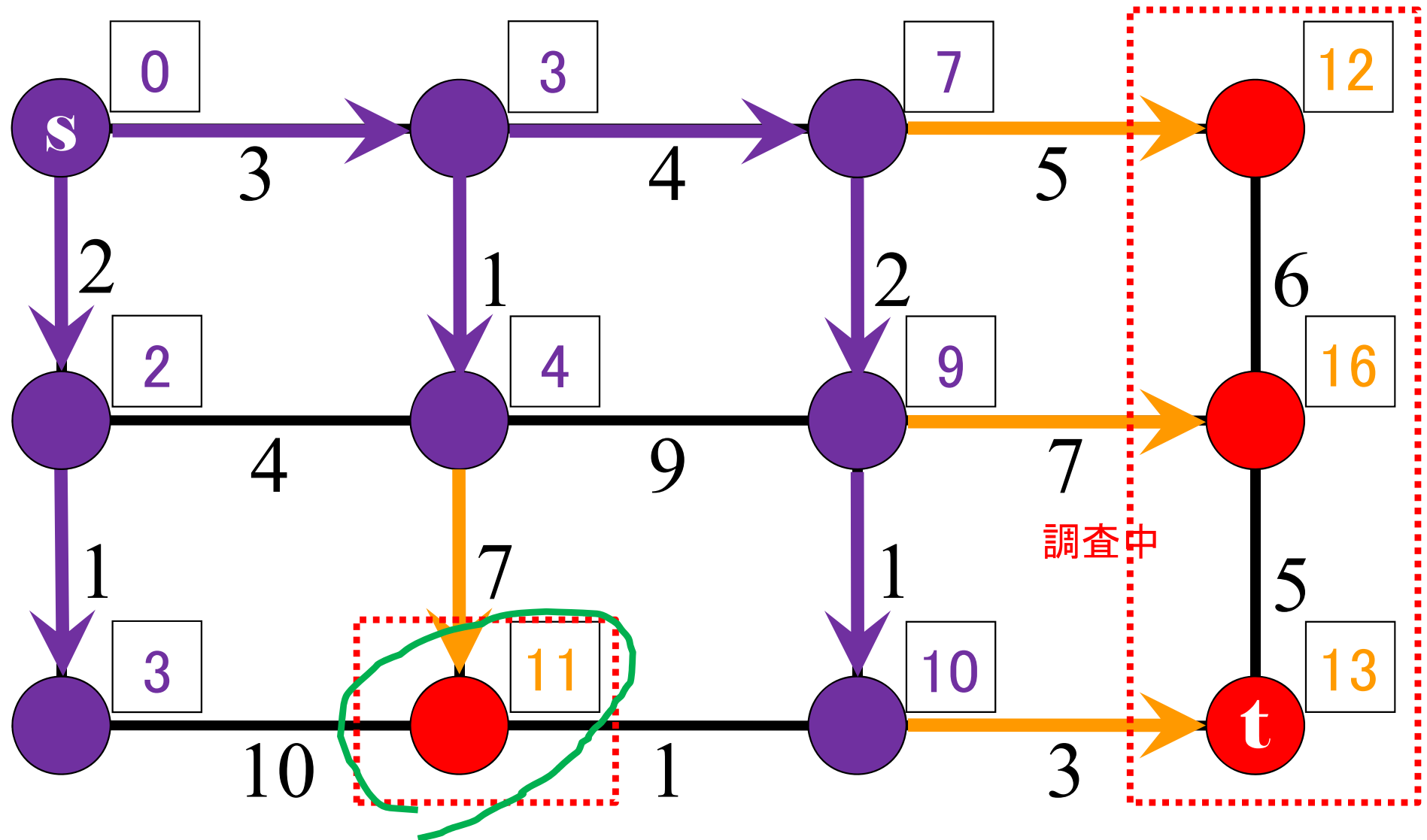
step1-3: 全枝終了したら, 調査中から外しラベル値確定
(その点までの暫定最短路が最短路として確定)

以降step1-1~step1-3を
終了条件を満たすまで繰り返す



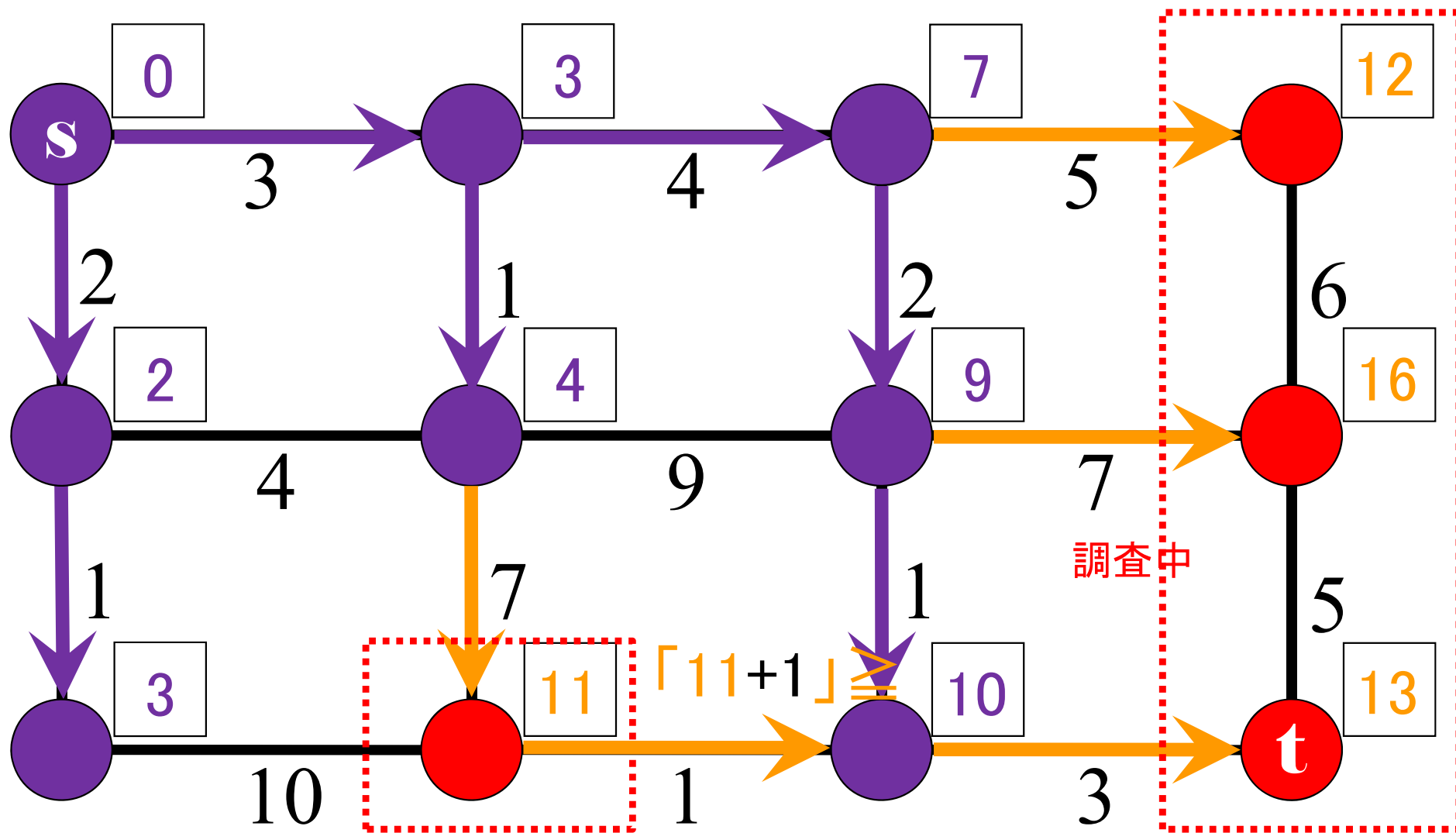
Dijkstra法 (更新法)

step1-1: 調査中点のうちで、ラベル値が最小の点を見つける



Dijkstra法 (更新法)

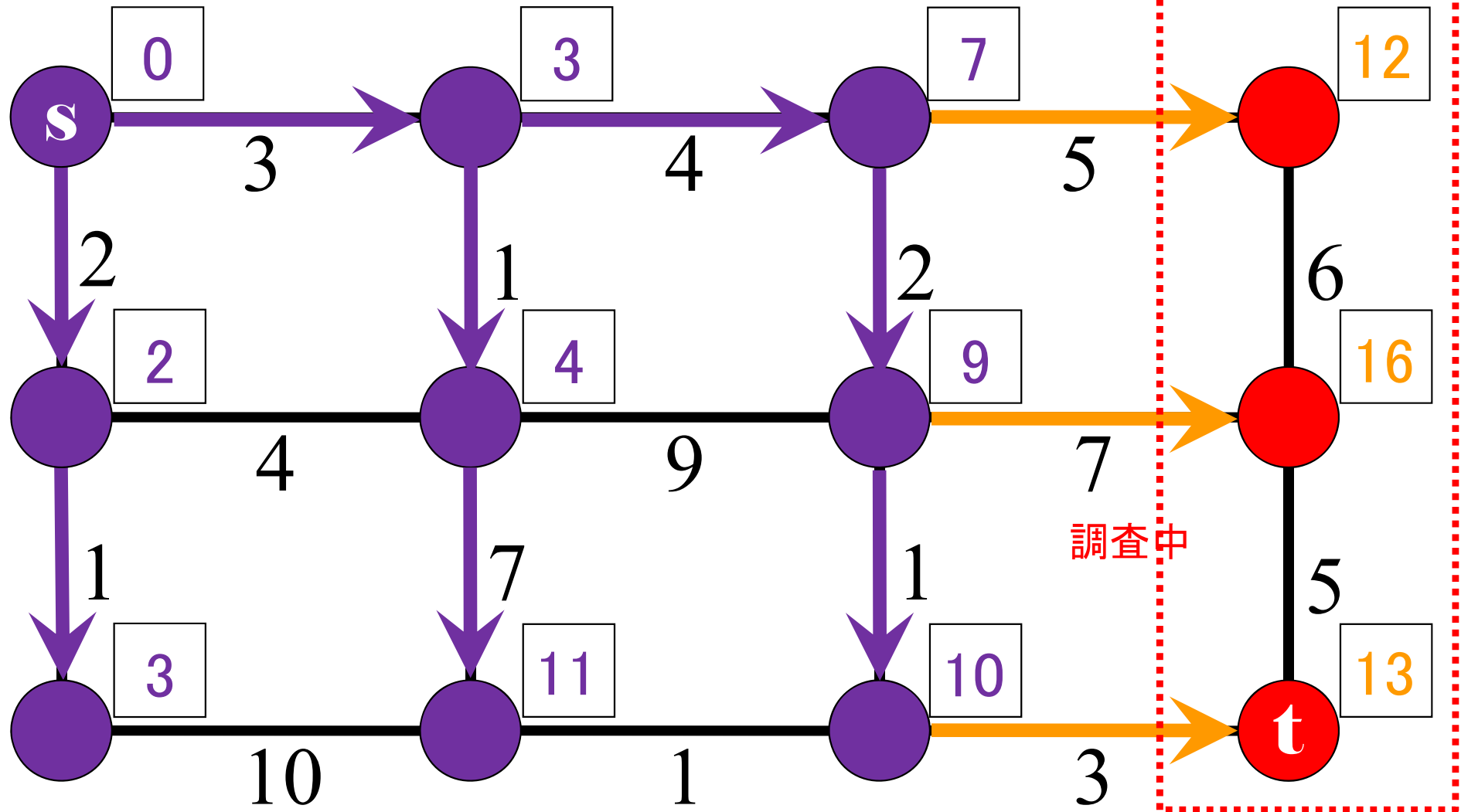
step1-2: その点から出る全枝について「ラベル+枝コスト」を計算し、枝先点のラベル値と比較し、もし、小さい($<$)→枝先点のラベル更新(暫定最短路)し、枝先点を調査中に追加, そうでないなら何もしない



Dijkstra法 (更新法)

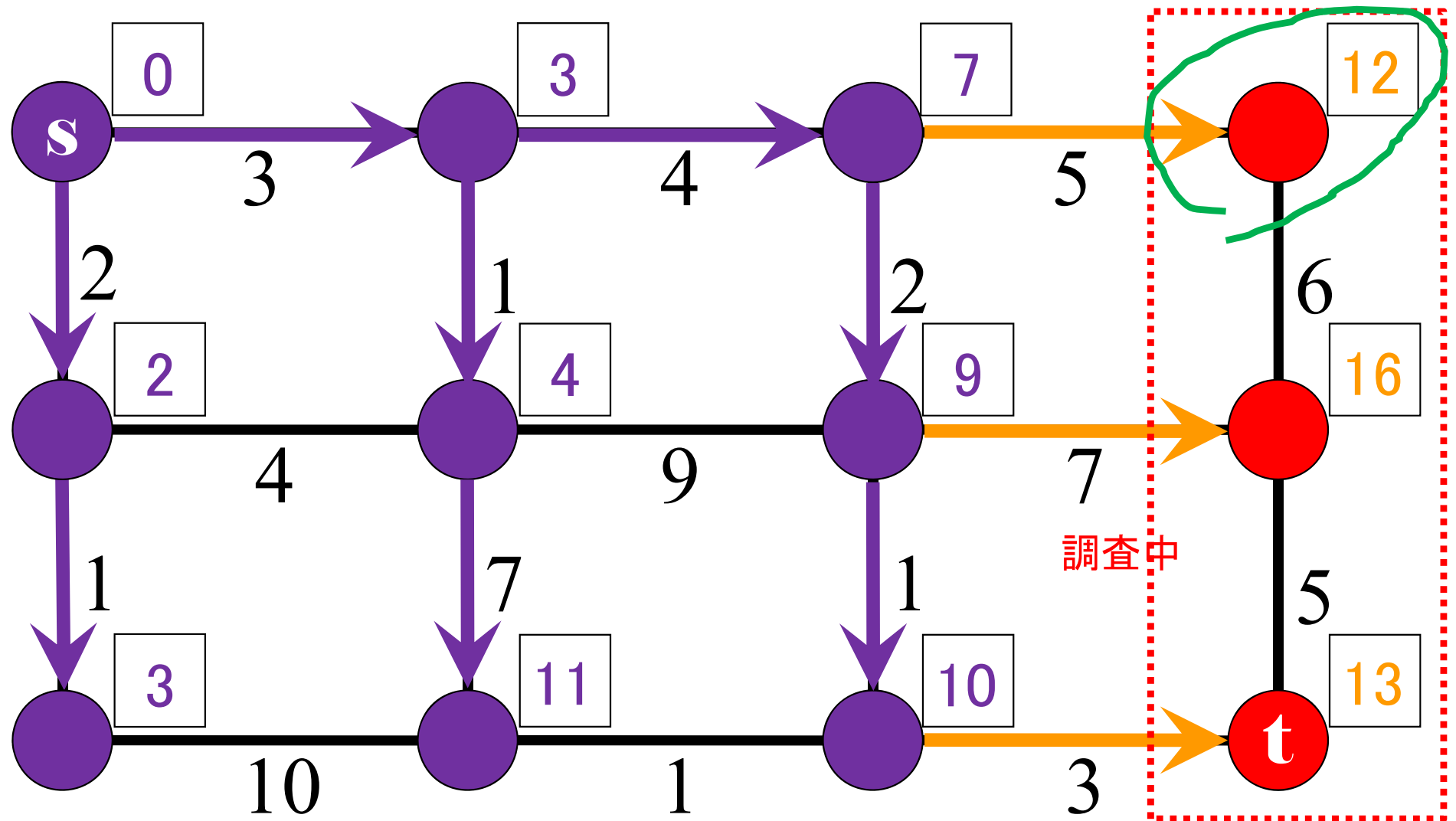
step1-3: 全枝終了したら, 調査中から外しラベル値確定
(その点までの暫定最短路が最短路として確定)

以降step1-1~step1-3を
終了条件を満たすまで繰り返す



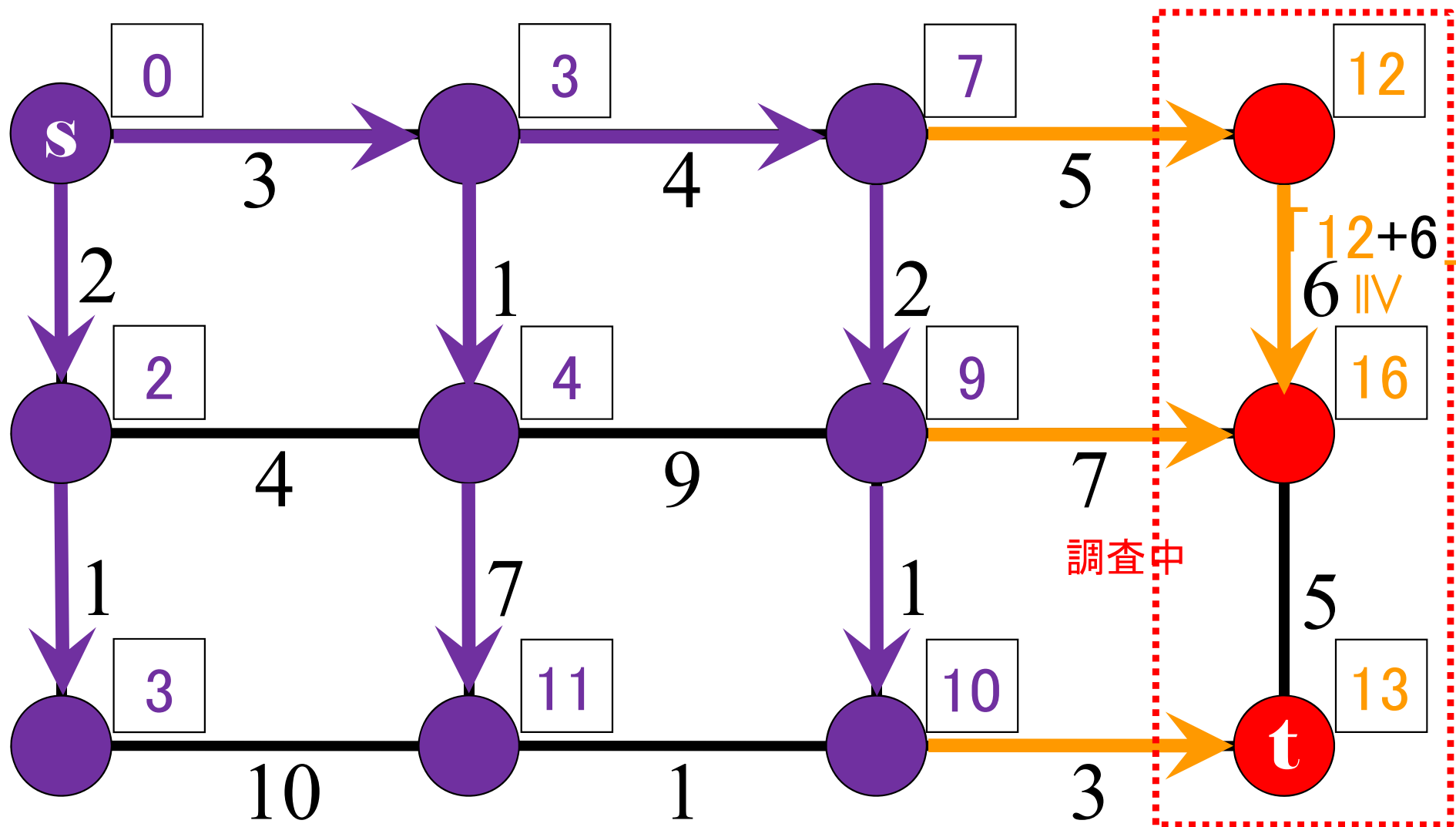
Dijkstra法 (更新法)

step1-1: 調査中点のうちで、ラベル値が最小の点を見つける



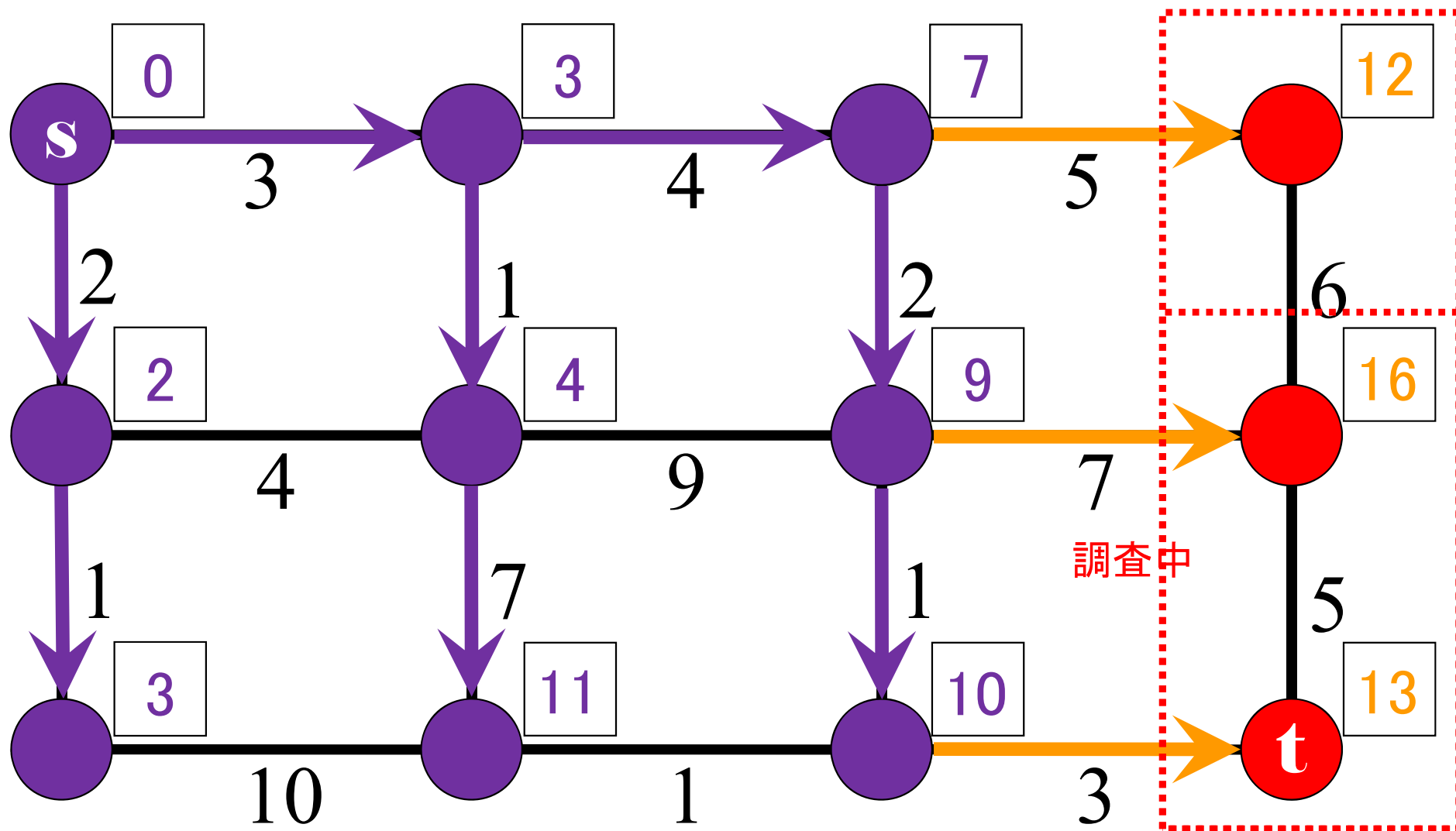
Dijkstra法 (更新法)

step1-2: その点から出る全枝について「ラベル+枝コスト」を計算し、枝先点のラベル値と比較し、もし、小さい($<$)→枝先点のラベル更新(暫定最短路)し、枝先点を調査中に追加, そうでないなら何もしない



Dijkstra法 (更新法)

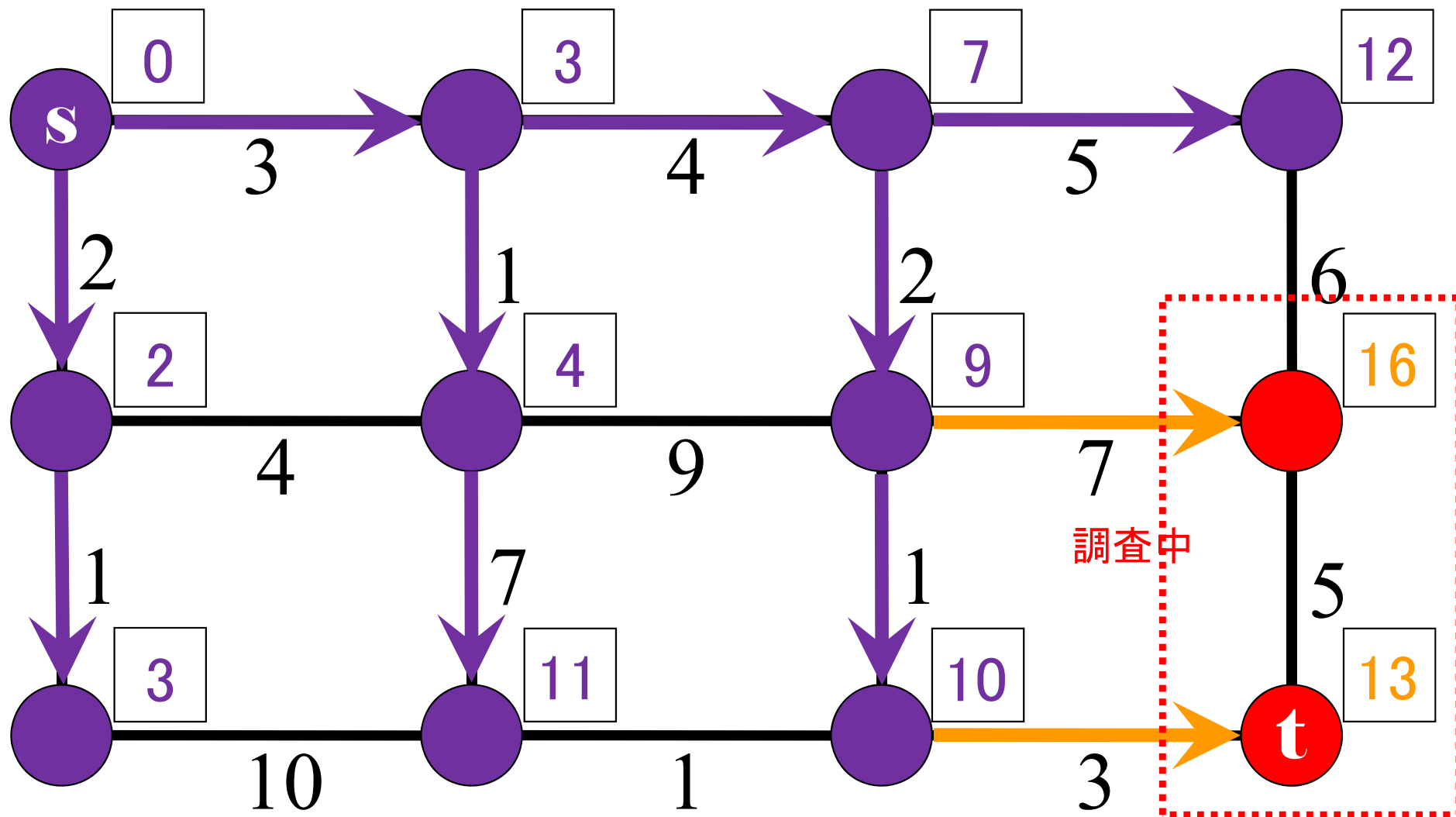
step1-2: その点から出る全枝について「ラベル+枝コスト」を計算し、枝先点のラベル値と比較し、もし、小さい($<$)→枝先点のラベル更新(暫定最短路)し、枝先点を調査中に追加, そうでないなら何もしない



Dijkstra法 (更新法)

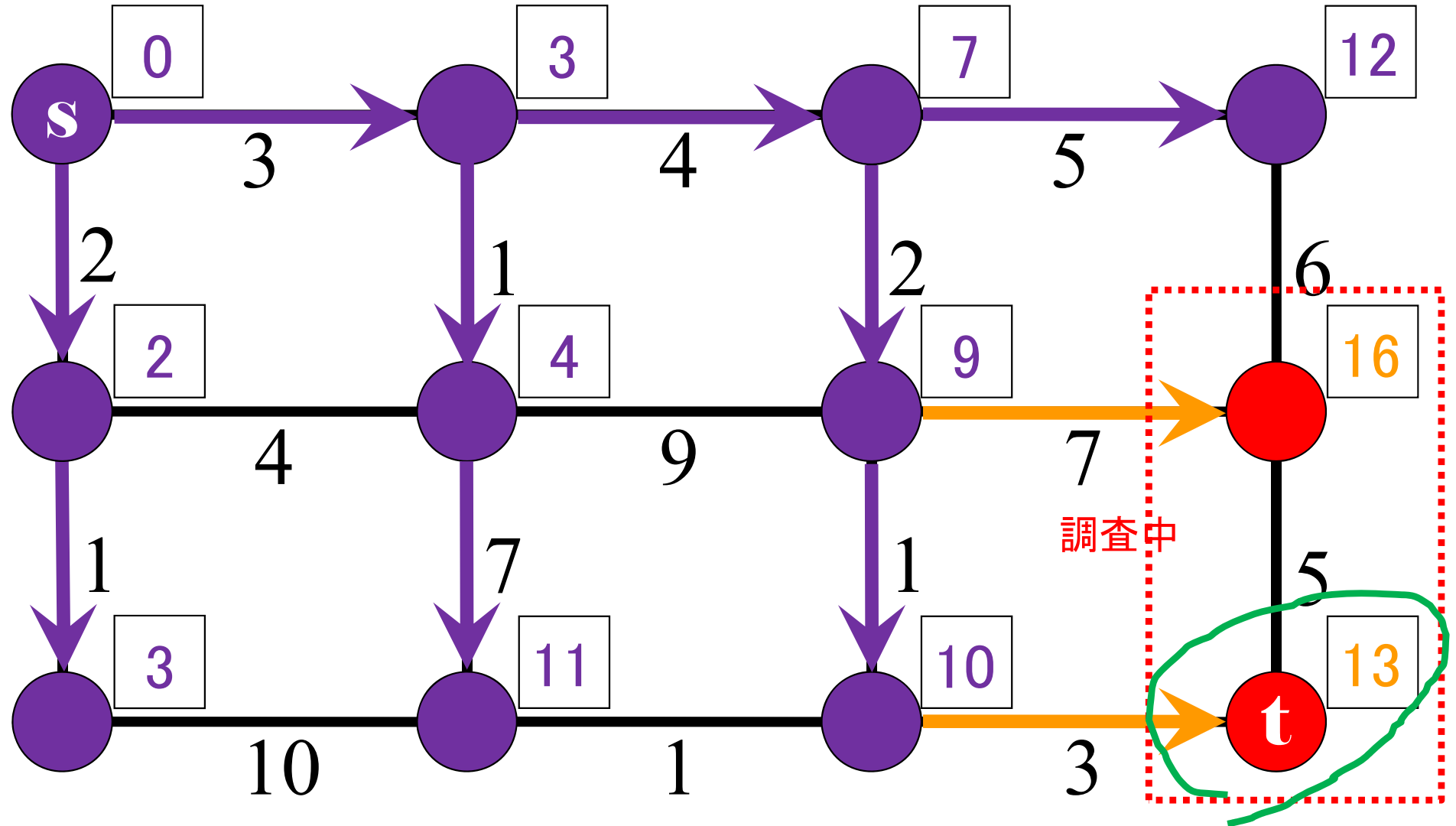
step1-3: 全枝終了したら, 調査中から外しラベル値確定
(その点までの暫定最短路が最短路として確定)

以降step1-1~step1-3を
終了条件を満たすまで繰り返す



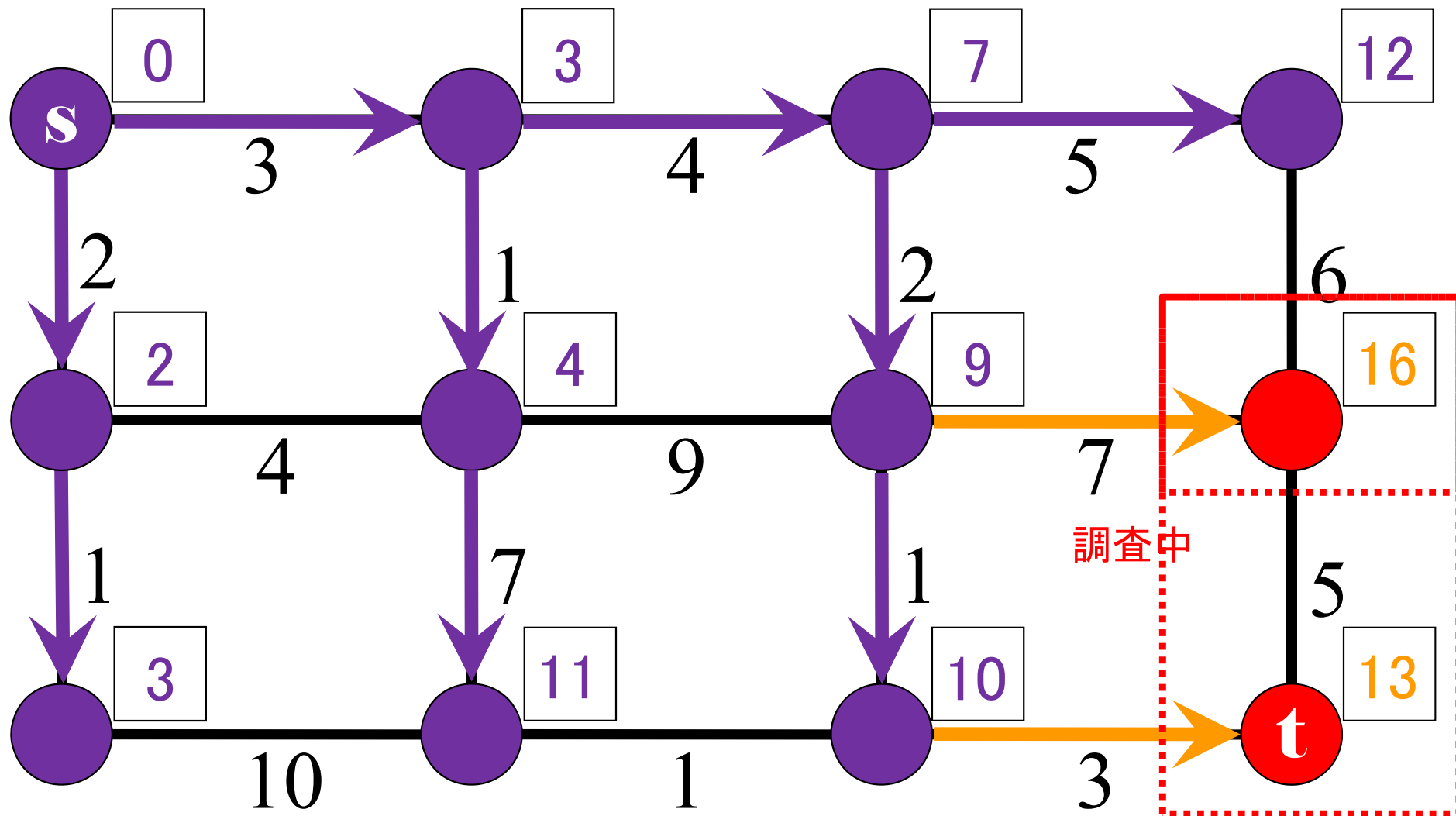
Dijkstra法 (更新法)

step1-1: 調査中点のうちで、ラベル値が最小の点を見つける



Dijkstra法 (更新法)

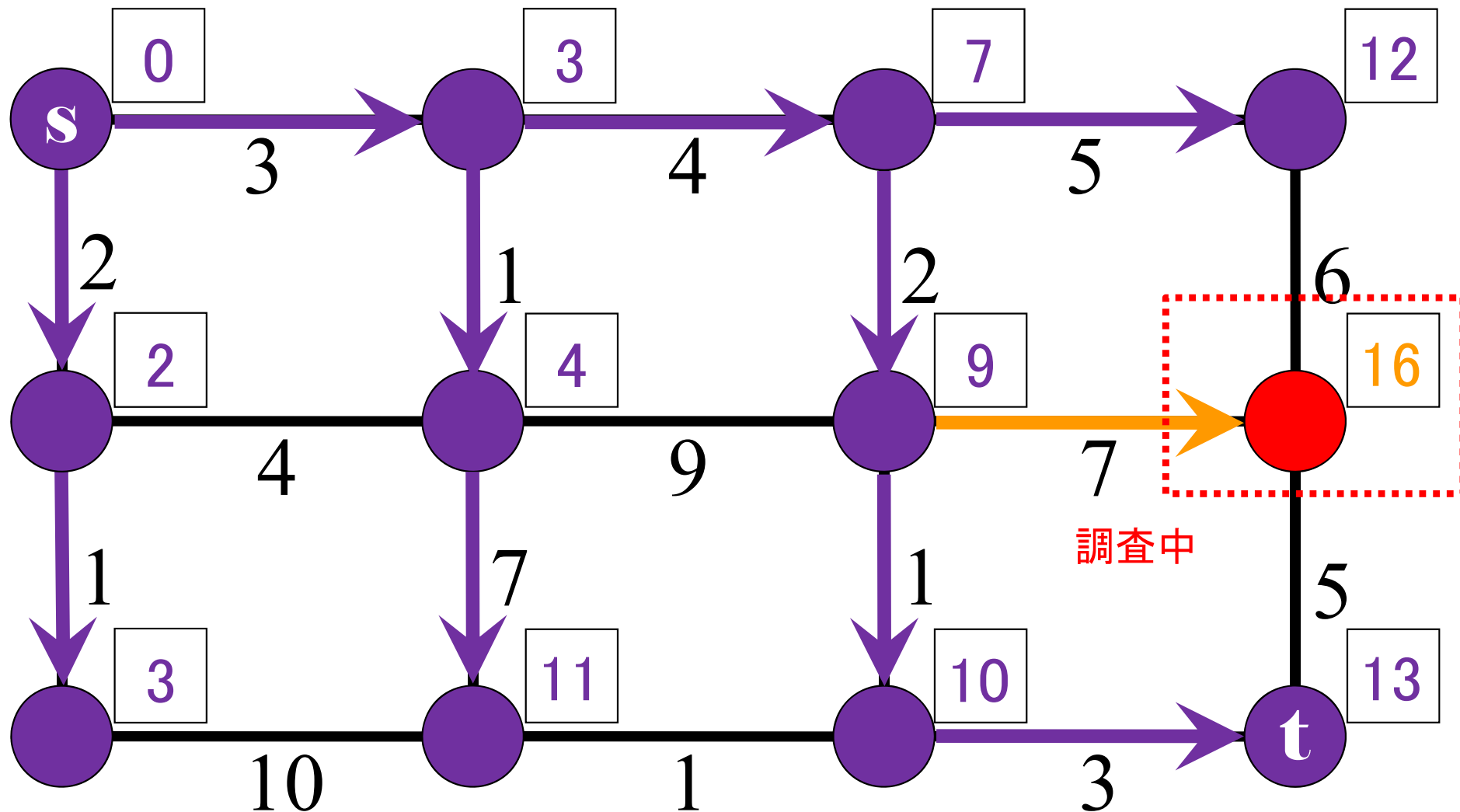
step1-2: その点から出る全枝について「ラベル+枝コスト」を計算し、枝先点のラベル値と比較し、もし、小さい($<$)→枝先点のラベル更新(暫定最短路)し、枝先点を調査中に追加, そうでないなら何もしない



Dijkstra法 (更新法)

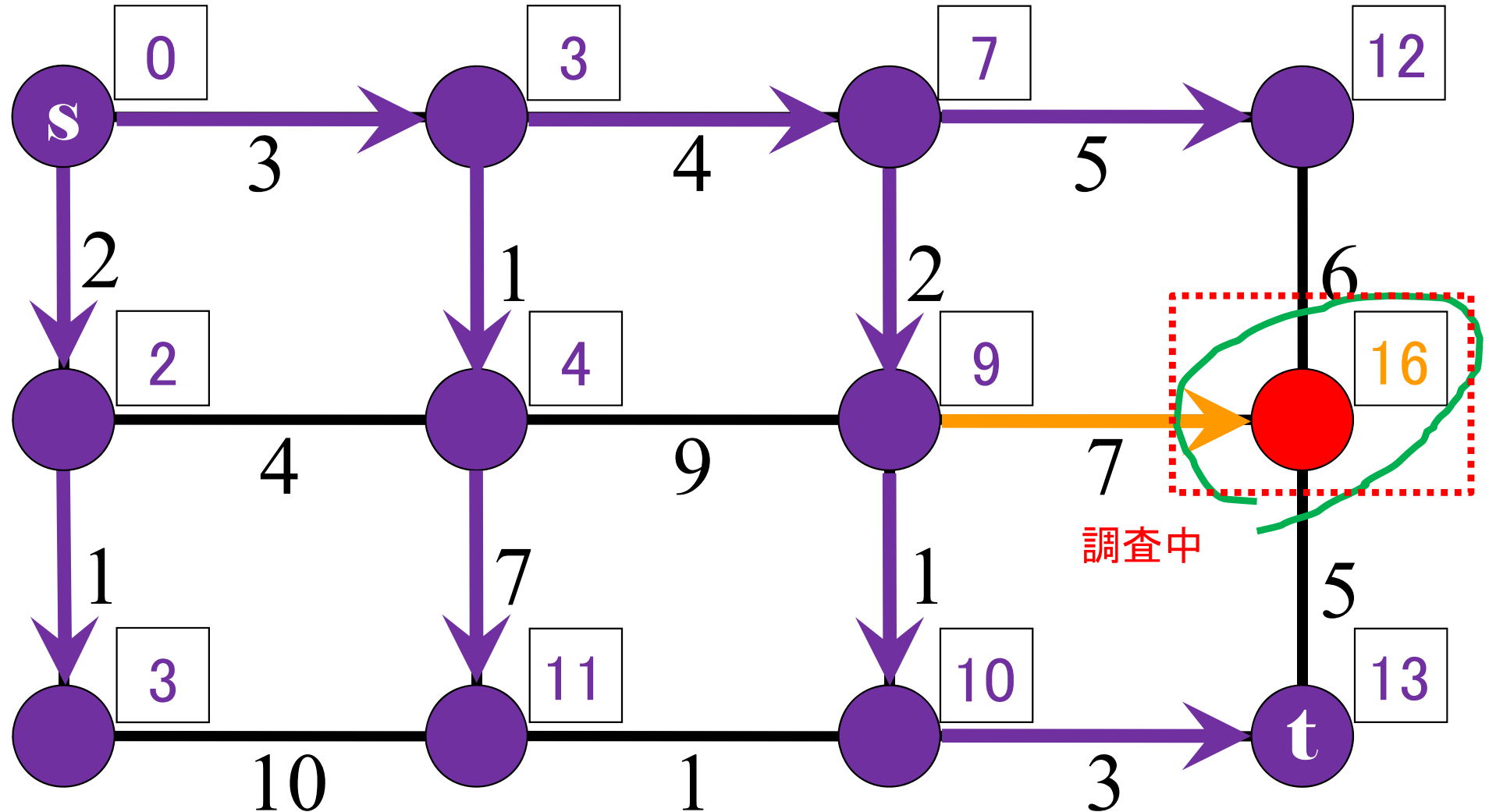
step1-3: 全枝終了したら, 調査中から外しラベル値確定
(その点までの暫定最短路が最短路として確定)

以降step1-1~step1-3を
終了条件を満たすまで繰り返す



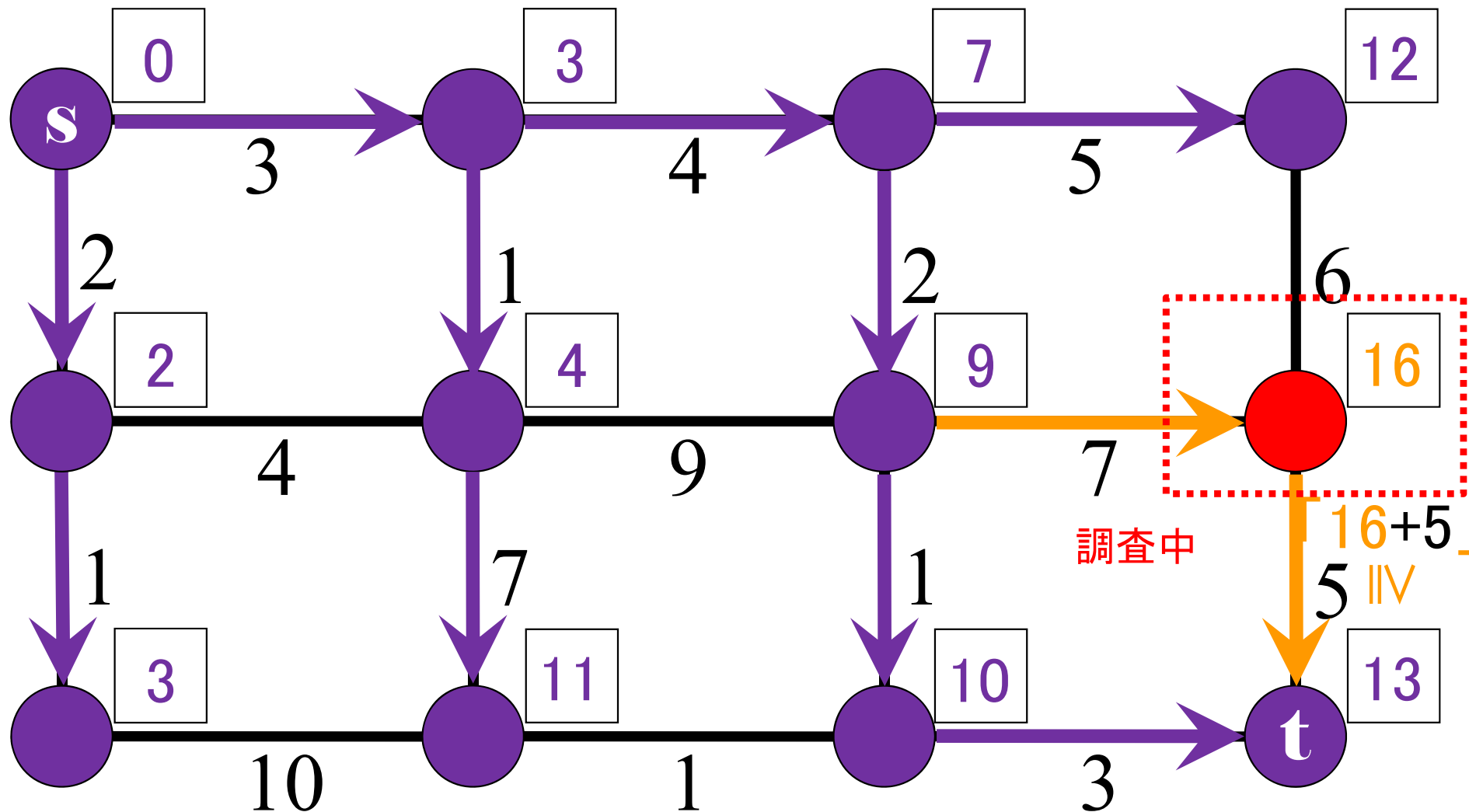
Dijkstra法 (更新法)

step1-1: 調査中点のうちで、ラベル値が最小の点を見つける



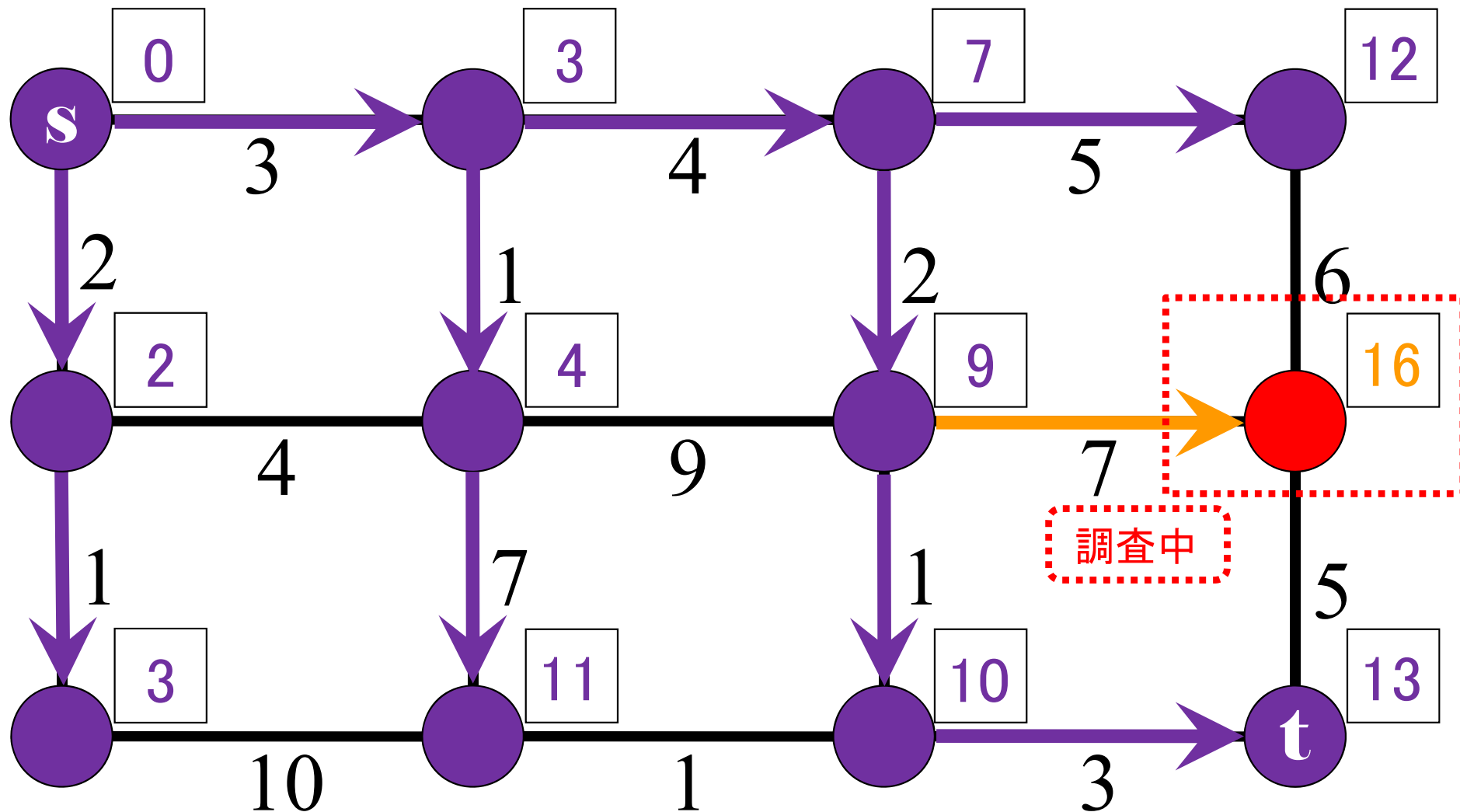
Dijkstra法 (更新法)

step1-2: その点から出る全枝について「ラベル+枝コスト」を計算し、枝先点のラベル値と比較し、もし、小さい($<$)→枝先点のラベル更新(暫定最短路)し、枝先点を調査中に追加, そうでないなら何もしない



Dijkstra法 (更新法)

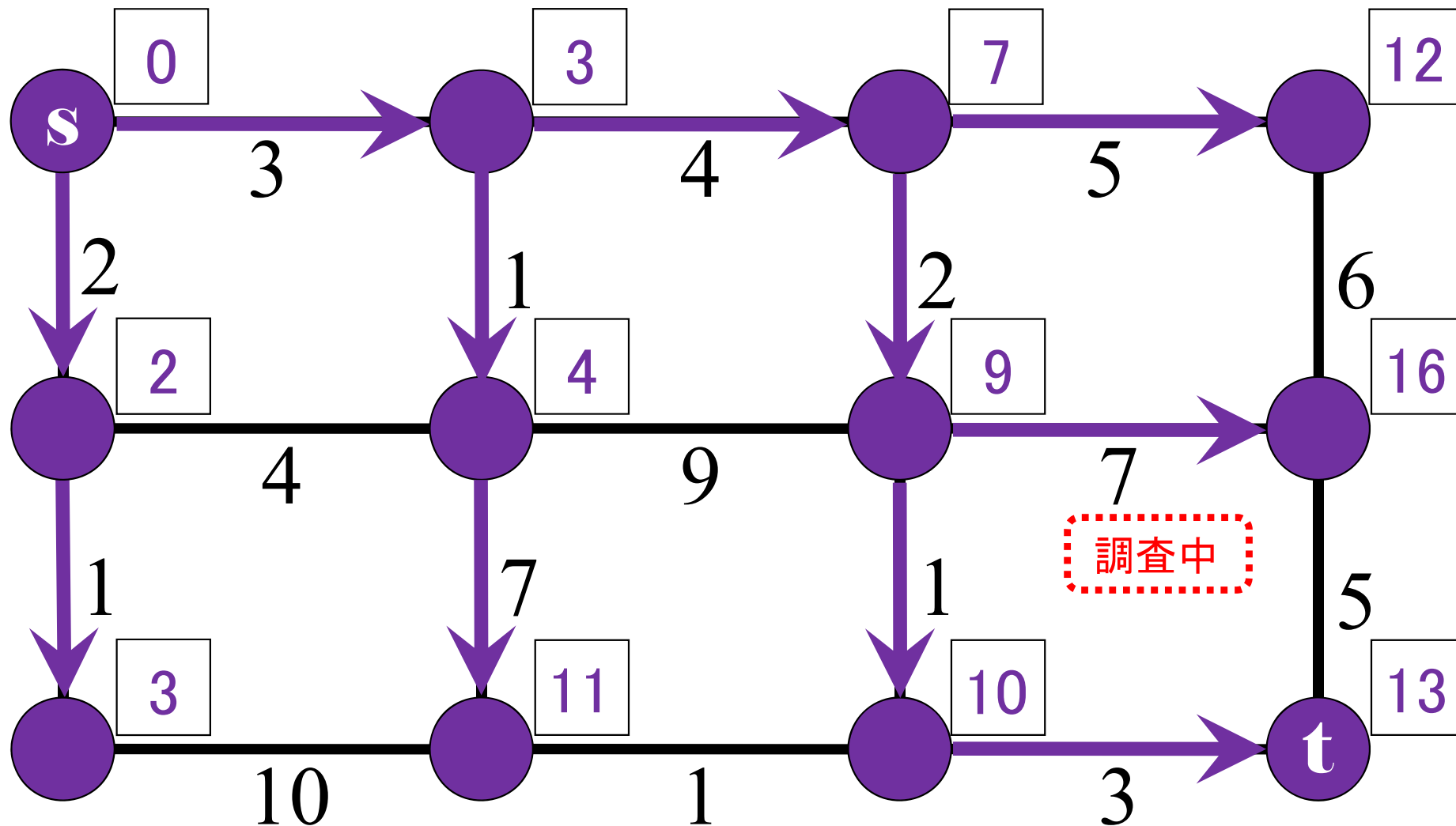
step1-2: その点から出る全枝について「ラベル+枝コスト」を計算し、枝先点のラベル値と比較し、もし、小さい($<$)→枝先点のラベル更新(暫定最短路)し、枝先点を調査中に追加, そうでないなら何もしない



Dijkstra法 (更新法)

step1-3: 全枝終了したら, 調査中から外しラベル値確定
(その点までの暫定最短路が最短路として確定)

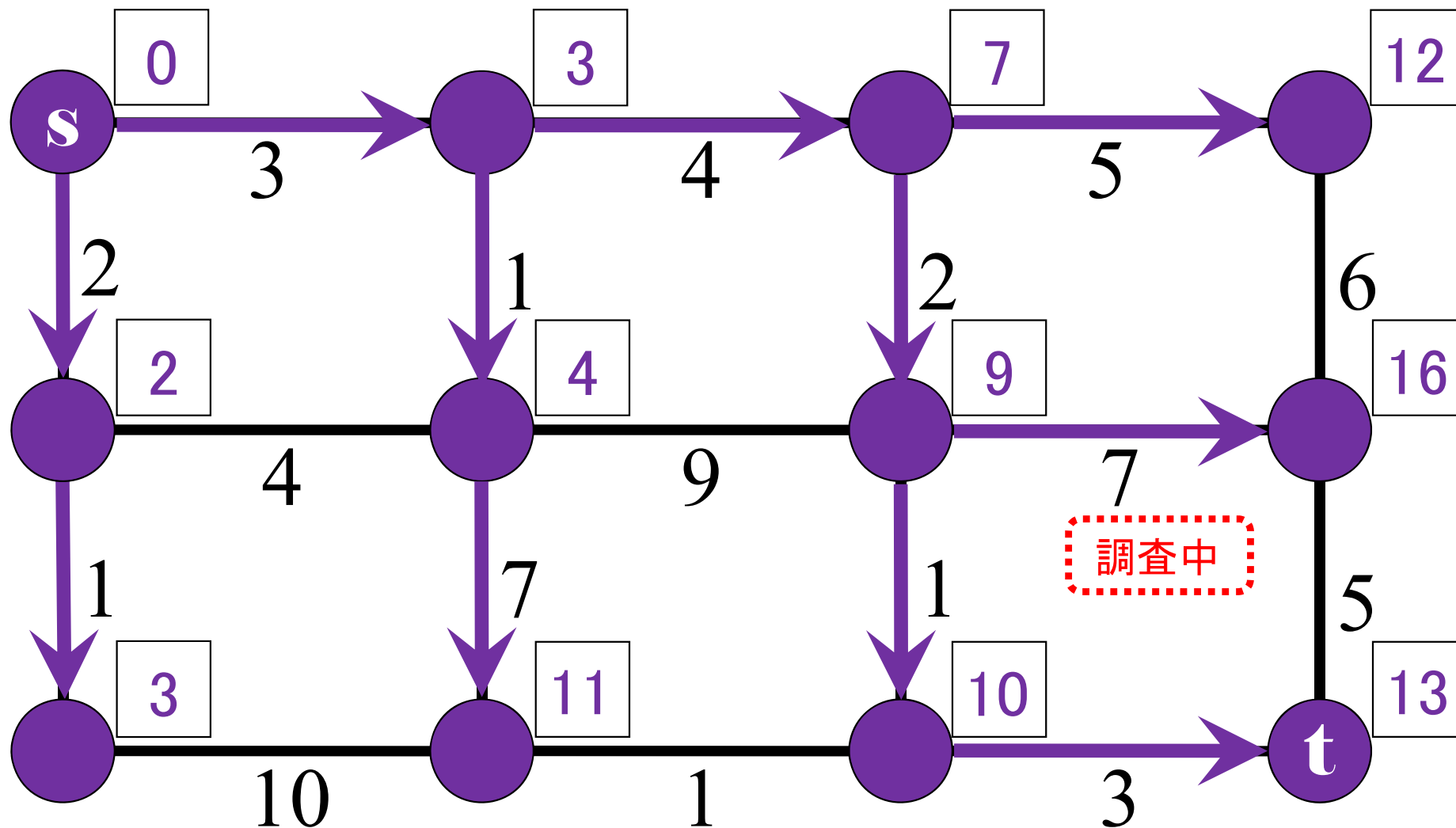
以降step1-1~step1-3を
終了条件を満たすまで繰り返す



Dijkstra法

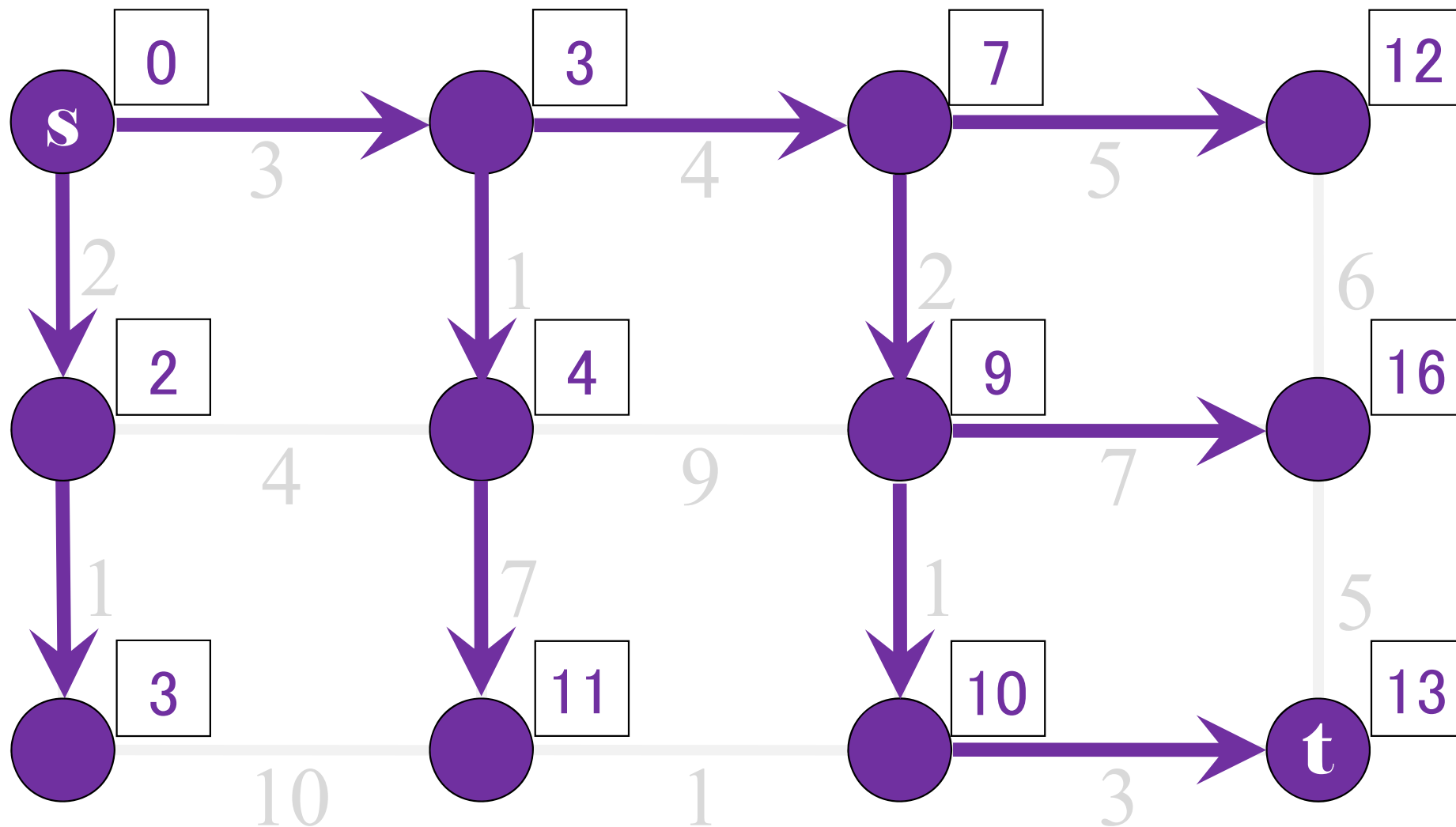
(終了判定)

step2: 調査中の点集合が空なら終了



Dijkstra法終了後

【注】「スタートから全頂点への最短路が求まっている」ことに注意



Dijkstra法

(初期設定)

step0: start点のラベルを0にし, その他のラベルを ∞ に設定する

start点 **S** を 調査中 の点集合 $\{s\}$ とする

(更新法)

step1-1: 調査中点のうちで, ラベル値が最小の点を見つける

step1-2: その点から出る全枝について「ラベル+枝コスト」を計算し, 枝先点のラベル値と比較し,
もし, 小さい($<$) \rightarrow 枝先点の ラベル更新 (暫定最短路) し,
枝先点を調査中に追加, そうでないなら何もしない

step1-3: 全枝終了したら, 調査中から外しラベル値確定
(その点までの 暫定最短路が最短路 として確定)

step1-1 ~ **step1-3** を 終了条件を満たすまで繰り返す

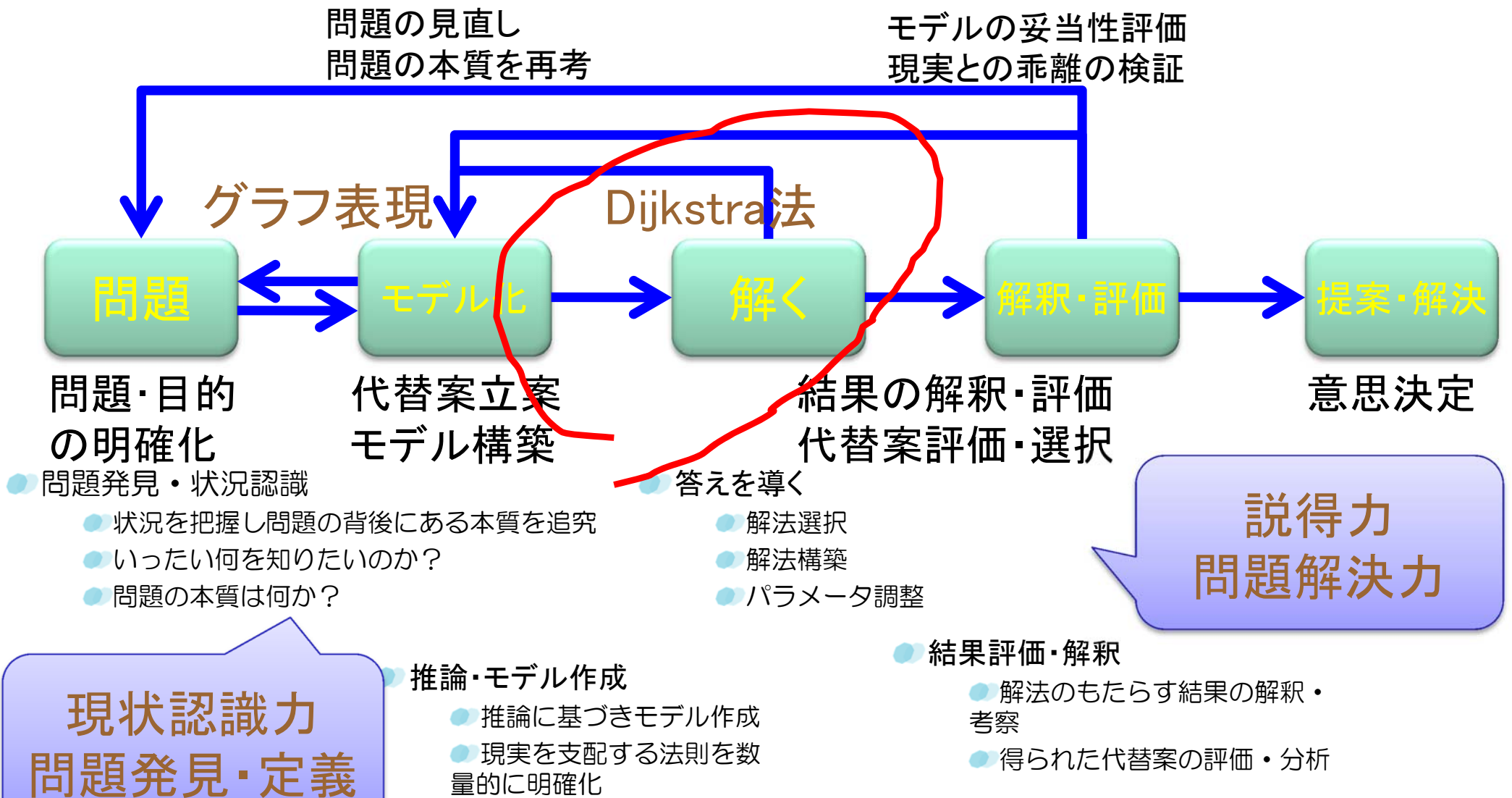
(終了判定)

step2: 調査中の点集合 $\{ \quad \}$ が 空 なら終了

意思決定

論理的思考力
データ分析, 統計学
数理的アプローチ

- 「問題の把握」から「意思決定」までの流れ



評価: Dijkstra法って速いのか?



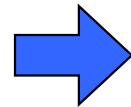
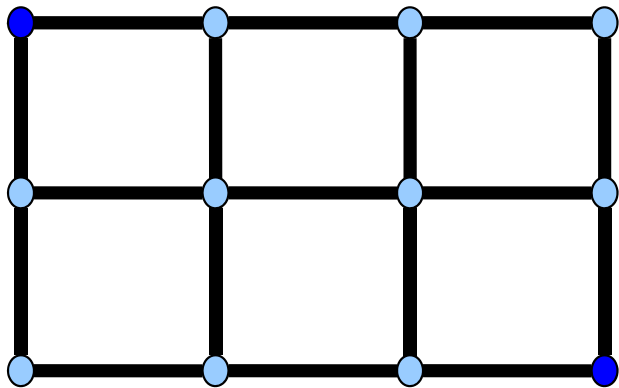
- 点の数を n とすると, 大雑把な見積もりで,

$$O(n^2) \quad \text{多項式オーダー}$$

$$O(m + n \log n)$$

- 点の数 n を右向枝数 R , 下向枝数 D で表すと

$$n = (R + 1) \times (D + 1)$$



$$n = (3 + 1) \times (2 + 1) = 12$$

$$n^2 = 12^2 = 144$$

コンピュータに計算させてみよう!

簡単のため n^2 の5倍の浮動小数点演算回数で計算できると仮定.

評価: Dijkstra法って速いのか？

10.51PFLOPS

51.2GFLOPS

| R(横) D(縦) | 全経路 | 京 & しらみつぶし | Core i7 & Dijkstra |
|-----------|-----------------------|--------------------------|--------------------|
| 3 2 | 10 | 0.0000000000 秒 | 0.0000000001 秒 |
| 6 4 | 210 | 0.0000000000 秒 | 0.0000000003 秒 |
| 10 5 | 3,003 | 0.0000000000 秒 | 0.0000000006 秒 |
| 20 10 | 30,045,015 | 0.0000000086 秒 | 0.0000000023 秒 |
| 25 25 | 1.3×10^{14} | 0.601382523 秒 | 0.0000000066 秒 |
| 30 30 | 1.2×10^{17} | 11 分 | 0.0000000094 秒 |
| 40 40 | 1.1×10^{23} | 26 年 | 0.0000000164 秒 |
| 50 50 | 1.0×10^{29} | 30,439,996 年 | 0.0000000254 秒 |
| 100 100 | 9.1×10^{58} | 4.0×10^{27} 宙齡 | 0.0000000996 秒 |
| 500 500 | 2.7×10^{299} | 5.9×10^{268} 宙齡 | 0.000024512 秒 |

世界最速 SuperComp
+ 力技 (しょぼい方法)

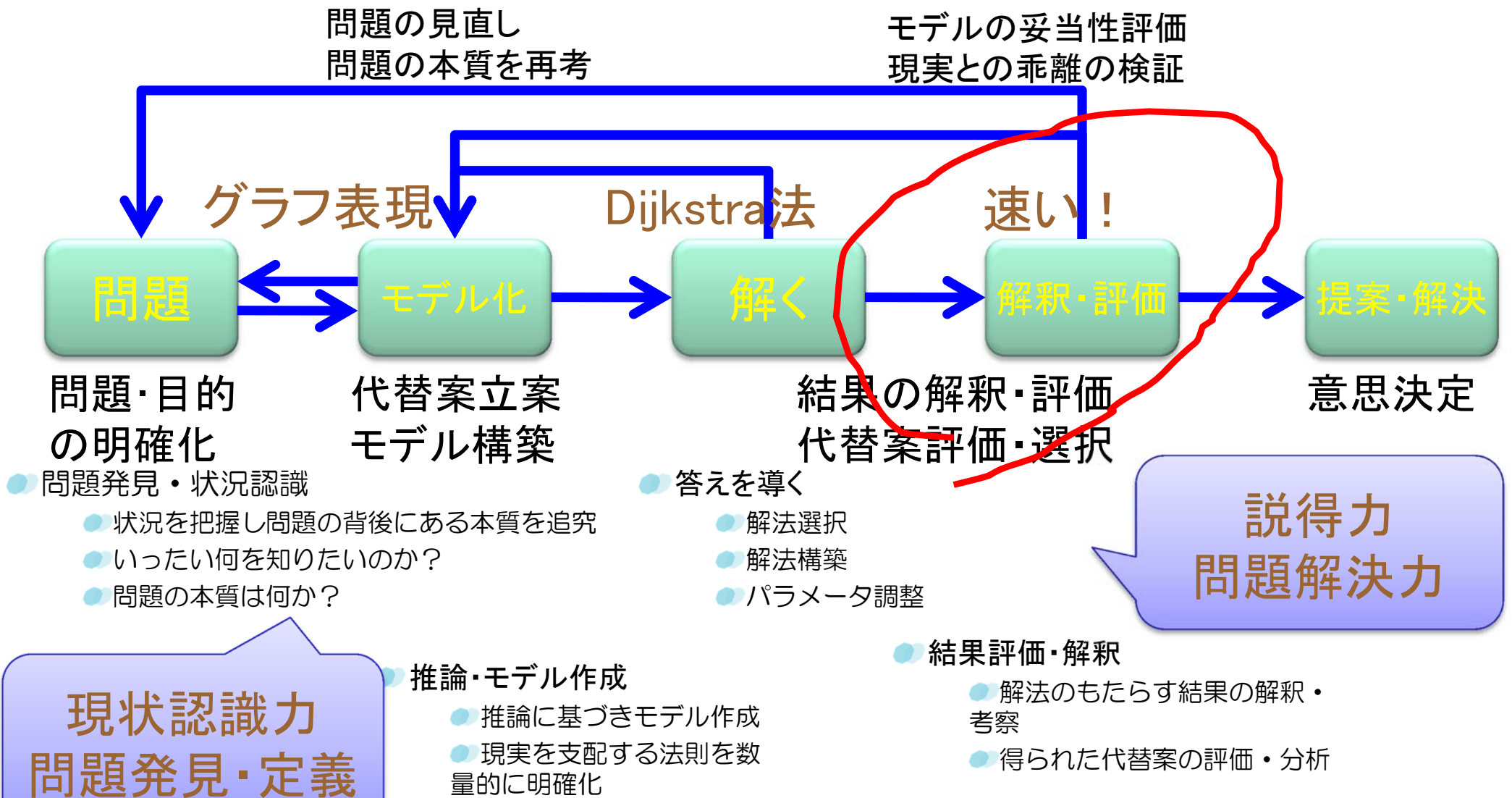


そこらのPC
+ 人間の知恵

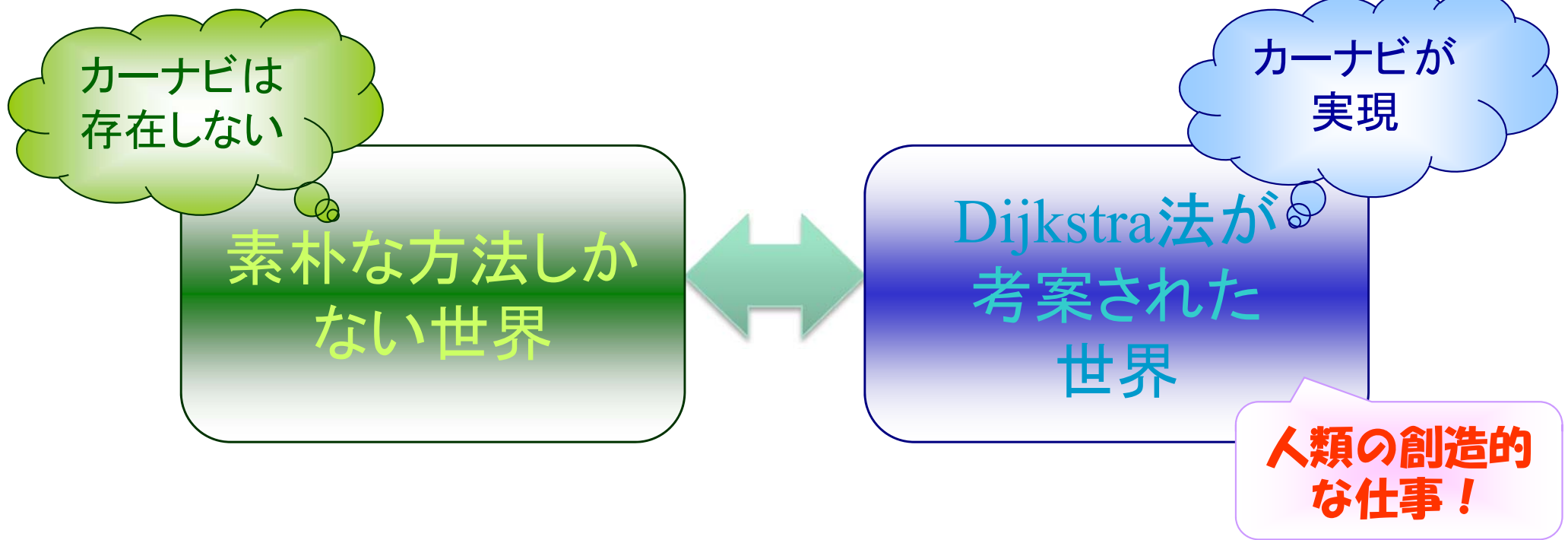
意思決定

論理的思考力
データ分析, 統計学
数理的アプローチ

- 「問題の把握」から「意思決定」までの流れ



意思決定支援・ビジネスサポート



参考文献

コンピュータに仕事を奪われつつある人類...

- [1] 新井紀子
「コンピュータが仕事を奪う」日経新聞社(2010)
- [2] E. Brynjolfsson, A. McAfee, 村井章子訳
「機械との競争」日経BP社(2013)



もっと知りたい人へ

- 参考文献

- グリッツマン, ブランデンベルク「**最短経路の本**」 シュプリンガー(2008)
- W.J.クック「**驚きの数学 巡回セールスマン問題**」 青土社(2013)
- 山本, 久保「**巡回セールスマン問題への招待**」 朝倉書店(1997)
- 久保, 松井「**組合せ最適化『短編集』**」 朝倉書店(1999)
- 松井, 根本, 宇野「**入門オペレーションズ・リサーチ**」東海大出版(2008)

- 関連する授業

- 「**ネットワークモデル分析**」(4セメ)
- 「**最適化モデル分析**」(5セメ) etc...