

問題解決技法入門

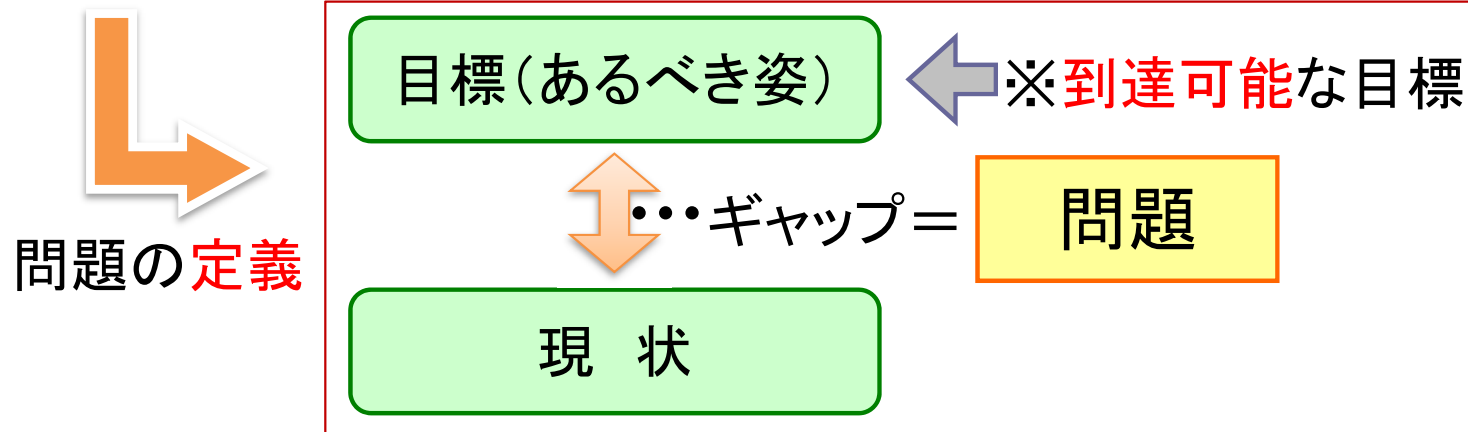
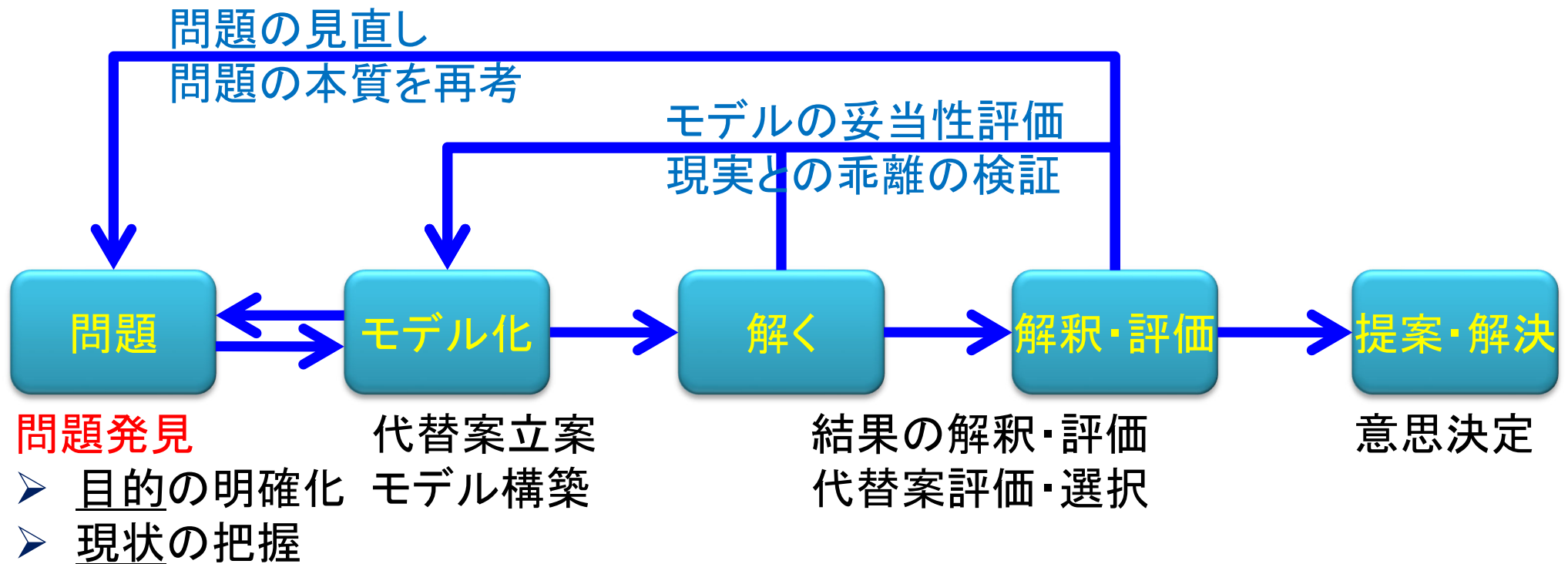
2. Graph / Optimization

3. Shortest Path Problem

堀田 敬介

問題解決とは？

➤ 問題発見・問題解決から意思決定まで

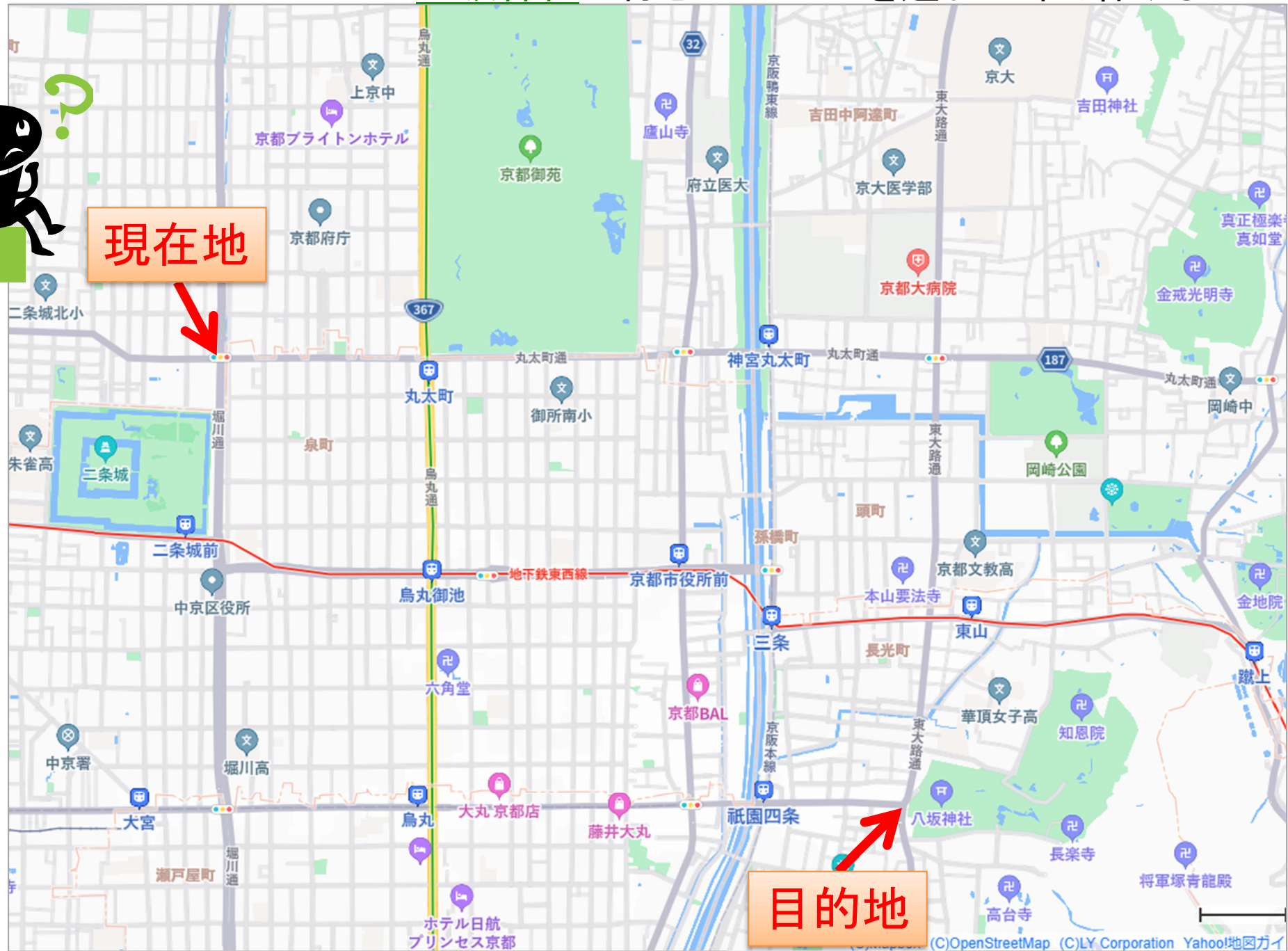


ルート探索

問) 今、堀川丸太町交差点にいる。大通りのみを使い
八坂神社に行きたい。どこを通れば早く着けるか？



現在地



目的地

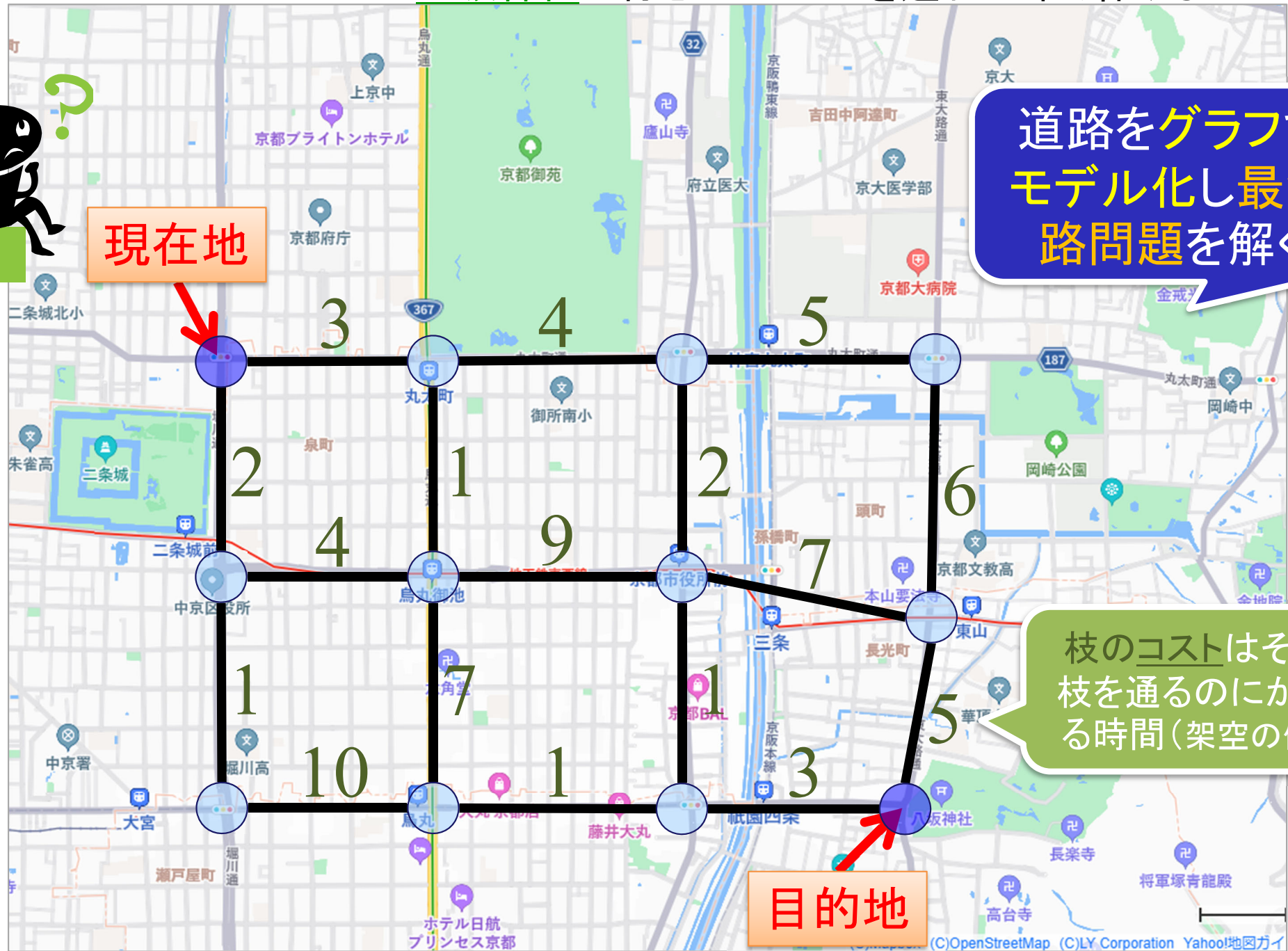
ルート探索

問) 今, 堀川丸太町 交差点にいる. 大通りのみを使い
八坂神社に行きたい. どこを通れば早く着けるか?



現在地

道路をグラフで
モデル化し最短
路問題を解く

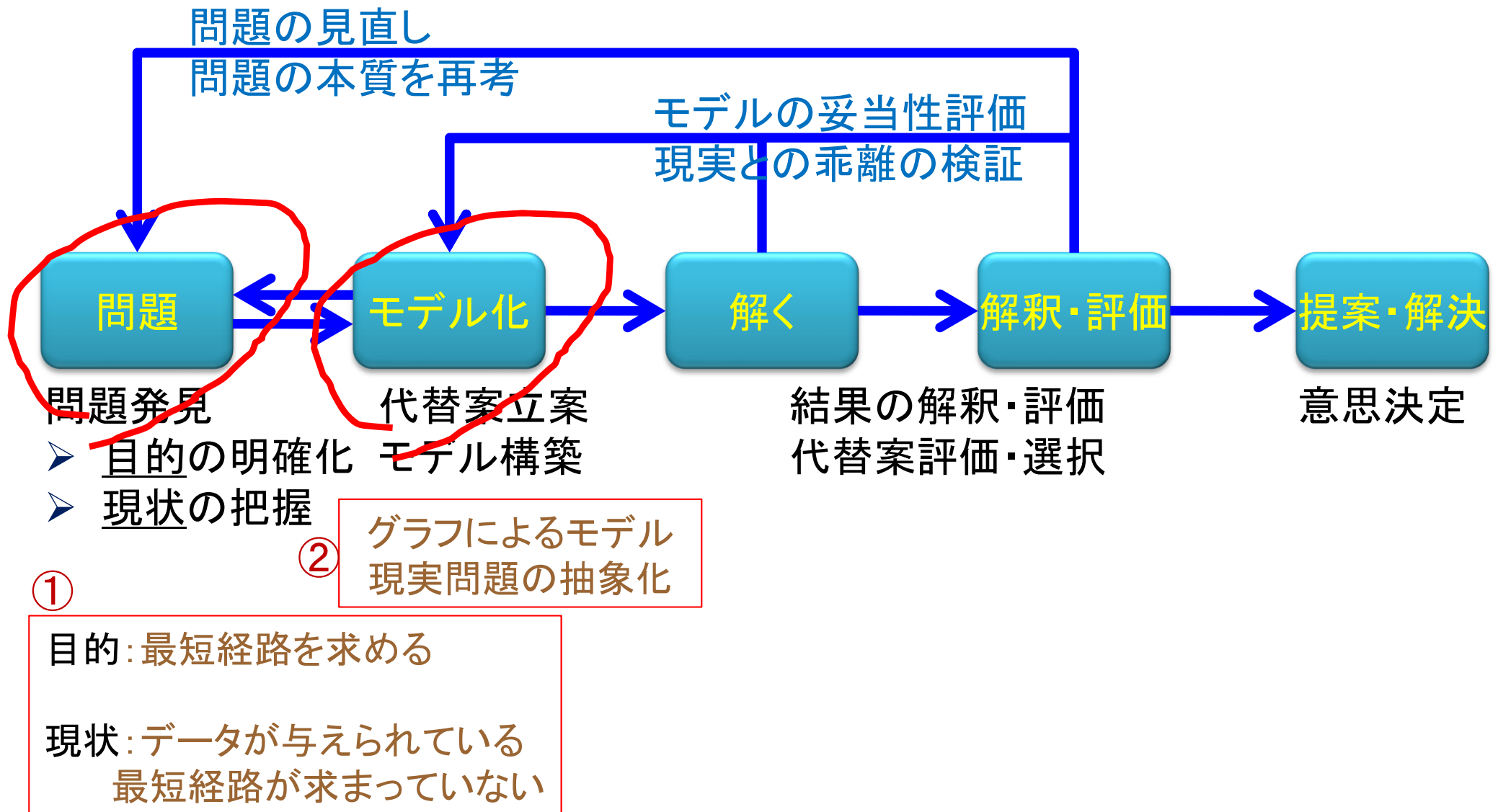


枝のコストはその
枝を通るのにかかる
時間(架空の値)

目的地

問題解決とは？

➤ 問題発見・問題解決から意思決定まで

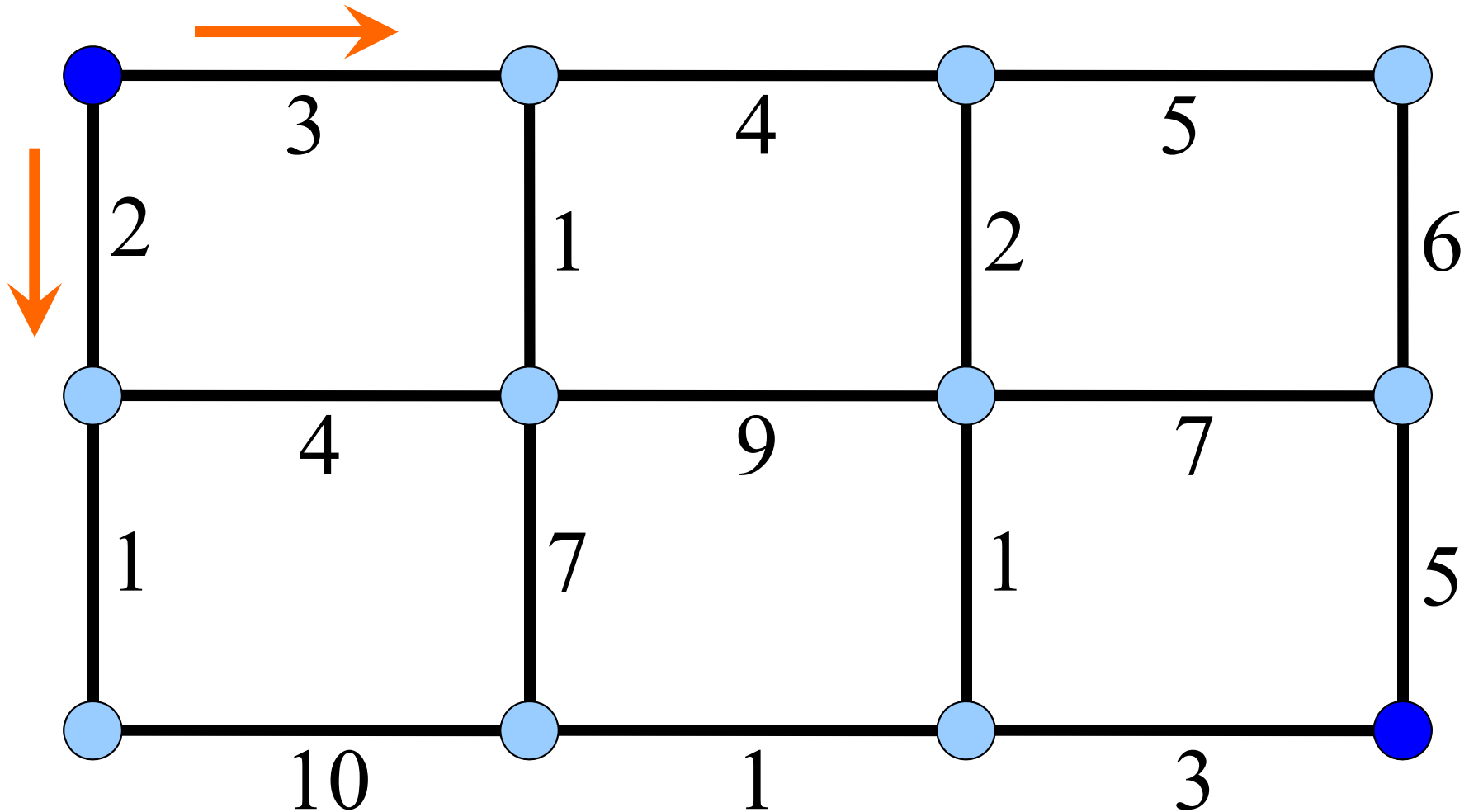


解く: どうやって解くか？

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！

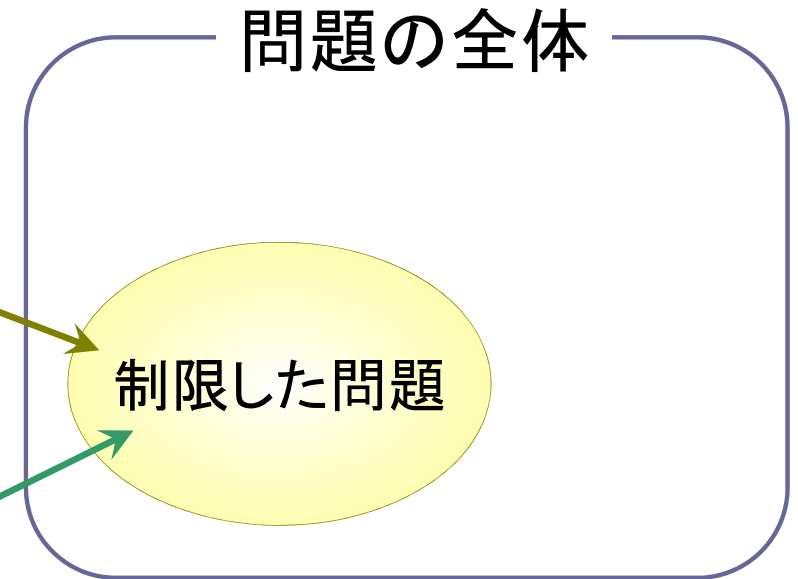


難しいなら易くすればいいのさ！

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！



任意のグラフの最短路問題

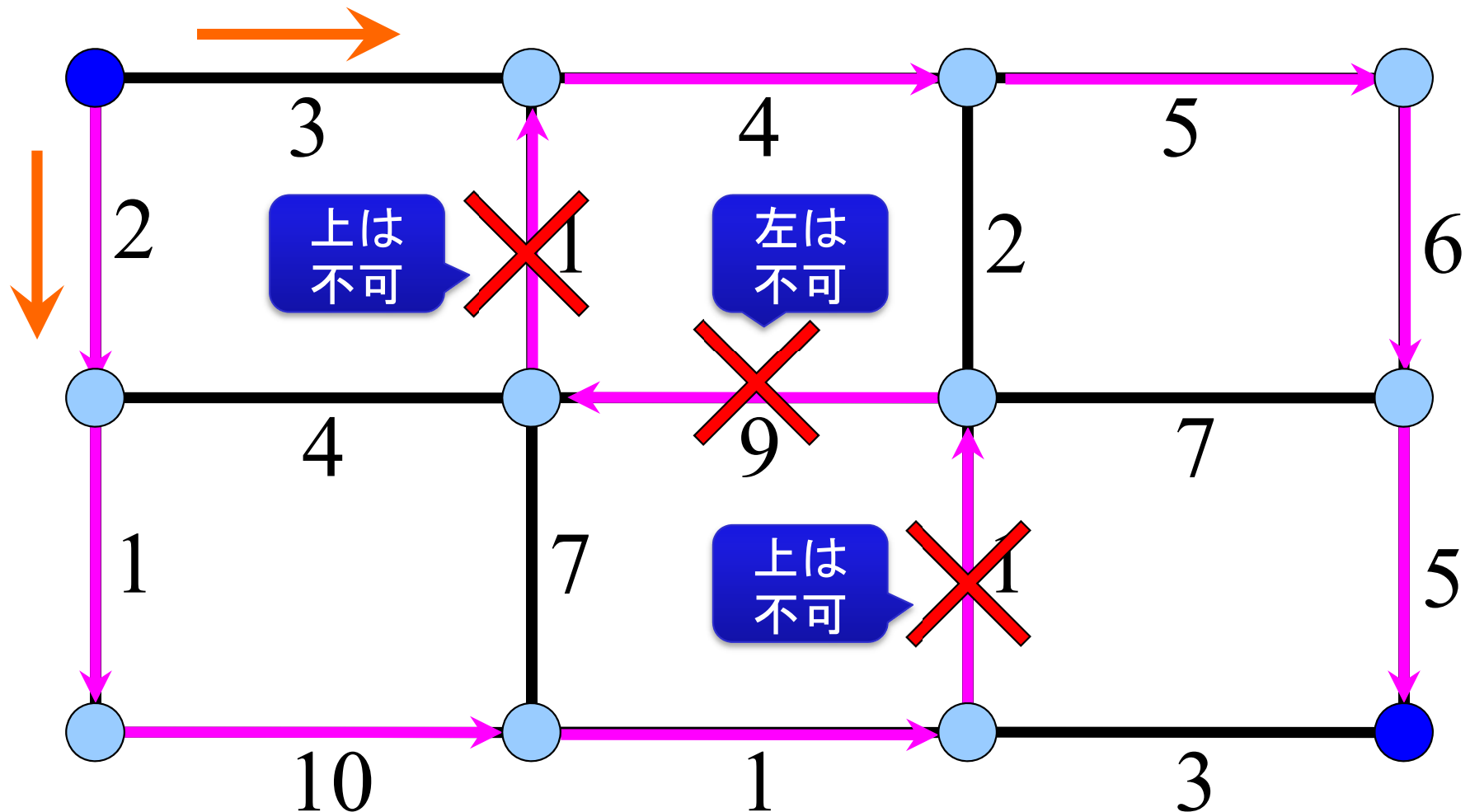
制限した問題

- **格子グラフ**だけを扱う
- 出発地: **左上点**, 目的地: **右下点**に固定
- 移動は**右・下**方向へのみ可

難しいなら易くすればいいのさ！

制限した問題

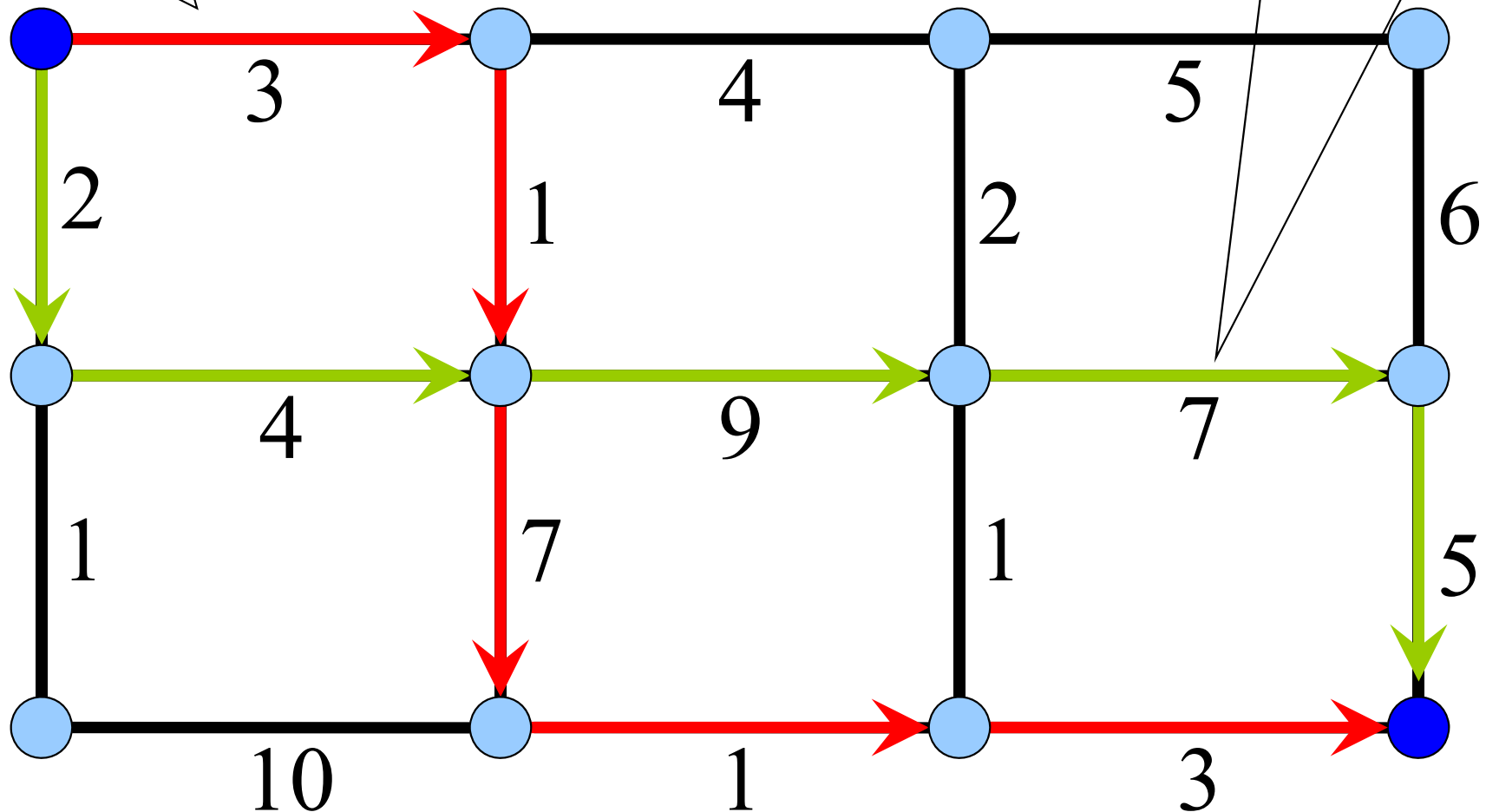
- 格子グラフ
- 出発地: 左上点, 目的地: 右下点
- 移動は右・下方向のみ



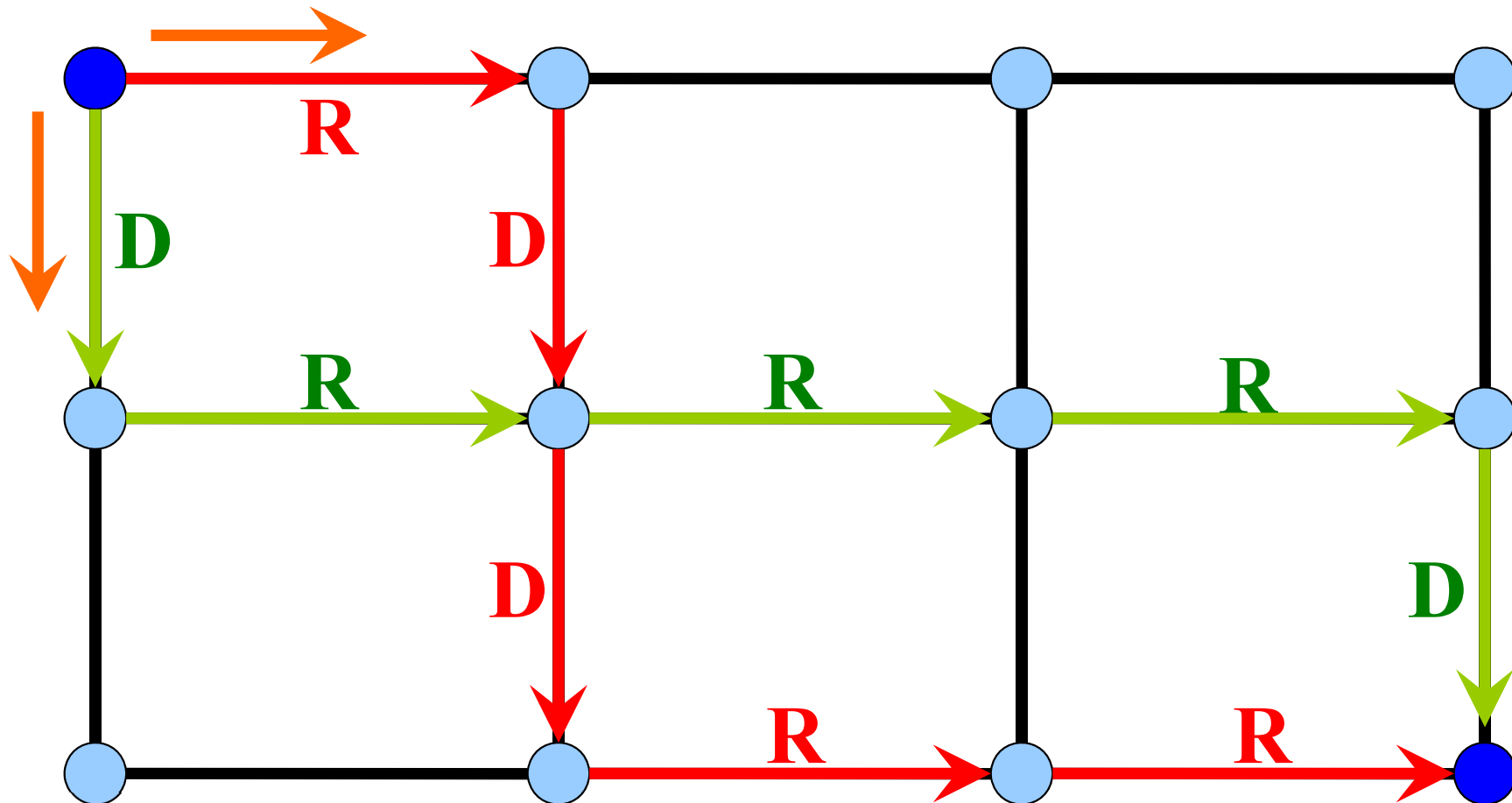
難しいなら易くすればいいのさ！

$$3+1+7+1+3 = \mathbf{15}$$

$$2+4+9+7+5 = \mathbf{27}$$



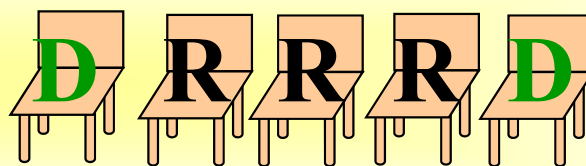
さて、経路は全部で幾つあるのか？



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



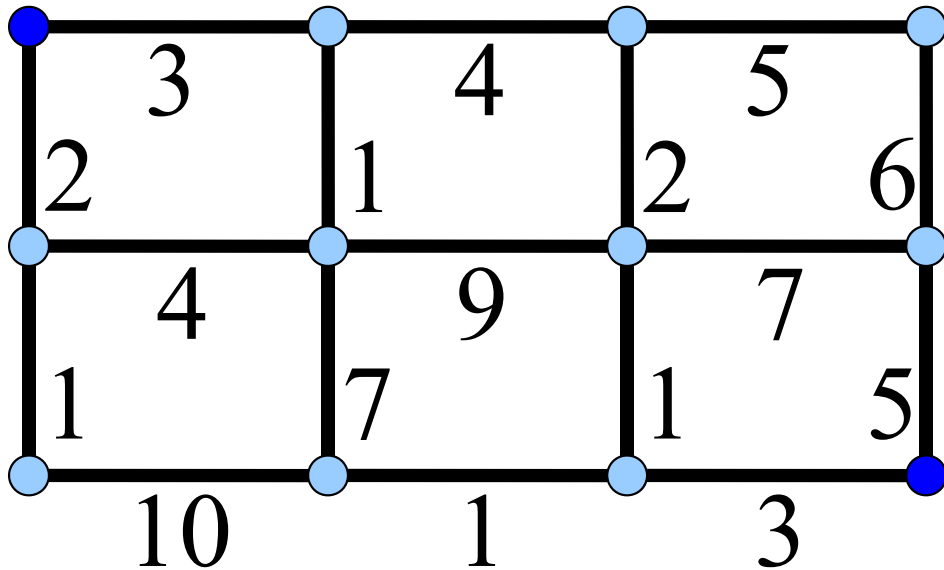
$$= \frac{(R+D)!}{R!D!} \text{通り}$$

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

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

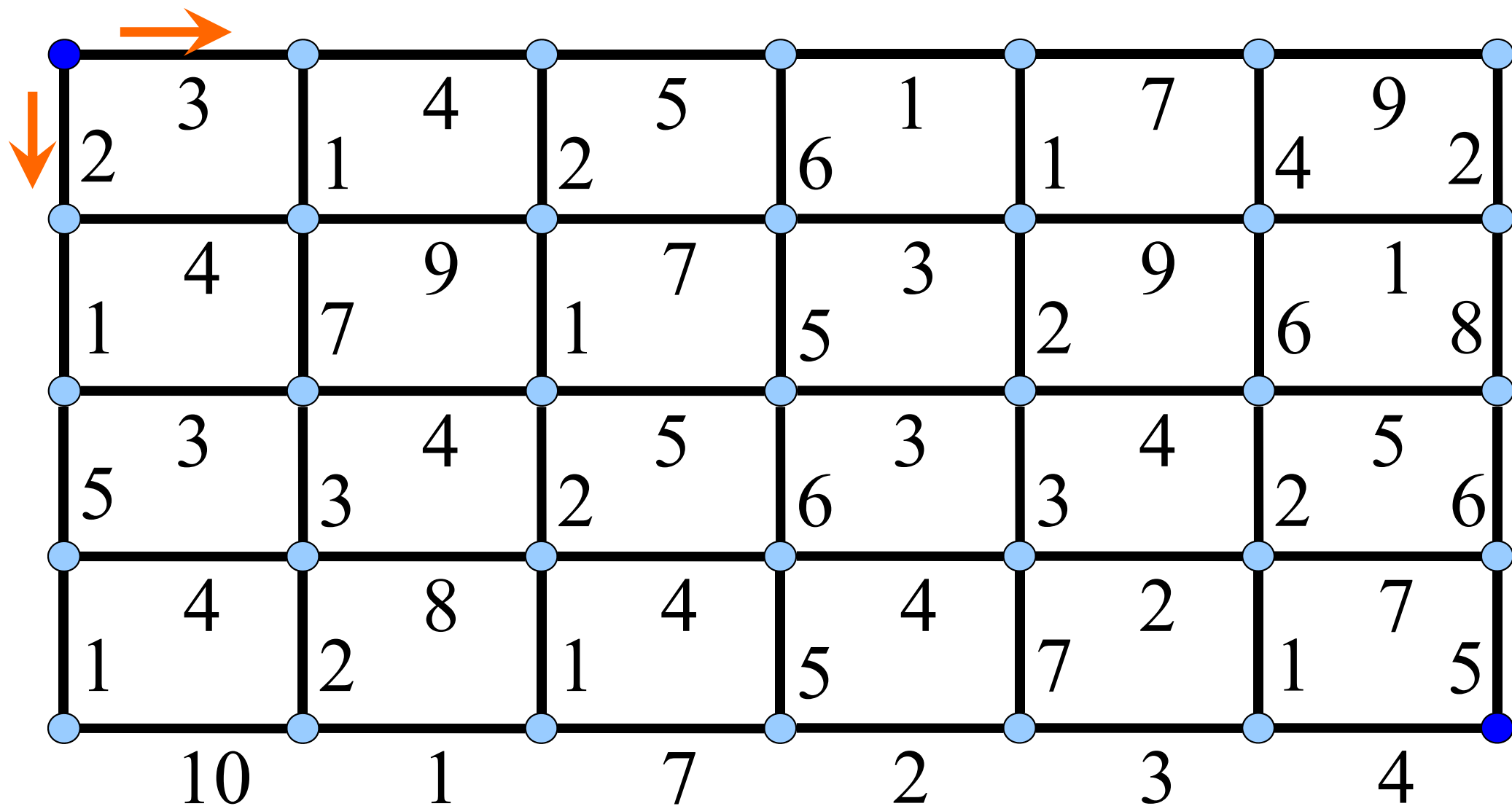


- **A**: 全列挙したよ ①～⑩

の10通り計算し⑧が最短だ！

- 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 **RRRRD**: $3+4+5+6+5=23$

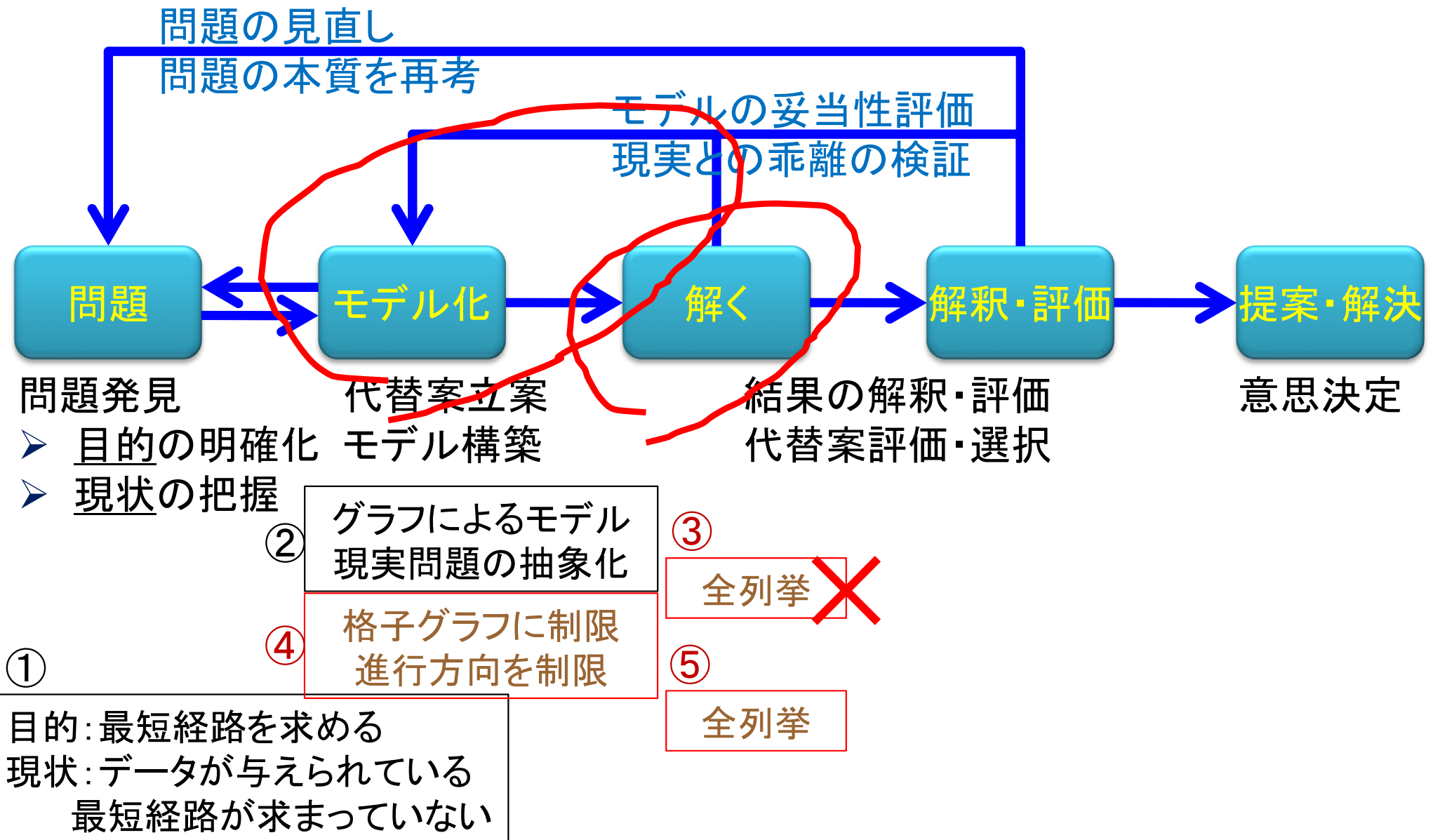
経路は全部で幾つ？



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

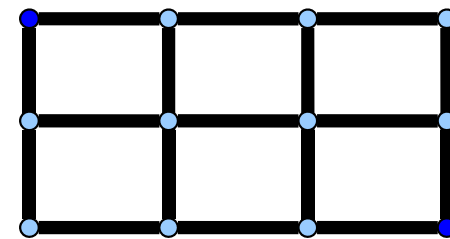
問題解決とは？

➤ 問題発見・問題解決から意思決定まで



経路は全部で幾つ？【全列举】

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？【全列举】

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) : **51.2 GFLOPS** ...1秒間に**512億**回
 - PS3 : **218 GFLOPS** ...1秒間に**2180億**回
 - PS4 : **1.84 TFLOPS** ...1秒間に**1兆8400億**回
 - PS5 : **10.28 TFLOPS** ...1秒間に**10兆2800億**回
 - 京 : **10.51 PFLOPS** ...1秒間に**1京510兆**回
 - 富岳 : **415.53 PFLOPS** ...1秒間に**41京5530兆**回

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

(※京: Top500.org 世界最速[2回] 2011年6,11月)

(※富岳: Top500.org 世界最速[4回] 2020年6月～2021年11月)]

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

K(キロ) $\simeq \times 10^3$ = 千倍

M(メガ) $\simeq \times 10^6$ = 百万倍

G(ギガ) $\simeq \times 10^9$ = 10億倍

T(テラ) $\simeq \times 10^{12}$ = 1兆倍

P(ペタ) $\simeq \times 10^{15}$ = 千兆倍

E(エクサ) $\simeq \times 10^{18}$ = 百京倍

経路は全部で幾つ？【全列举】

10.28 TFLOPS 415.53 PFLOPS

R(横) D(縦)	全経路	PS5	富岳
3 2	10	0.0000000000 秒	0.0000000000 秒
6 4	210	0.0000000000 秒	0.0000000000 秒
10 5	3,003	0.0000000004 秒	0.0000000000 秒
20 10	30,045,015	0.000087680 秒	0.0000000002 秒
25 25	1.3×10^{14}	10.25 分	0.015210768 秒
30 30	1.2×10^{17}	7.99 日	17.076684942 秒
40 40	1.1×10^{23}		
50 50	1.0×10^{29}		
100 100	9.1×10^{58}		
500 500	2.7×10^{299}		

経路は全部で幾つ？【全列举】

10.28 TFLOPS

415.53 PFLOPS

R(横) D(縦)	全経路	PS5	富岳
3 2	10	0.0000000000 秒	0.0000000000 秒
6 4	210	0.0000000000 秒	0.0000000000 秒
10 5	3,003	0.0000000004 秒	0.0000000000 秒
20 10	30,045,015	0.000087680 秒	0.0000000002 秒
25 25	1.3×10^{14}	10.25 分	0.015210768 秒
30 30	1.2×10^{17}	7.99 日	17.076684942 秒
40 40	1.1×10^{23}	26,529.43 年	239.56 日
50 50	1.0×10^{29}	2.26 宙齡	769,918.79 年
100 100	9.1×10^{58}	4.05×10^{30} 宙齡	1.00×10^{26} 宙齡
500 500	2.7×10^{299}	6.04×10^{271} 宙齡	1.49×10^{267} 宙齡

圧倒的な計算力をもつコンピュータ
ですら、**全列举(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



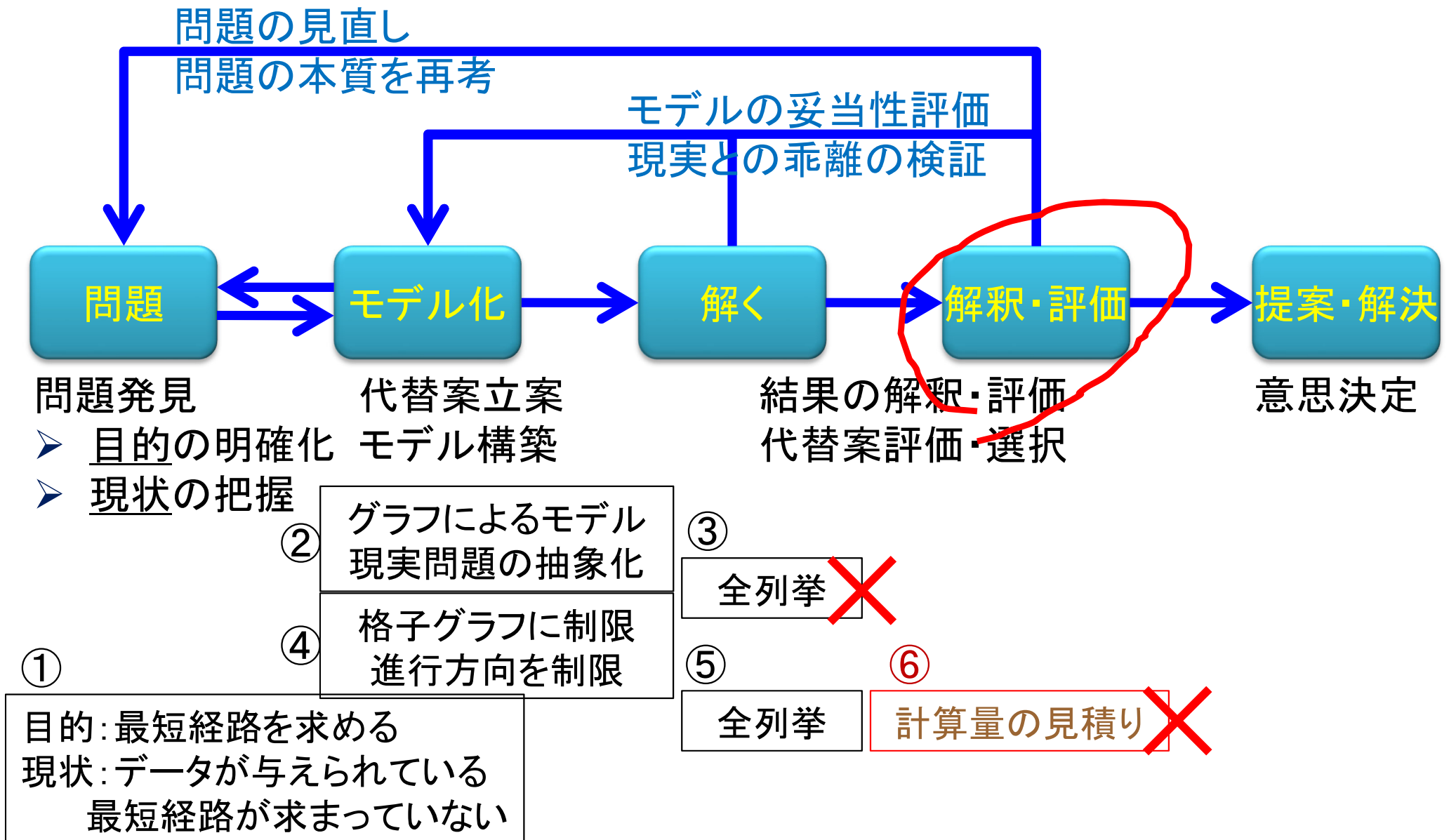
参考：大きい数を表す接頭辞

- 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) $\times 10^{20}$
- 秭(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】「無量大数」は「無限大 ∞ 」とは違う

問題解決とは？

➤ 問題発見・問題解決から意思決定まで



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



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: 各点 v に距離ラベル $d(v)$ と親ラベル $p(v)$ を設定する

✓ 始点 0 について $d(0)=0$, $p(0)=-1$

✓ その他の点 v について $d(v)=\infty$, $p(v)=-1$

未確定点集合 $N=\{0,1,2,\dots,11\}$ とする

(更新法)

step1-1: 未確定点 $v \in N$ について $d(v)$ が最小の点 を見つける

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$ + 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、もし $d(v)+c(e) < d(u)$ なら、枝先点 u の距離ラベル $d(u)$ と 親ラベル $p(v)$ の値を以下のように更新する

✓ $d(u) := d(v) + c(e)$ 即ち、 $d(u)$ の値 を $d(v) + c(e)$ の値にする

✓ $p(u) := v$ 即ち、 $p(u)$ の値 を v にする

step1-3: その点 v から出る全枝 $e \in E$ の作業が全て終了したら、その点 $v \in N$ を未確定点集合 N から除去する

(終了判定)

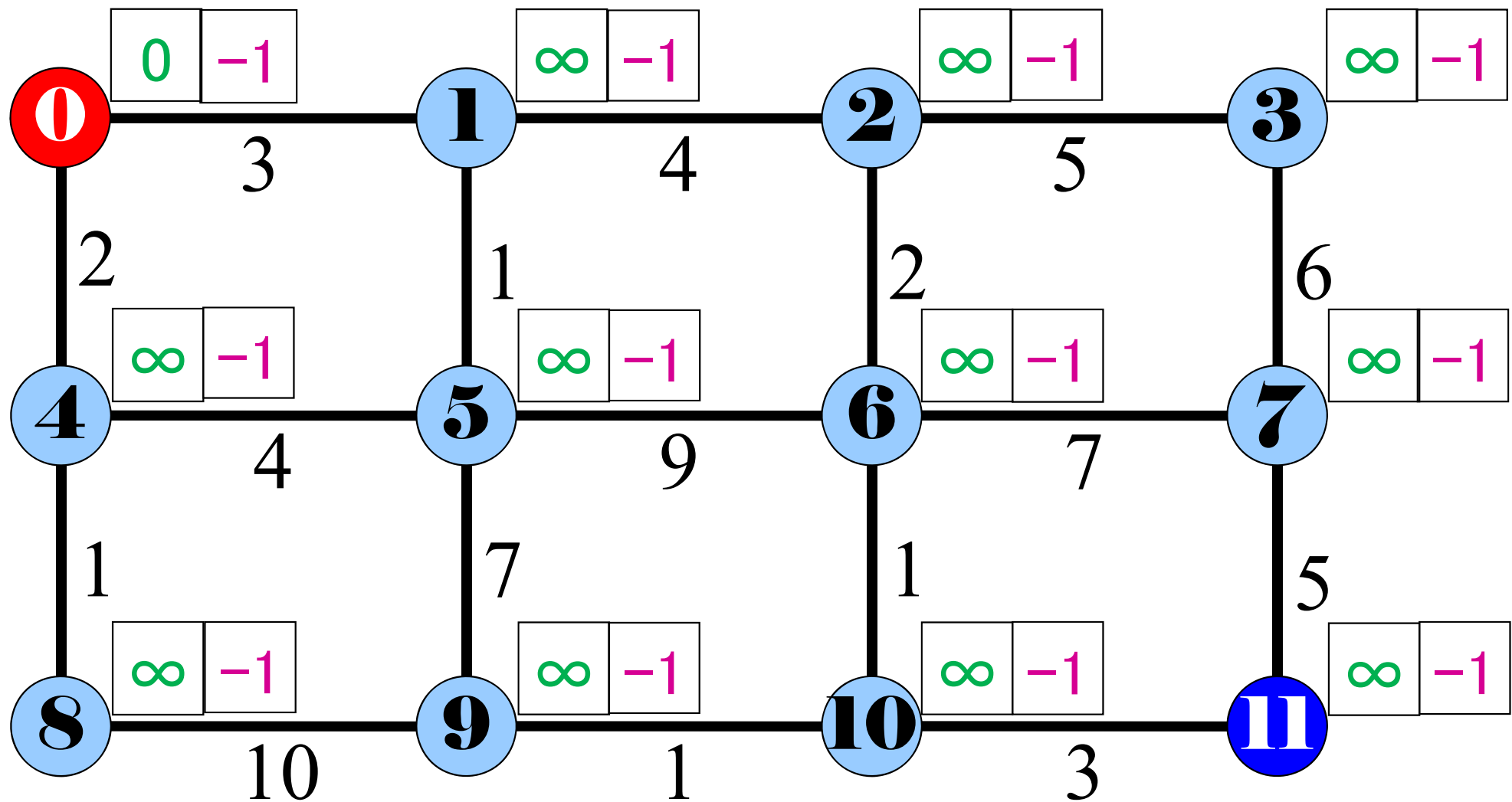
step2: 未確定点集合 N が空(くう) ($N=\emptyset$) になったら終了. そうでなければ step1-1 へ戻る

Dijkstra法 (初期設定)

step0: 各点 v に距離ラベル $d(v)$ と親ラベル $p(v)$ を設定する

- ✓ 始点 s について $d(s)=0$, $p(s)=-1$
- ✓ その他の点 v について $d(v)=\infty$, $p(v)=-1$

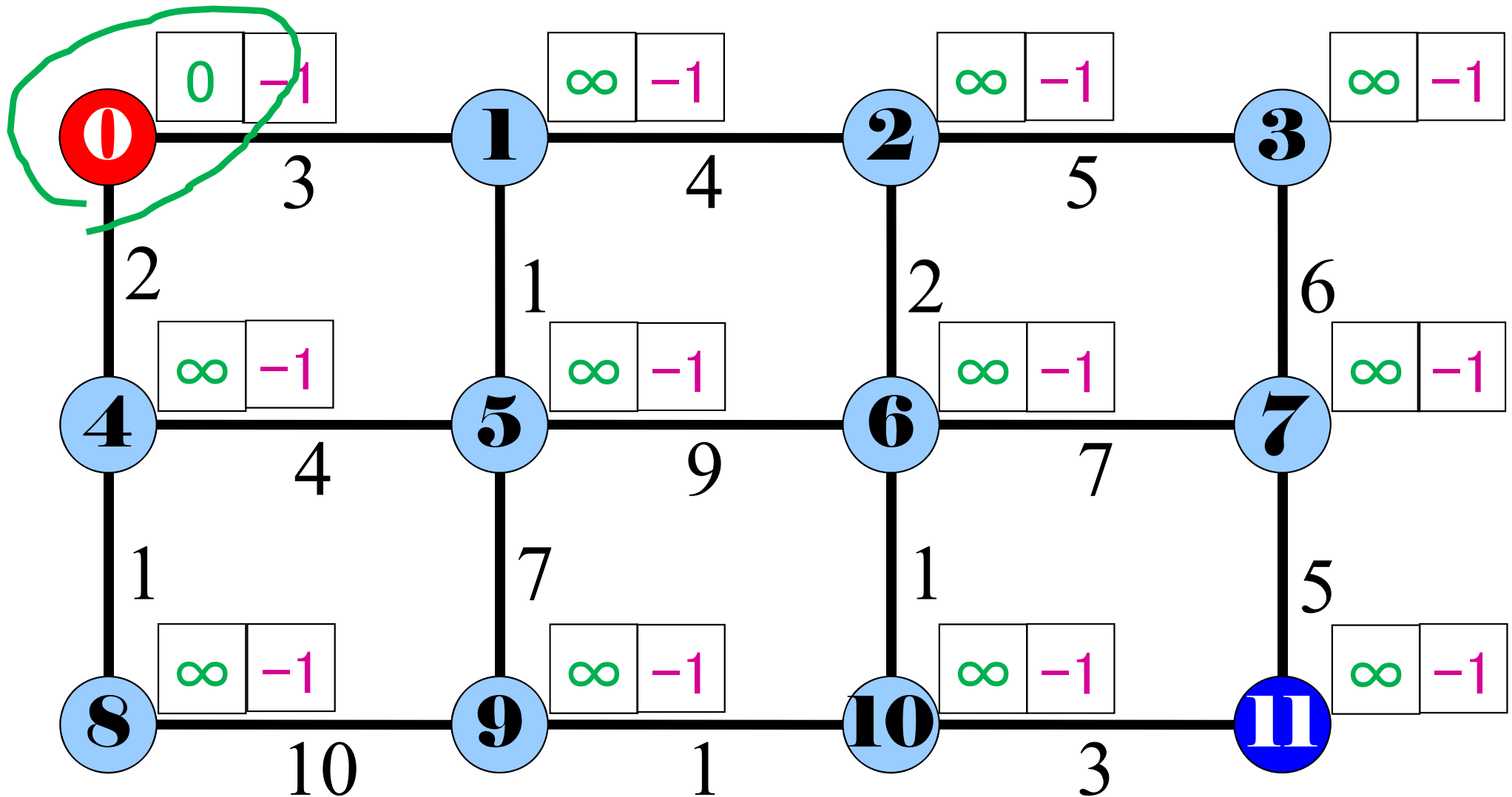
未確定点集合 $N=\{0,1,2,\dots,11\}$ とする



Dijkstra法 (更新法)

step1-1: 未確定点 $v \in N$ について $d(v)$ が最小の点を見つける

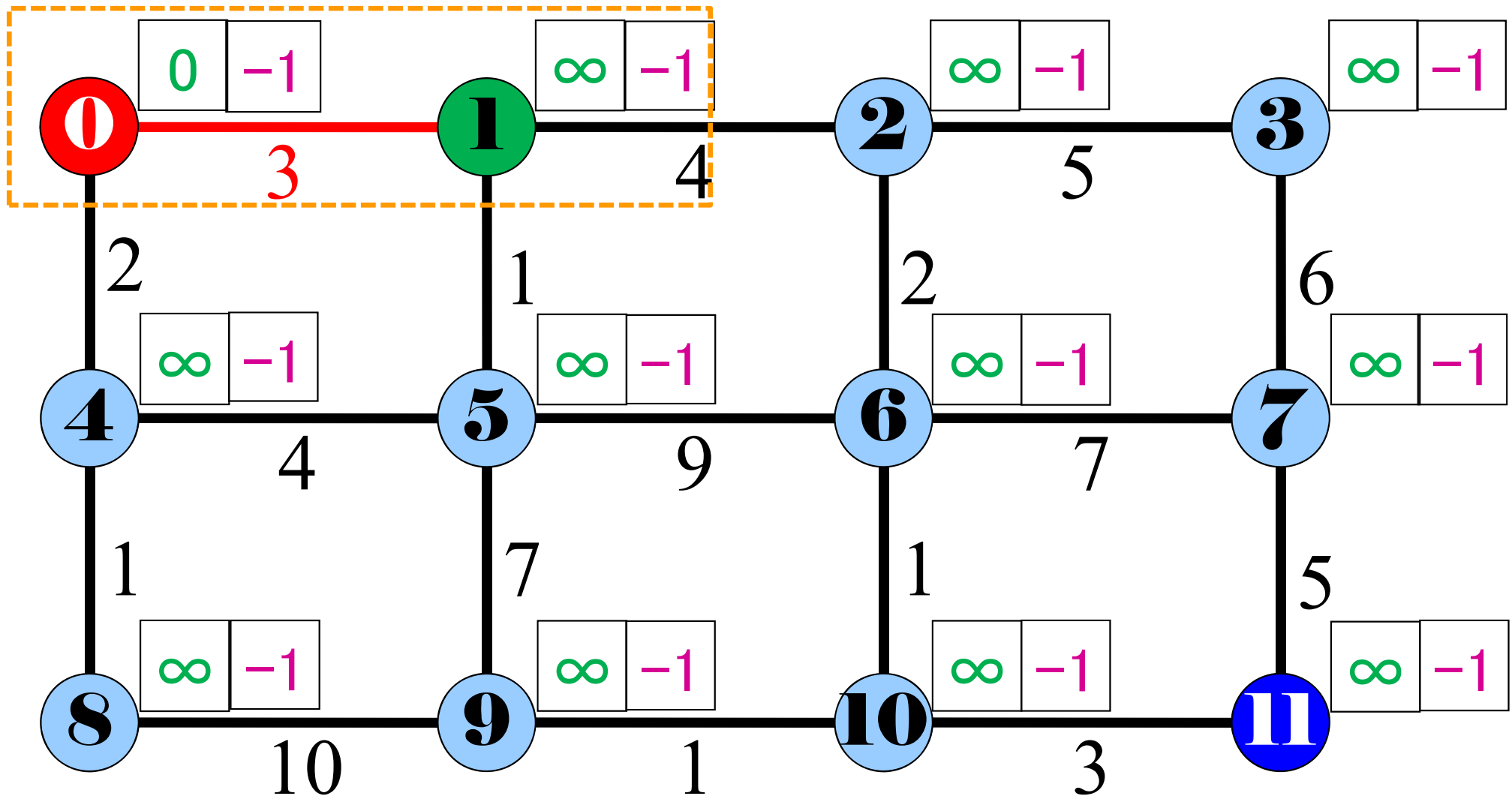
未確定点集合 $N = \{0, 1, 2, \dots, 11\}$ で $d(v)$ が最小の点 0 を見つけた



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$ + 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

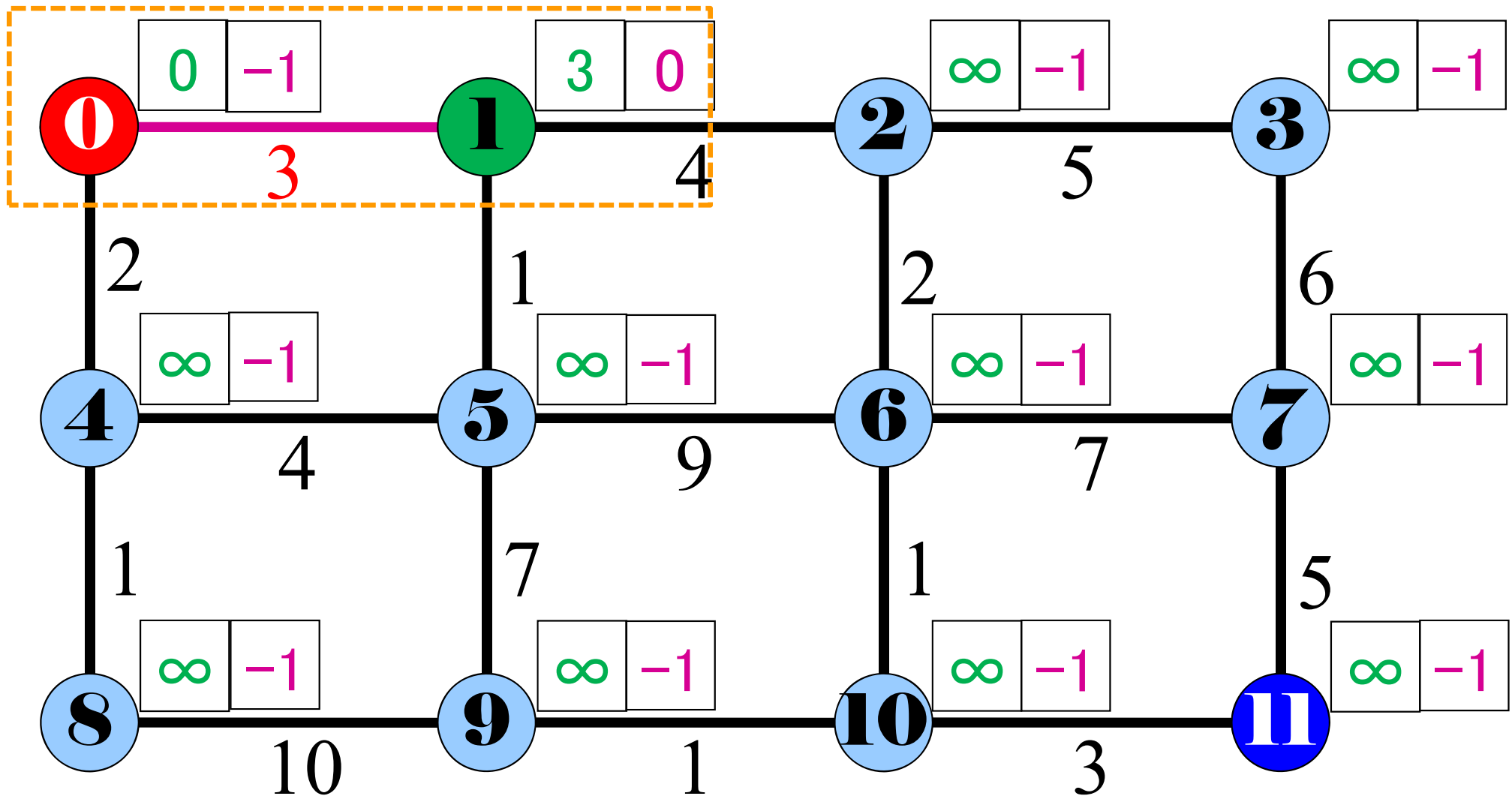
枝 e_{01} は $d(0)+c(e_{01}) = 0+3 = 3 < \infty = d(1)$ より, $d(1) := 3$, $p(1) := 0$ に



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$ + 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

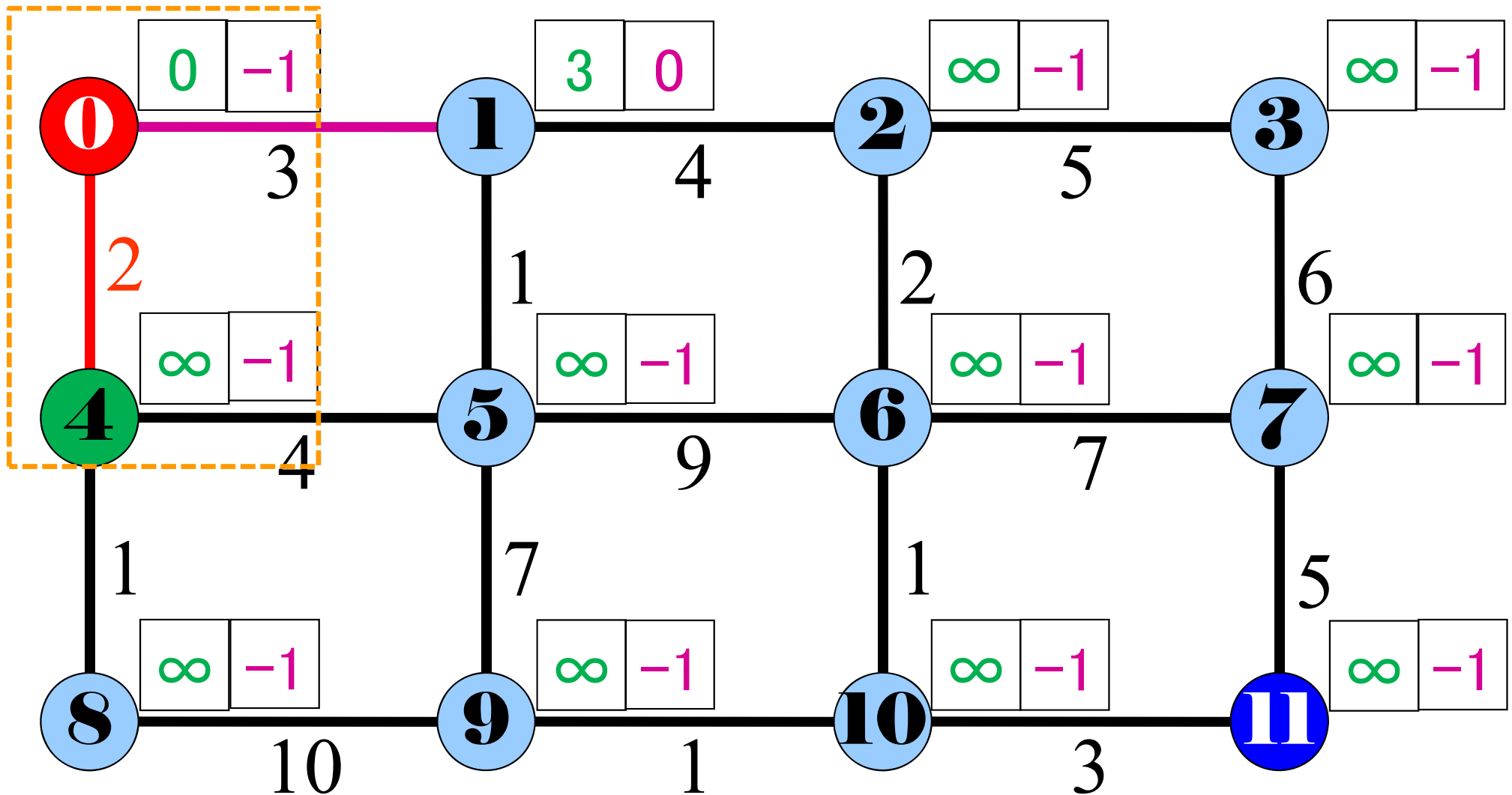
枝 e_{01} は $d(0)+c(e_{01}) = 0+3 = 3 < \infty = d(1)$ より, $d(1) := 3$, $p(1) := 0$ に



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$ + 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、もし $d(v) + c(e) < d(u)$ なら、 $d(u) := d(v) + c(e)$, $p(u) := v$ とする

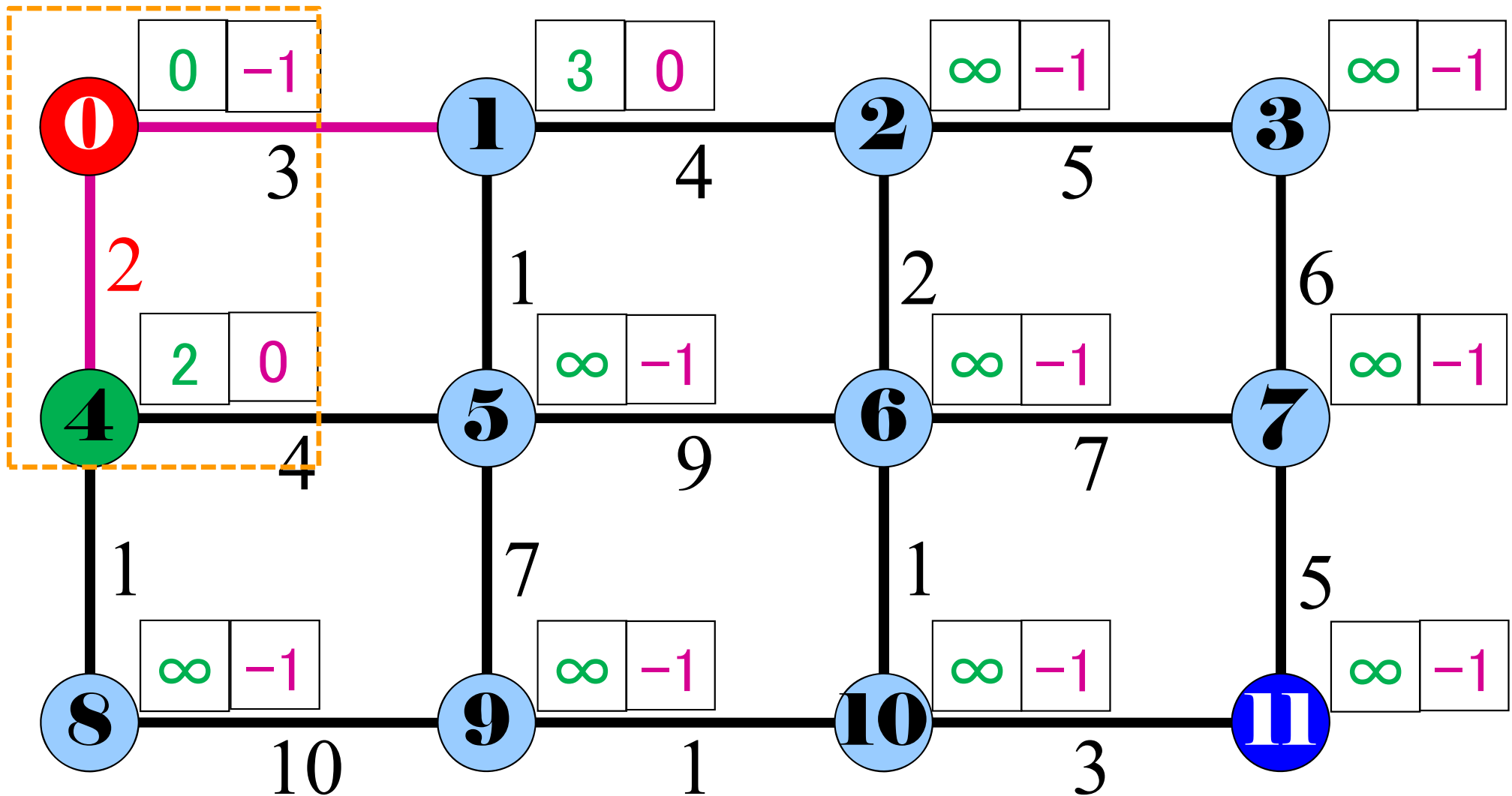
枝 e_{04} は $d(0)+c(e_{04}) = 0+2 = 2 < \infty = d(4)$ より, $d(4) := 2$, $p(4) := 0$ に



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$ + 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

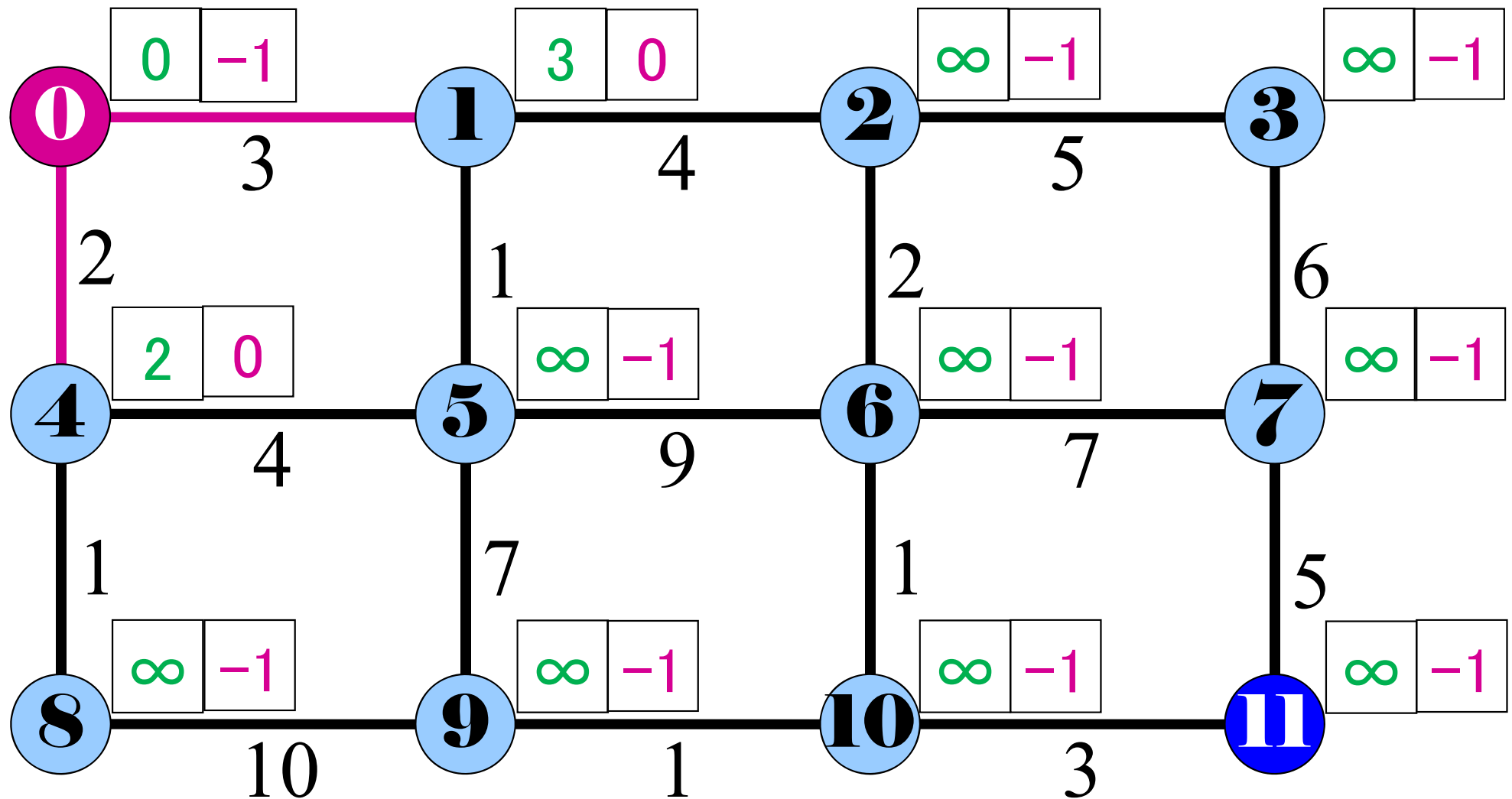
枝 e_{04} は $d(0)+c(e_{04}) = 0+2 = 2 < \infty = d(4)$ より, $d(4) := 2$, $p(4) := 0$ に



Dijkstra法 (更新法)

step1-3: その点 v から出る全枝 $e \in E$ の作業が全て終了したら,
その点 $v \in N$ を未確定点集合 N から除去する

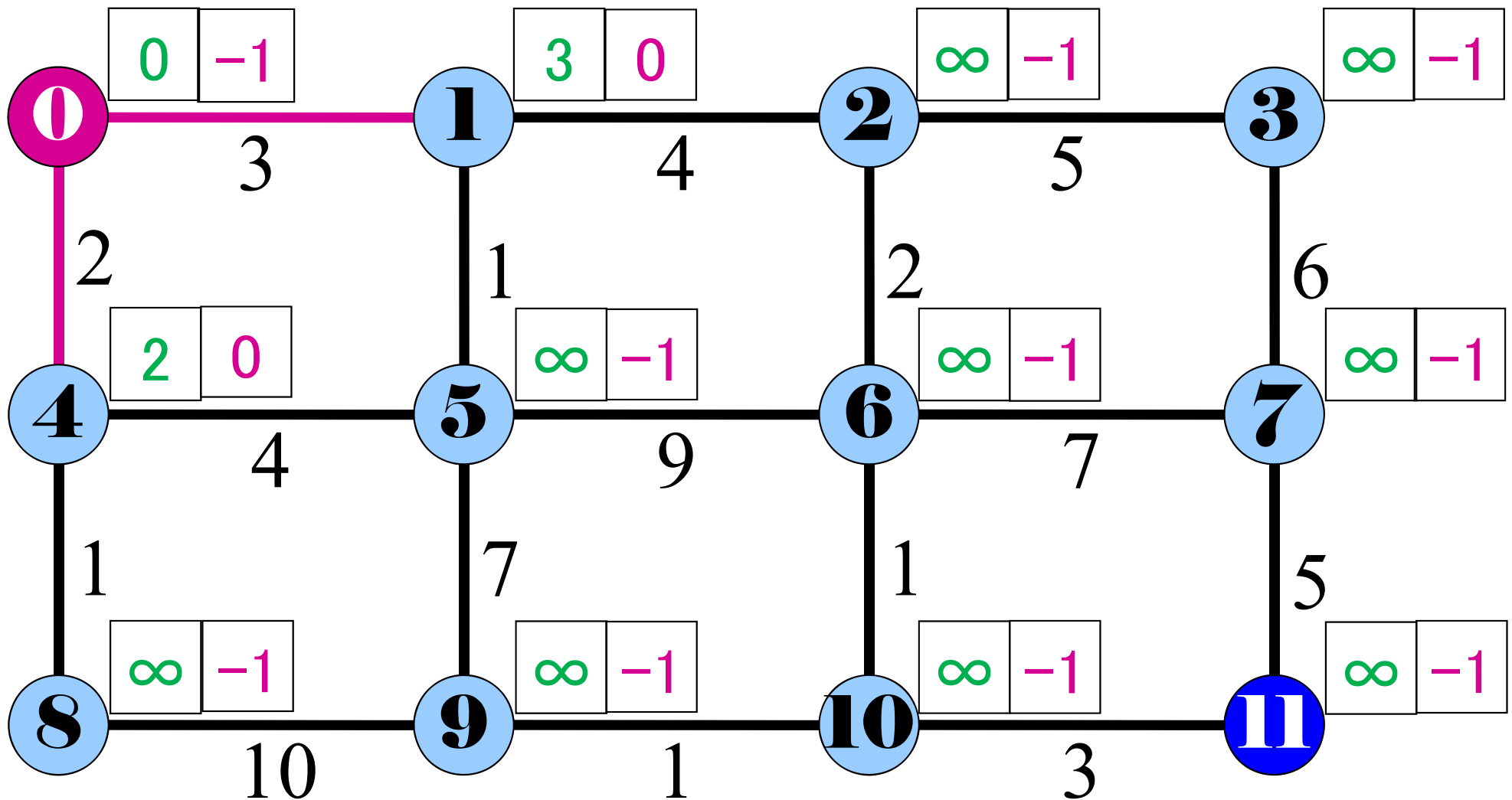
未確定点集合 $N = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$
 $\rightarrow N = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$



Dijkstra法 (終了判定)

step2:未確定点集合 N が空(くう) ($N=\emptyset$)になったら終了. そうでなければ step1-1 へ戻る

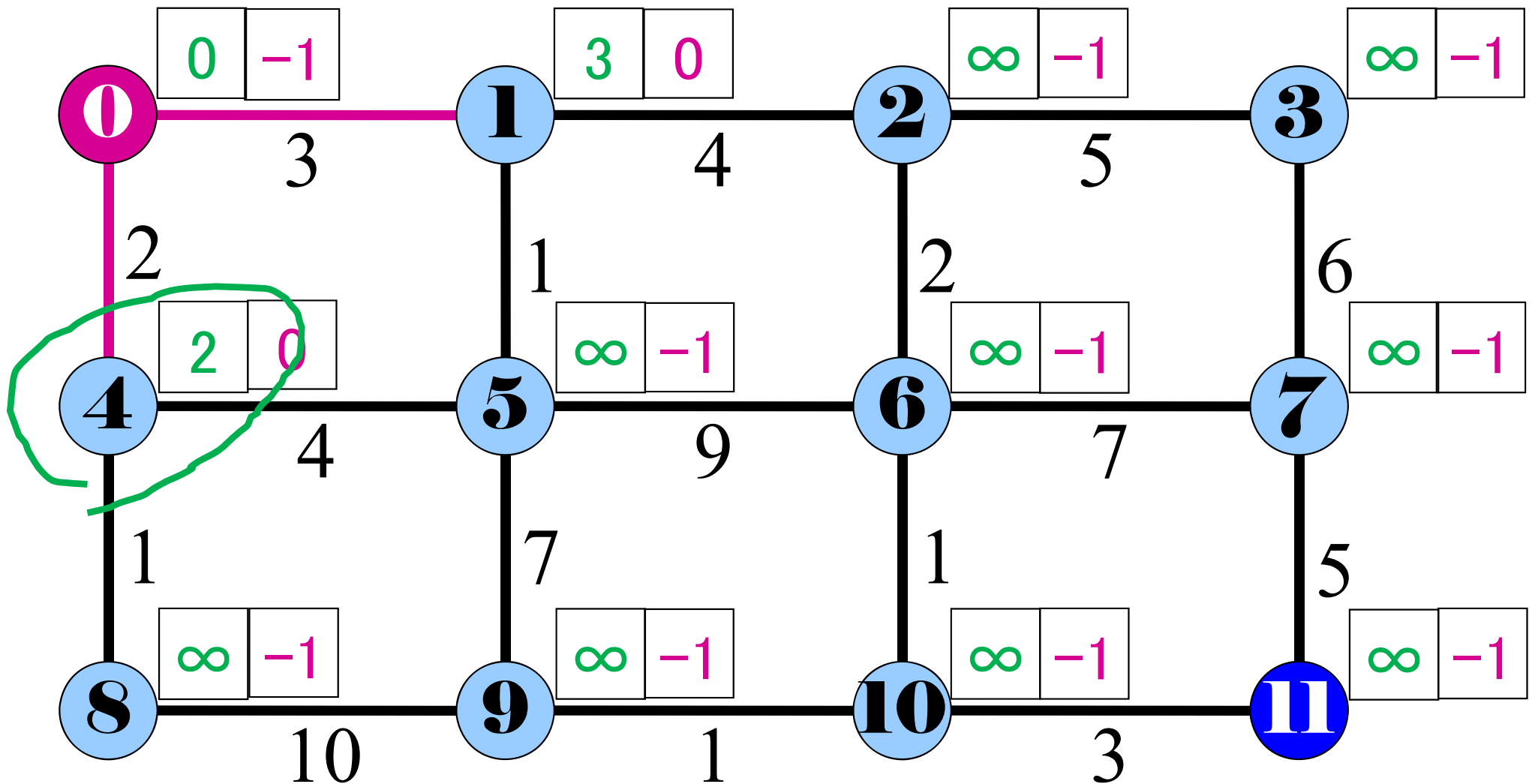
未確定点集合 $N=\{1,2,3,4,5,6,7,8,9,10,11\} \neq \emptyset$ より step1-1 へ戻る



Dijkstra法 (更新法)

step1-1: 未確定点 $v \in N$ について $d(v)$ が最小の点を見つける

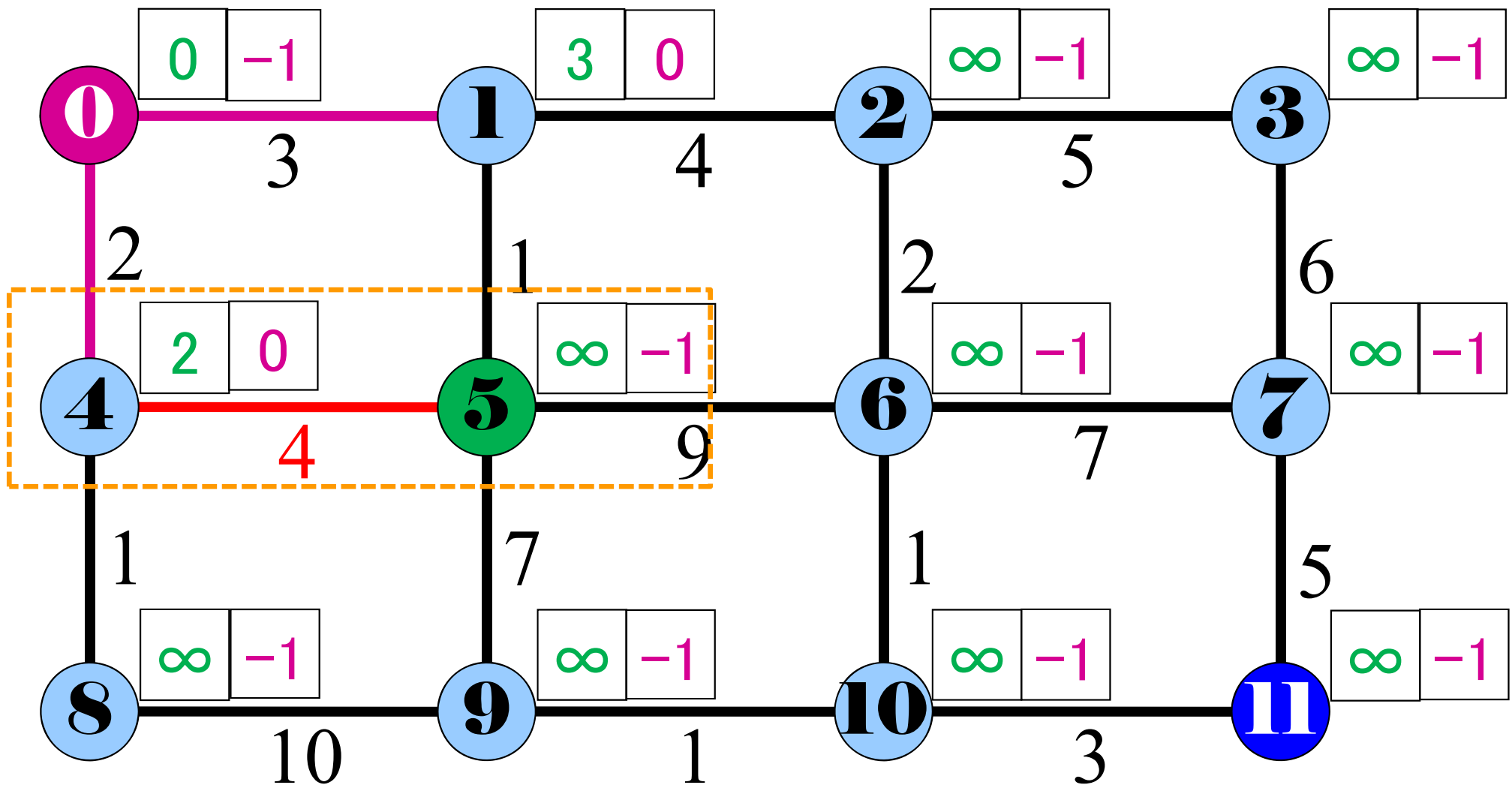
未確定点集合 $N = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$ で $d(\cdot)$ 最小点 4 を見つけた



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$
+ 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、
もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

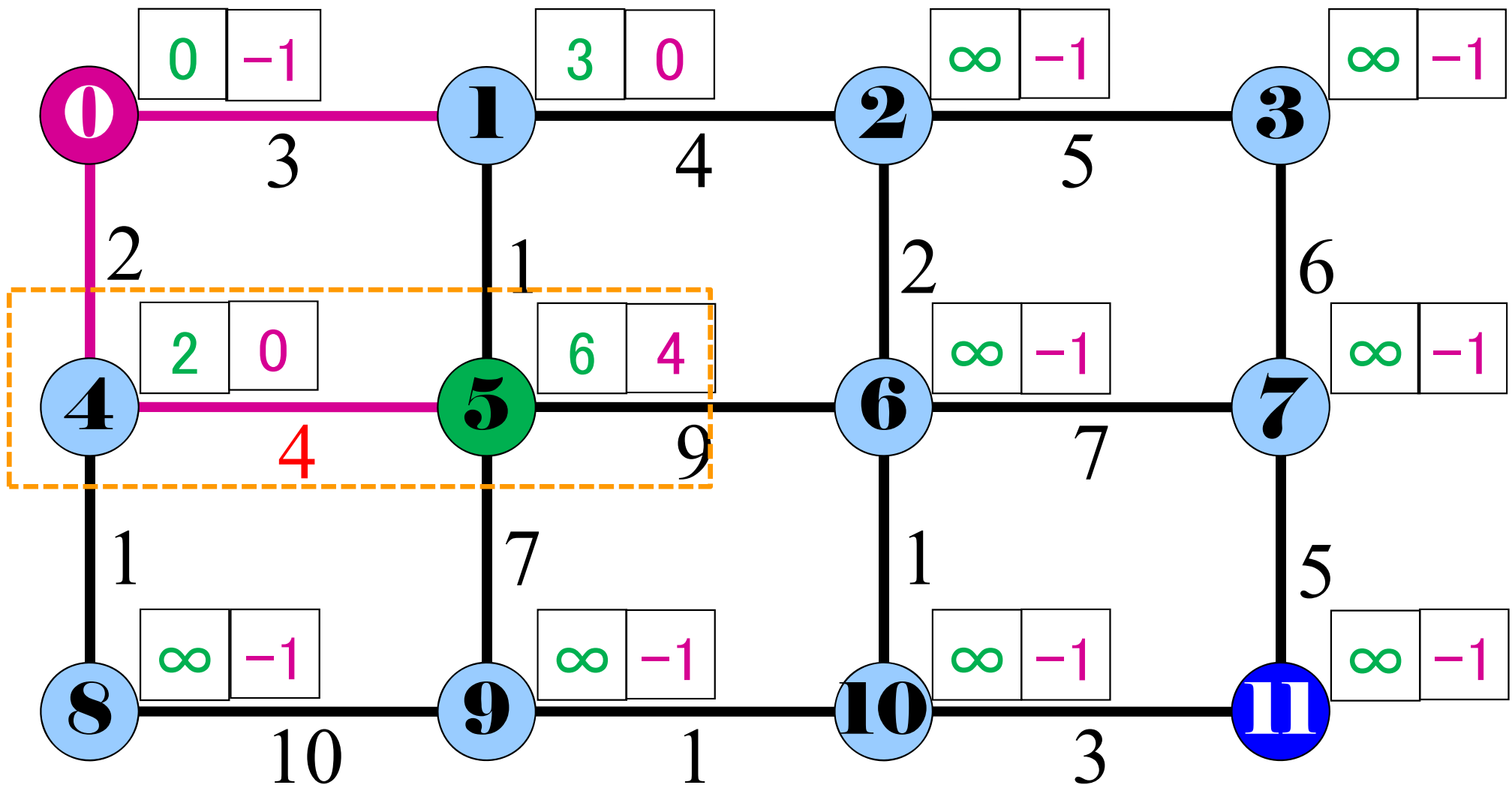
枝 e_{45} は $d(4)+c(e_{45}) = 2+4 = 6 < \infty = d(5)$ より、 $d(5) := 6$, $p(5) := 4$ に



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$
+ 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、
もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

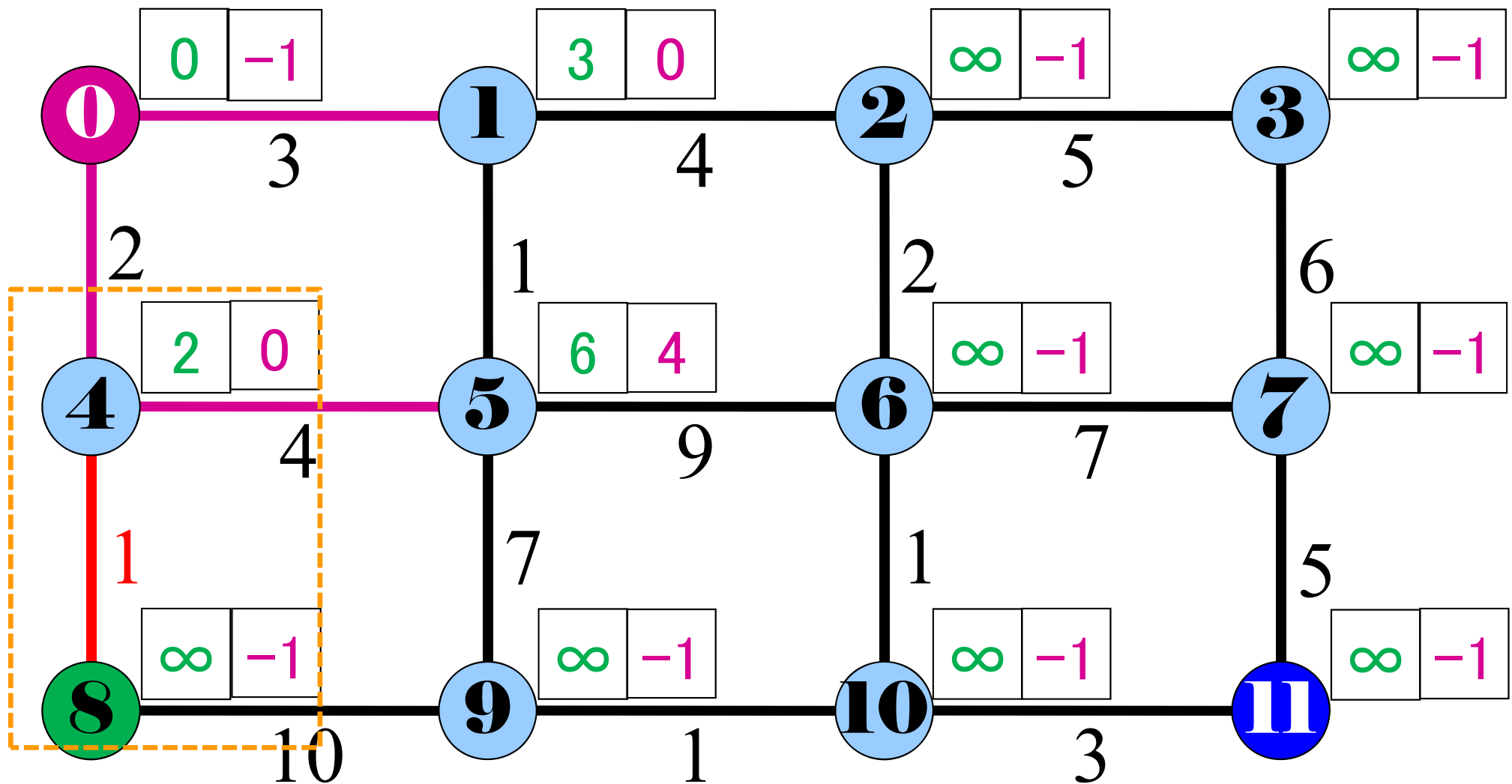
枝 e_{45} は $d(4)+c(e_{45}) = 2+4 = 6 < \infty = d(5)$ より、 $d(5) := 6$, $p(5) := 4$ に



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$
+ 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、
もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

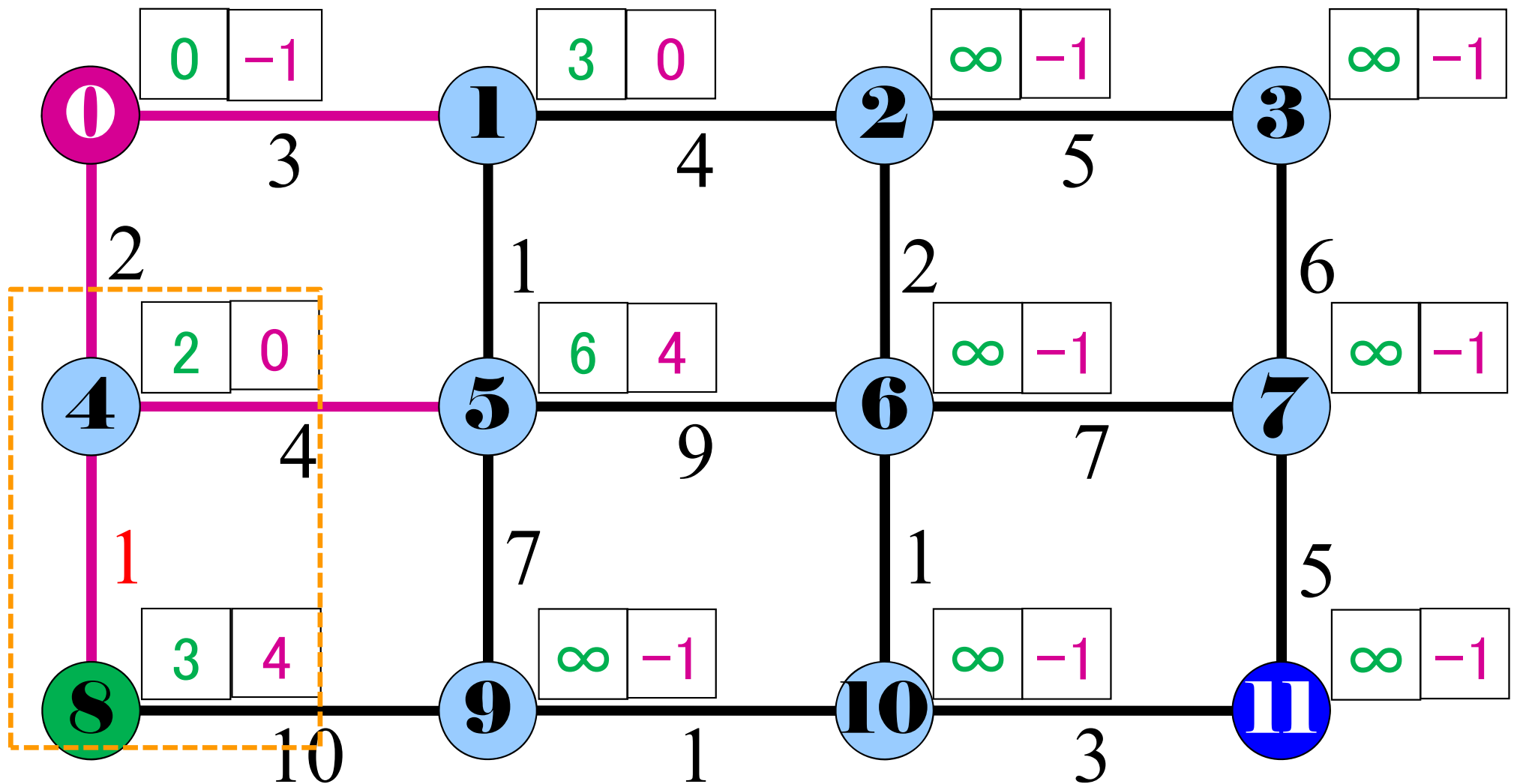
枝 e_{48} は $d(4)+c(e_{48}) = 2+1 = 3 < \infty = d(8)$ より, $d(8) := 3$, $p(8) := 4$ に



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$
+ 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、
もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

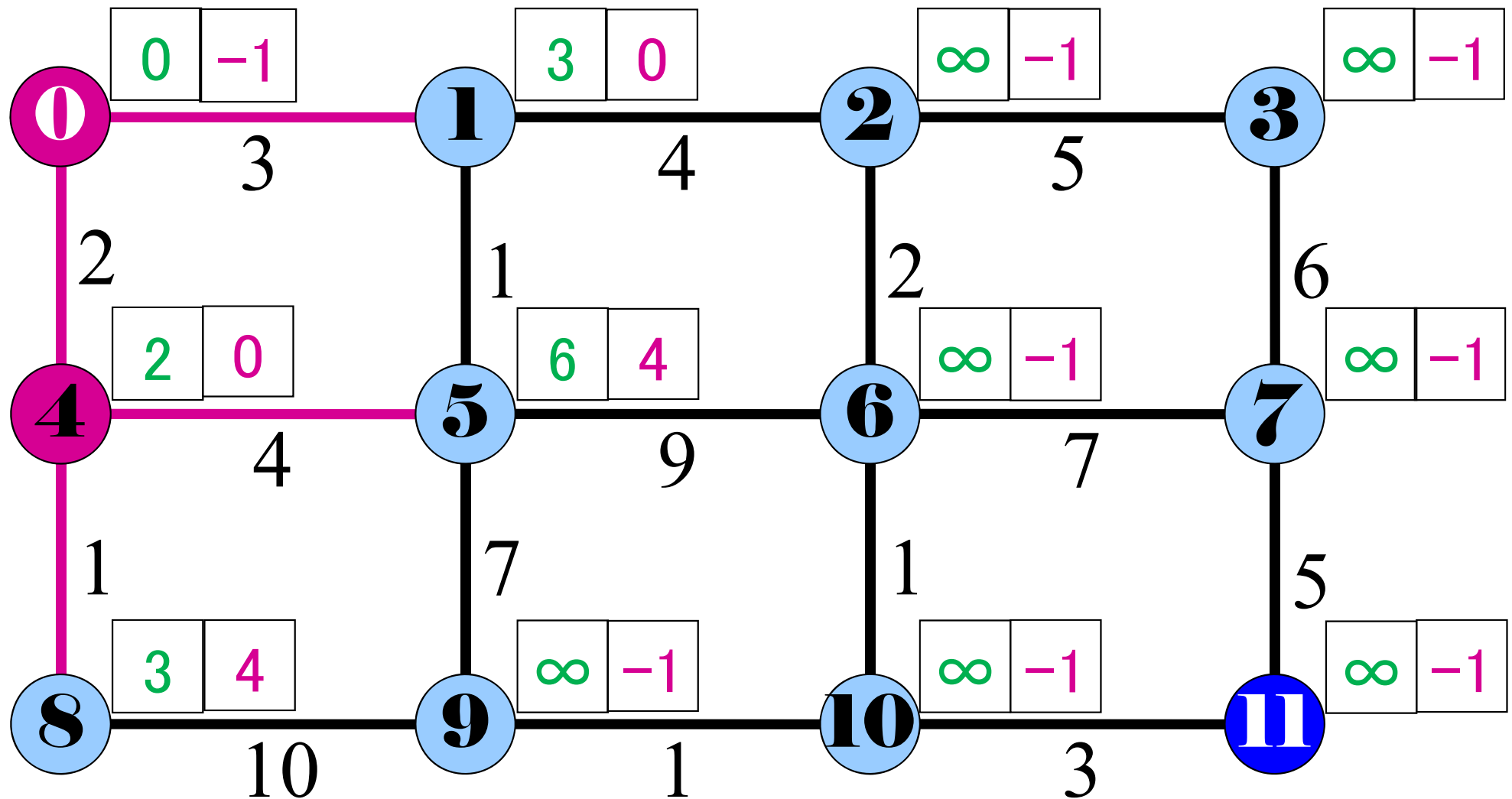
枝 e_{48} は $d(4)+c(e_{48}) = 2+1 = 3 < \infty = d(8)$ より, $d(8) := 3$, $p(8) := 4$ に



Dijkstra法 (更新法)

step1-3: その点 v から出る全枝 $e \in E$ の作業が全て終了したら,
その点 $v \in N$ を未確定点集合 N から除去する

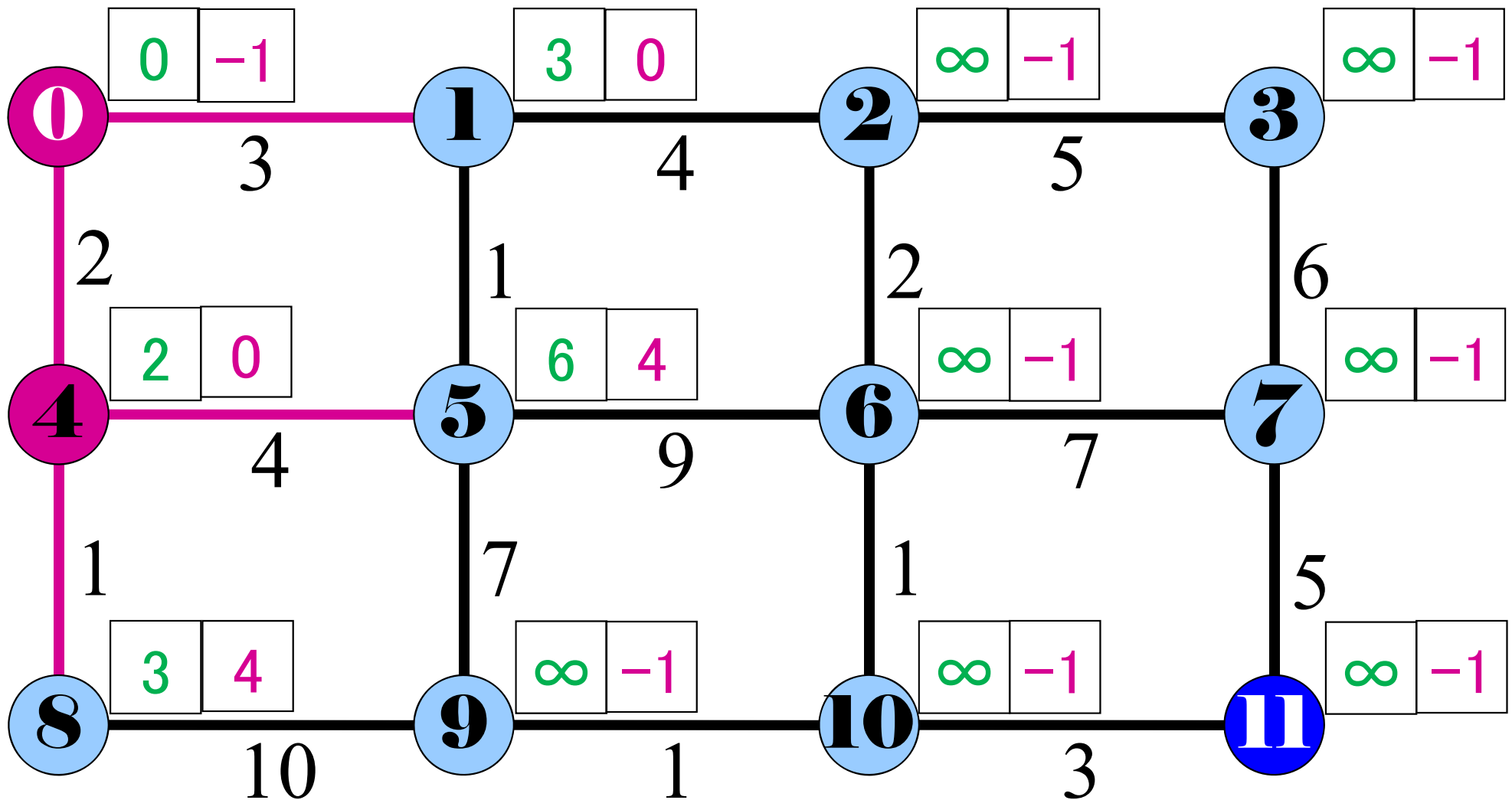
未確定点集合 $N = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$
 $\rightarrow N = \{1, 2, 3, 5, 6, 7, 8, 9, 10, 11\}$



Dijkstra法 (終了判定)

step2:未確定点集合 N が空(くう) ($N=\emptyset$)になったら終了. そうでなければ step1-1 へ戻る

未確定点集合 $N=\{1,2,3,5,6,7,8,9,10,11\} \neq \emptyset$ より step1-1 へ戻る

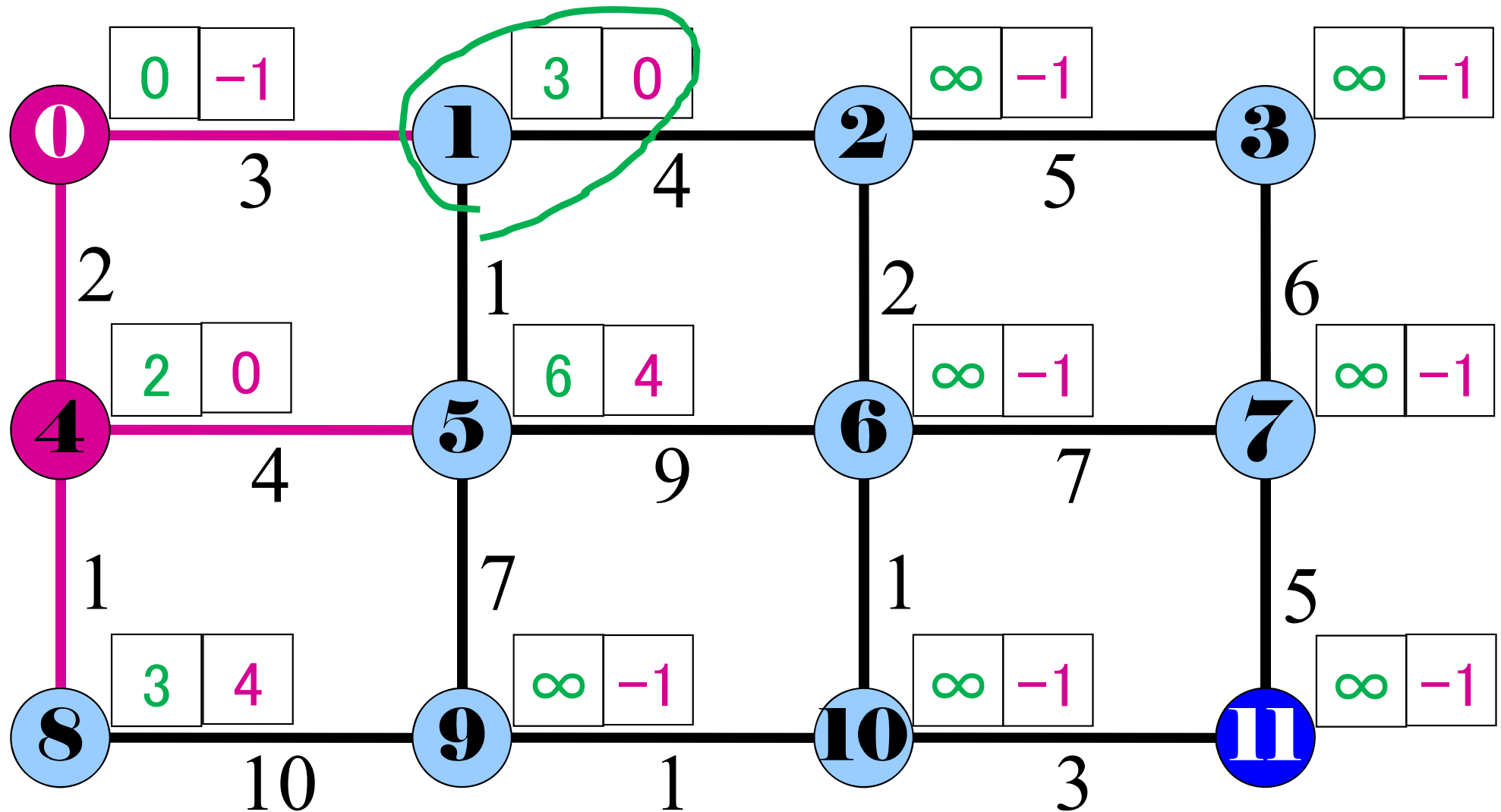


Dijkstra法 (更新法)

step1-1: 未確定点 $v \in N$ について $d(v)$ が最小の点を見つける

最小点が複数ある場合はどれでも良い

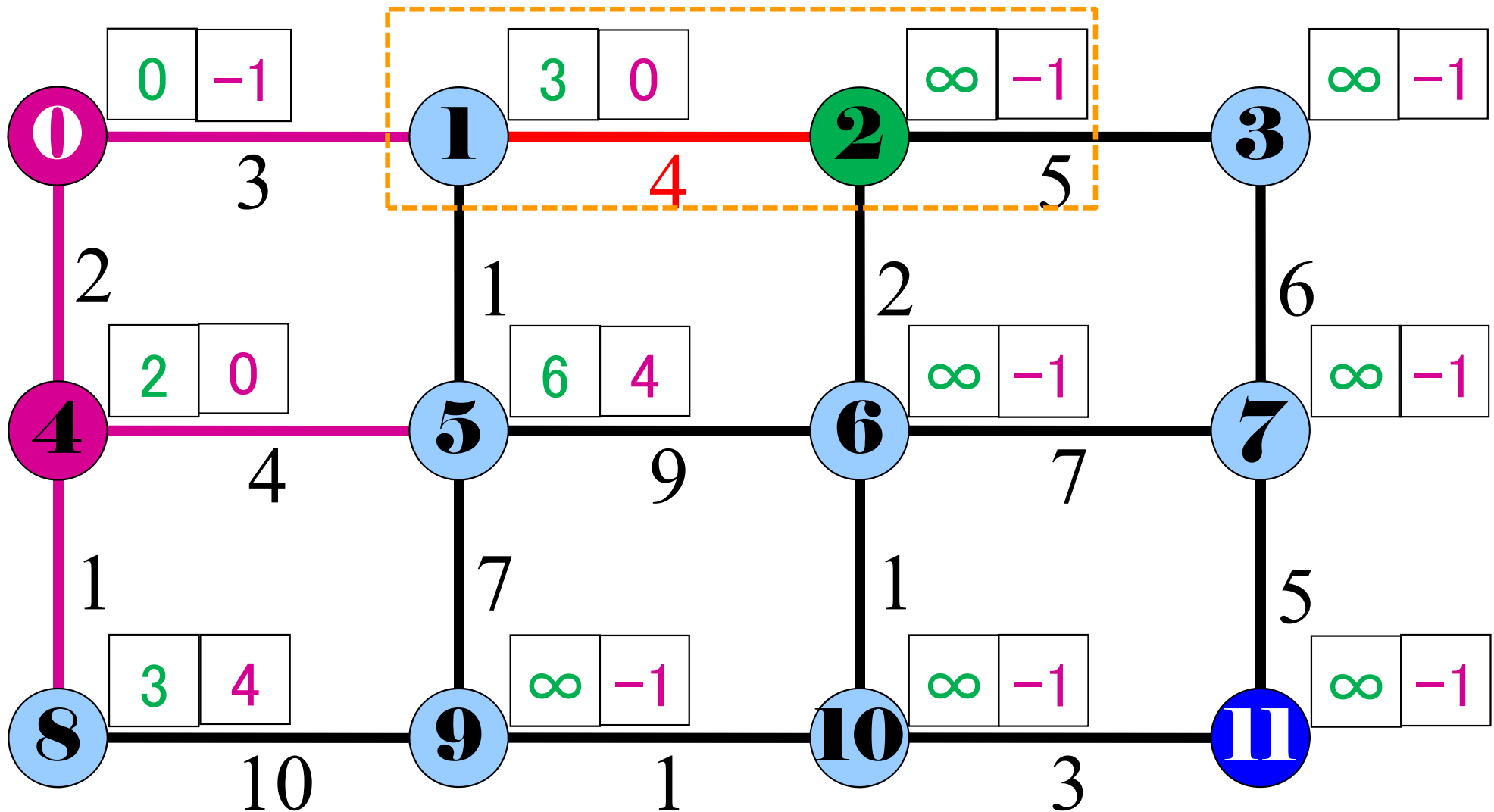
未確定点集合 $N = \{1, 2, 3, 5, 6, 7, 8, 9, 10, 11\}$ で $d(\cdot)$ 最小点 1 を見つけた



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$ + 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

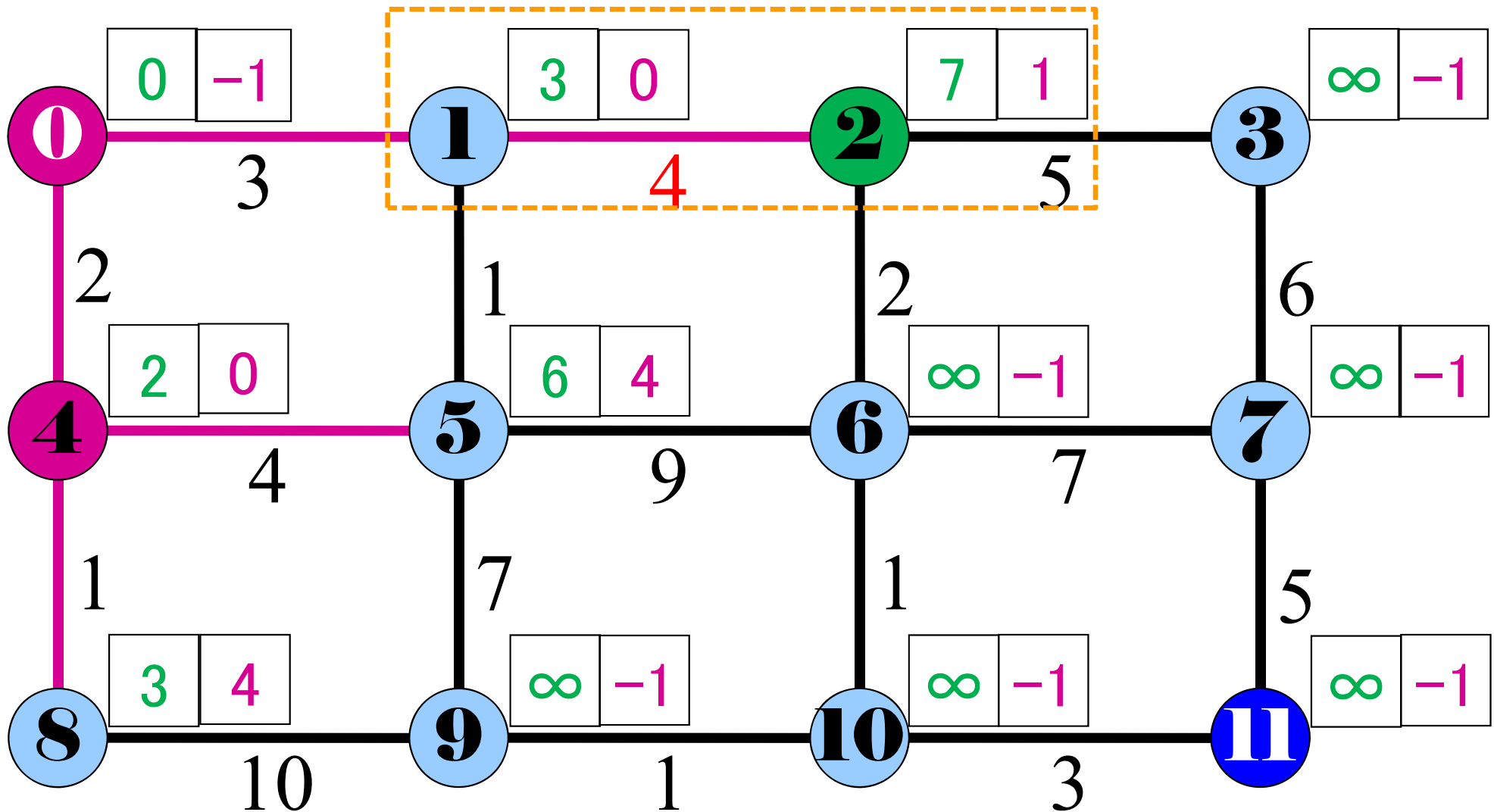
枝 e_{12} は $d(1)+c(e_{12}) = 3+4 = 7 < \infty = d(2)$ より, $d(2) := 7$, $p(2) := 1$ に



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$ + 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

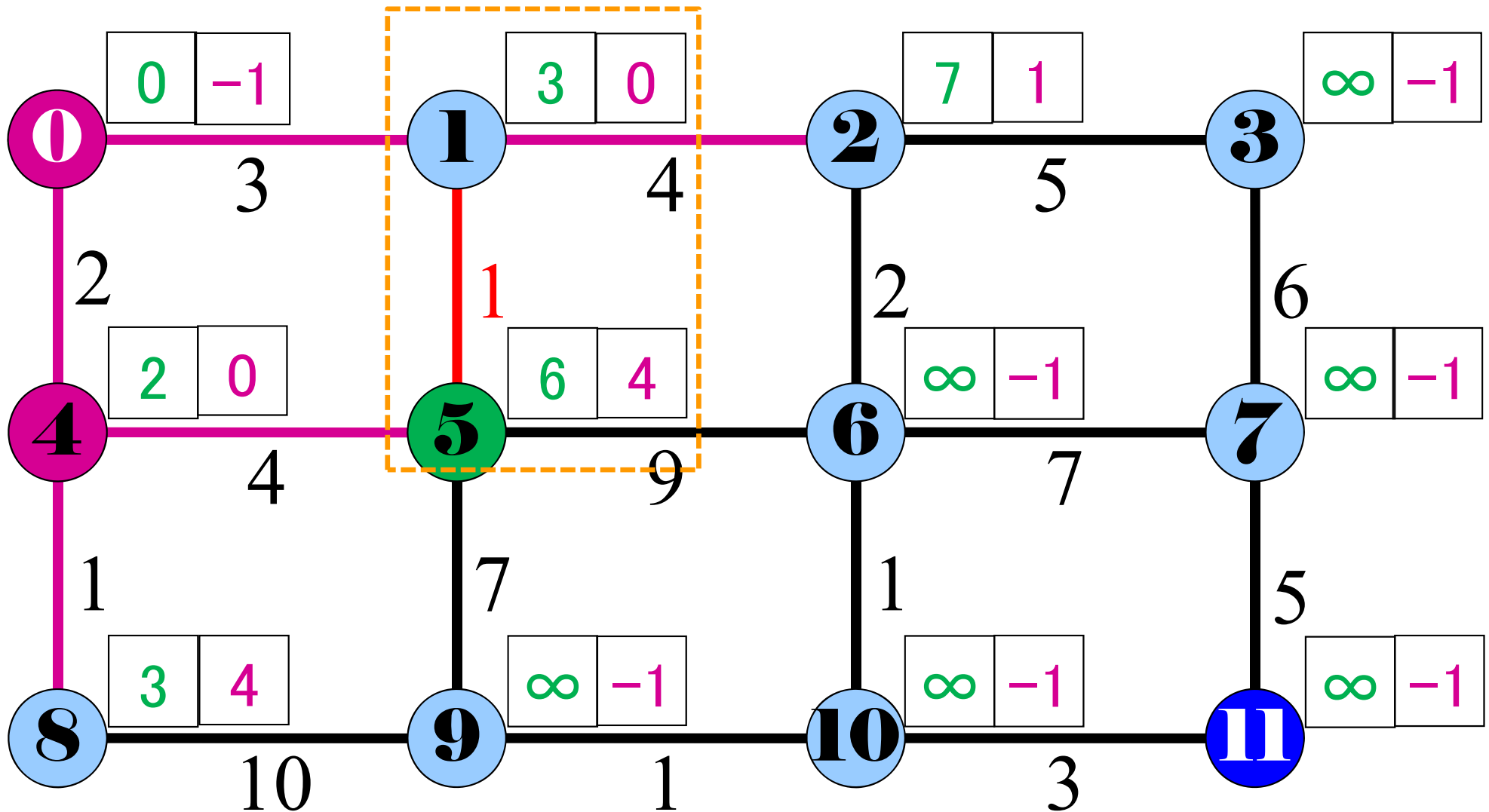
枝 e_{12} は $d(1)+c(e_{12}) = 3+4 = 7 < \infty = d(2)$ より, $d(2) := 7$, $p(2) := 1$ に



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$
+ 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、
もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

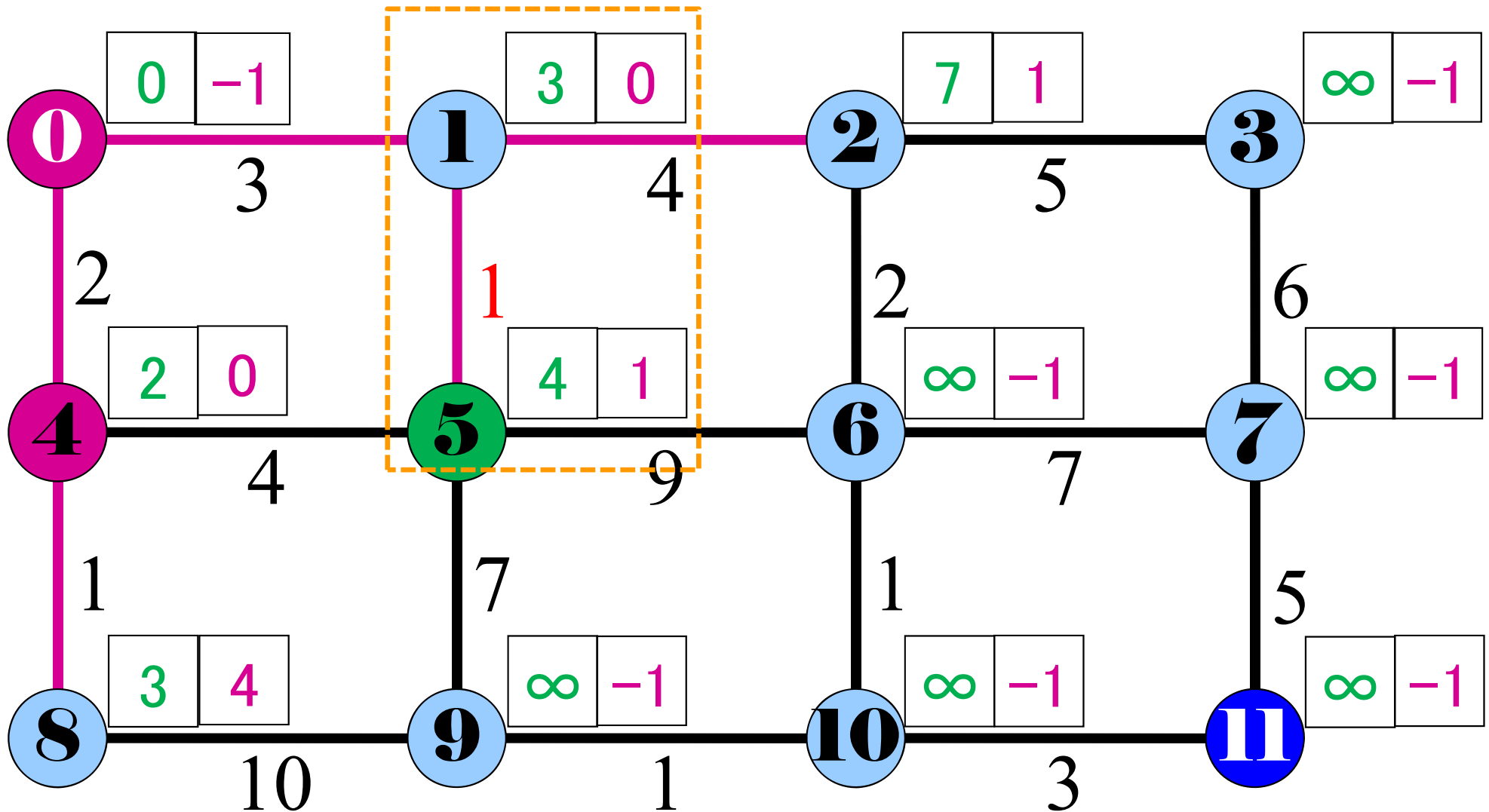
枝 e_{15} は $d(1)+c(e_{15}) = 3+1 = 4 < 6 = d(5)$ より、 $d(5) := 4$, $p(5) := 1$ に



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$
+ 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、
もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

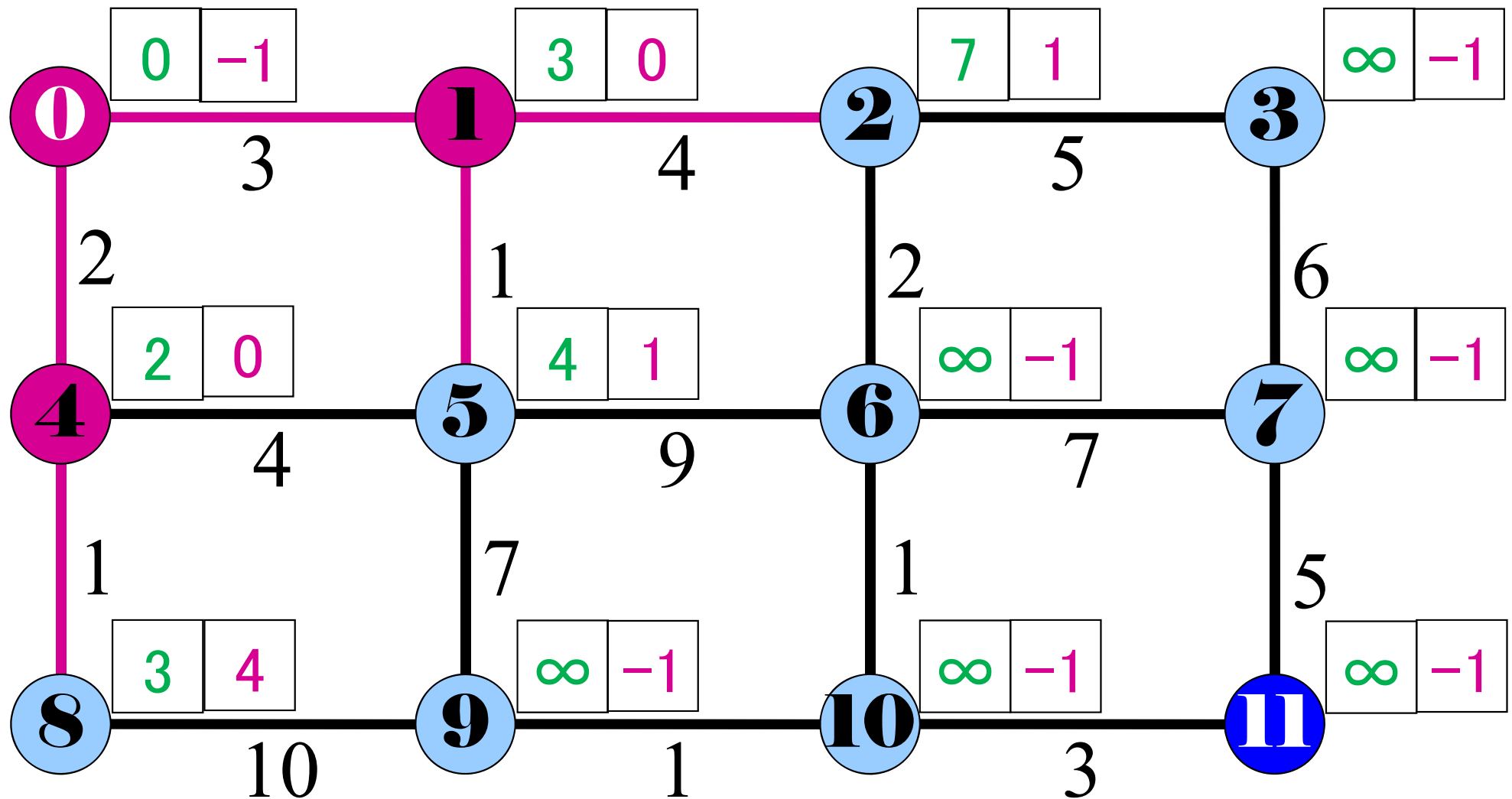
枝 e_{15} は $d(1)+c(e_{15}) = 3+1 = 4 < 6 = d(5)$ より、 $d(5) := 4$, $p(5) := 1$ に



Dijkstra法 (更新法)

step1-3: その点 v から出る全枝 $e \in E$ の作業が全て終了したら,
その点 $v \in N$ を未確定点集合 N から除去する

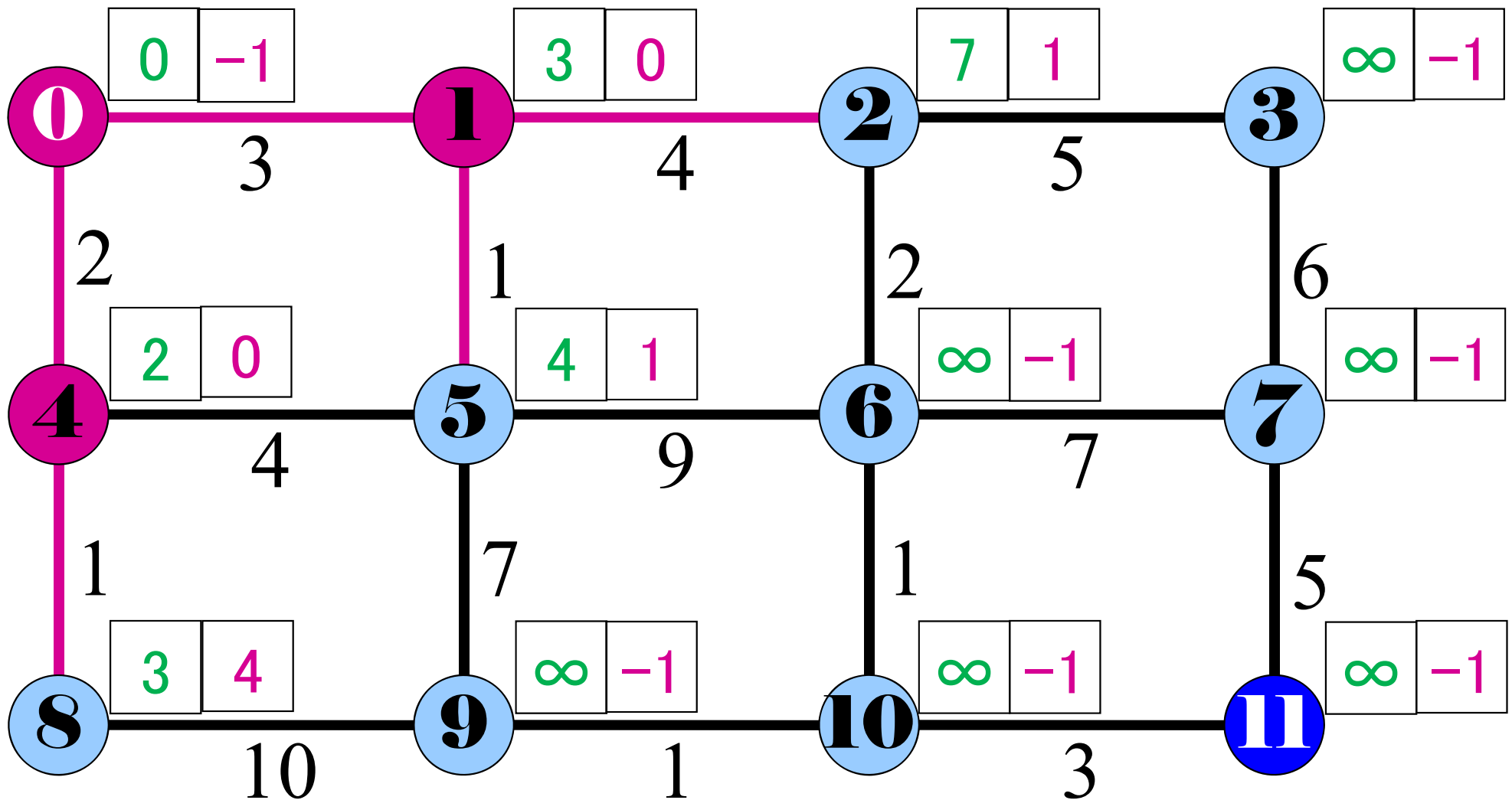
未確定点集合 $N = \{1, 2, 3, 5, 6, 7, 8, 9, 10, 11\}$
 $\rightarrow N = \{2, 3, 5, 6, 7, 8, 9, 10, 11\}$



Dijkstra法 (終了判定)

step2:未確定点集合 N が空(くう) ($N=\emptyset$)になったら終了. そうでなければ step1-1 へ戻る

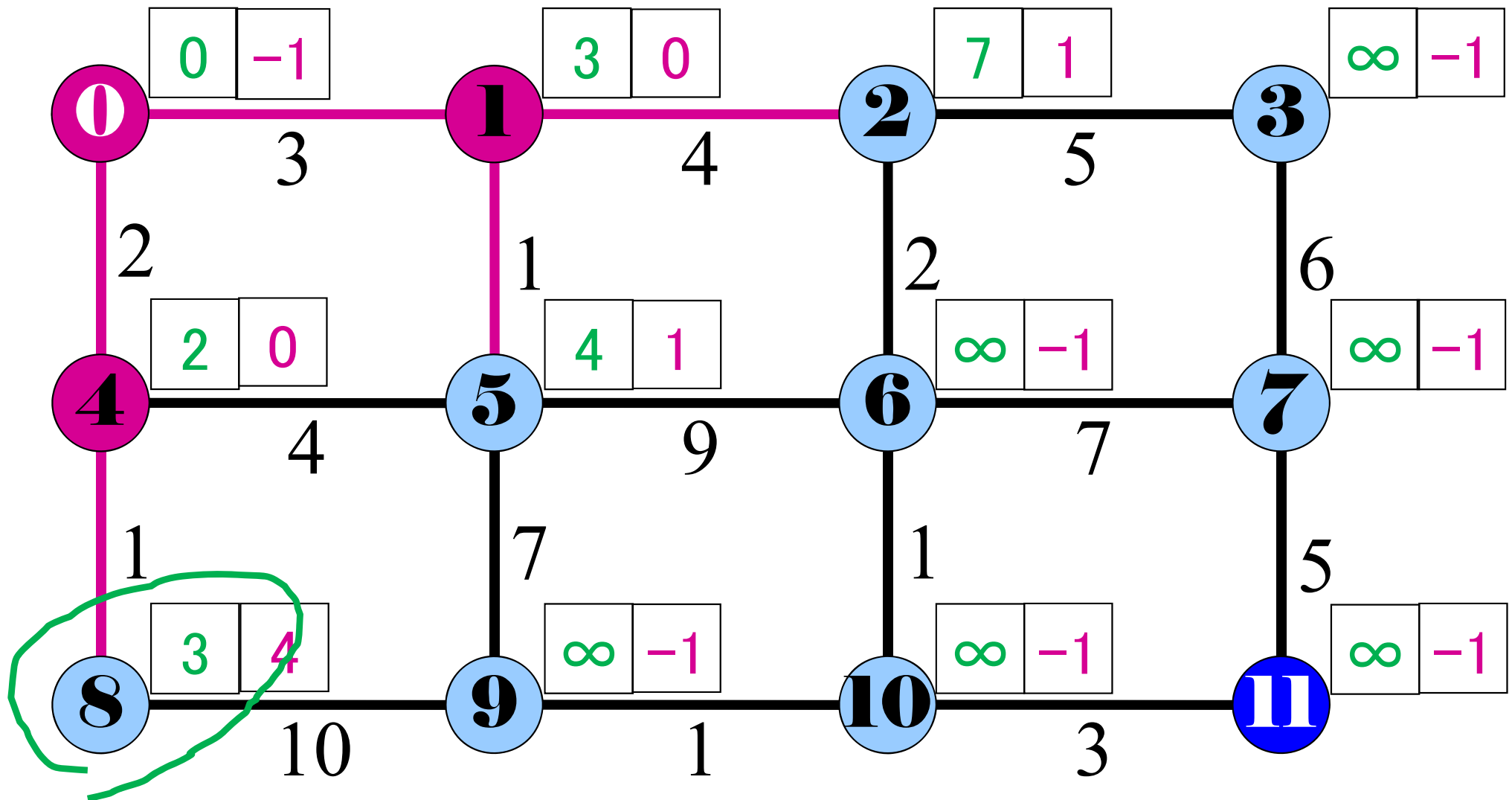
未確定点集合 $N=\{2,3,5,6,7,8,9,10,11\} \neq \emptyset$ より step1-1 へ戻る



Dijkstra法 (更新法)

step1-1: 未確定点 $v \in N$ について $d(v)$ が最小の点を見つける

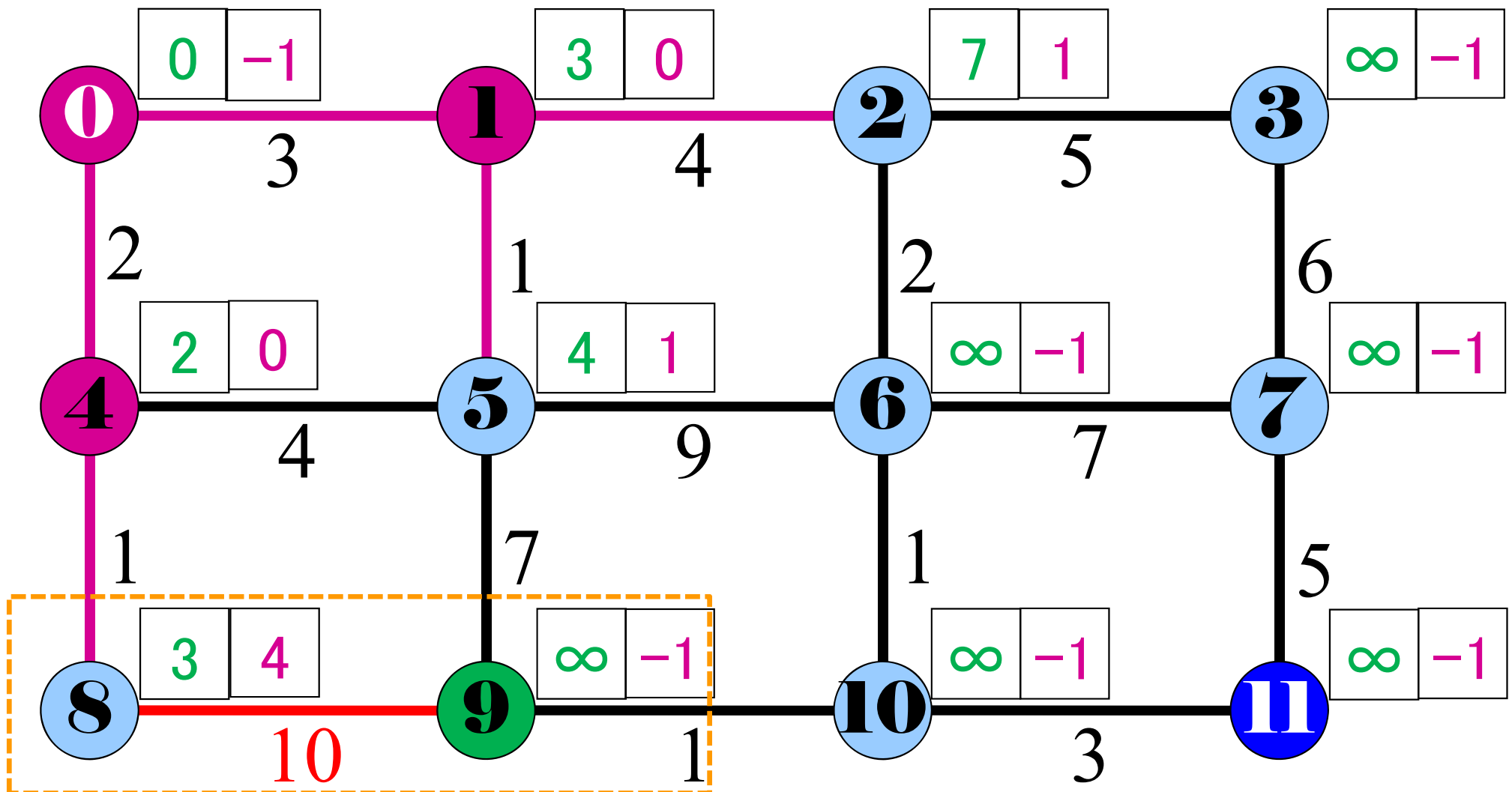
未確定点集合 $N = \{2, 3, 5, 6, 7, 8, 9, 10, 11\}$ で $d(\cdot)$ 最小点 8 を見つけた



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$
+ 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、
もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

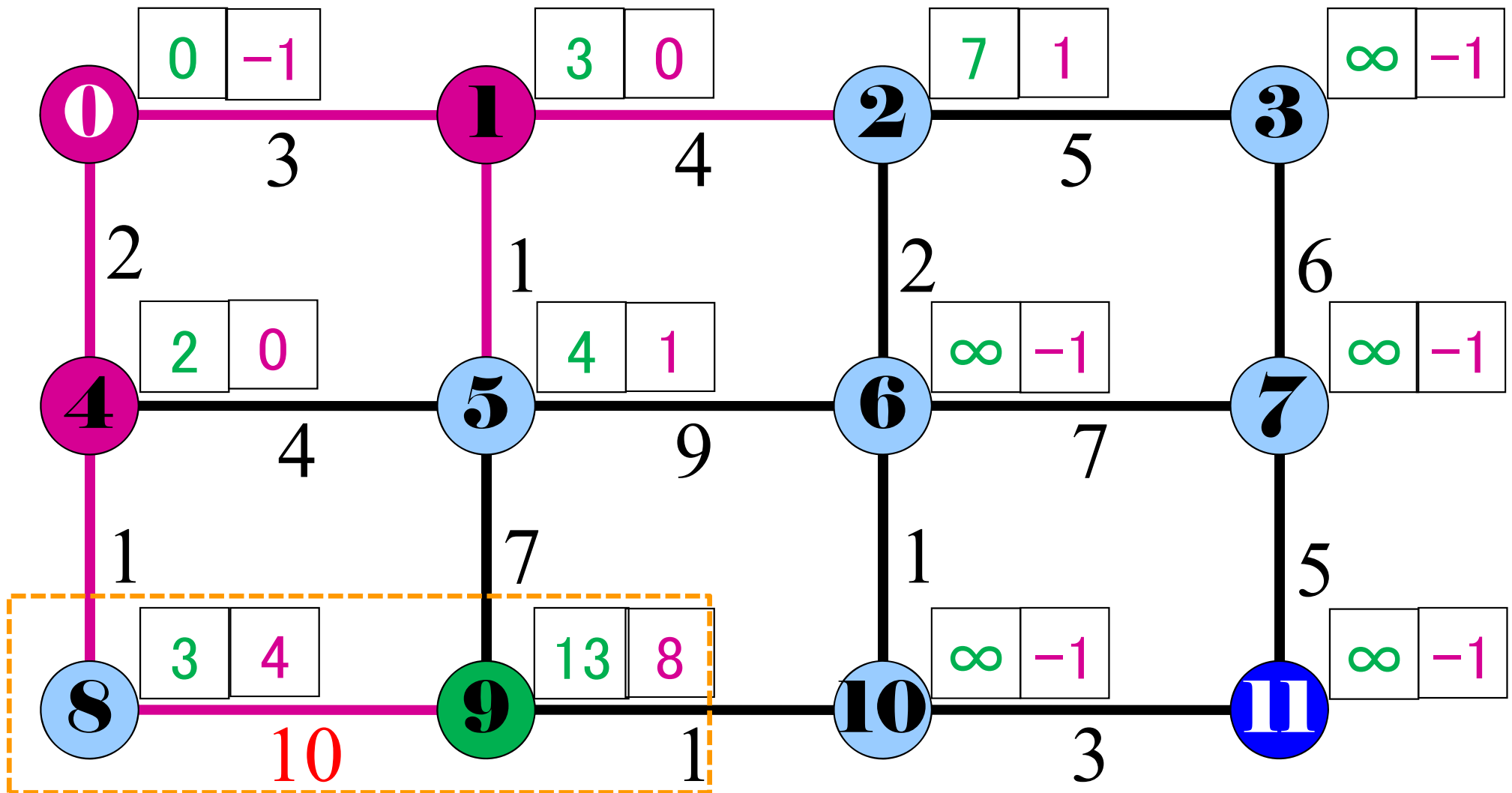
枝 e_{810} は $d(4)+c(e_{810}) = 3+10=13 < \infty=d(9)$ より, $d(9) := 13$, $p(9) := 8$ に



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$
+ 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、
もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

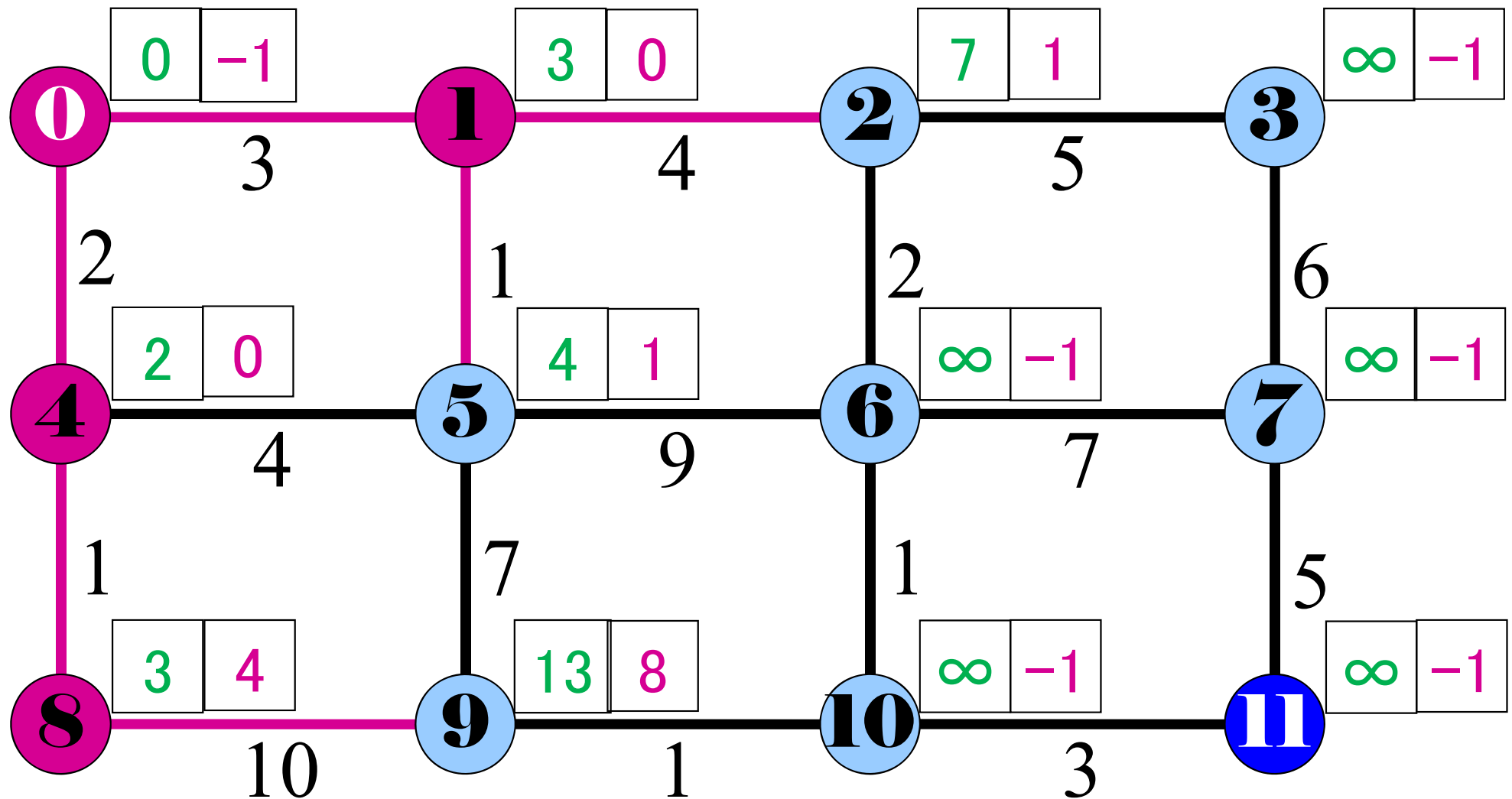
枝 e_{810} は $d(4)+c(e_{810}) = 3+10=13 < \infty = d(9)$ より, $d(9) := 13$, $p(9) := 8$ に



Dijkstra法 (更新法)

step1-3: その点 v から出る全枝 $e \in E$ の作業が全て終了したら,
その点 $v \in N$ を未確定点集合 N から除去する

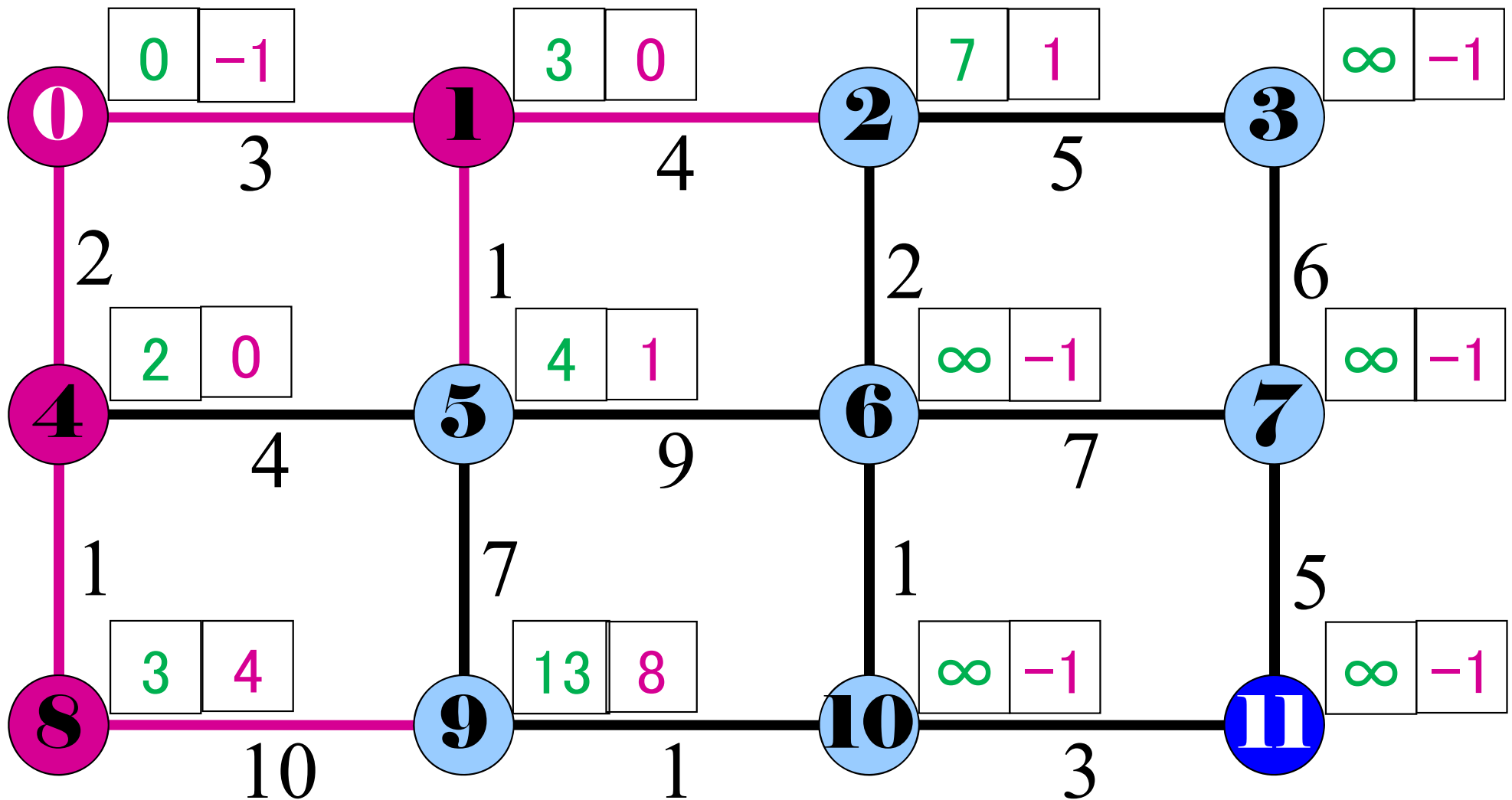
未確定点集合 $N = \{2, 3, 5, 6, 7, 8, 9, 10, 11\}$
 $\rightarrow N = \{2, 3, 5, 6, 7, 9, 10, 11\}$



Dijkstra法 (終了判定)

step2:未確定点集合 N が空(くう) ($N=\emptyset$)になったら終了. そうでなければ step1-1 へ戻る

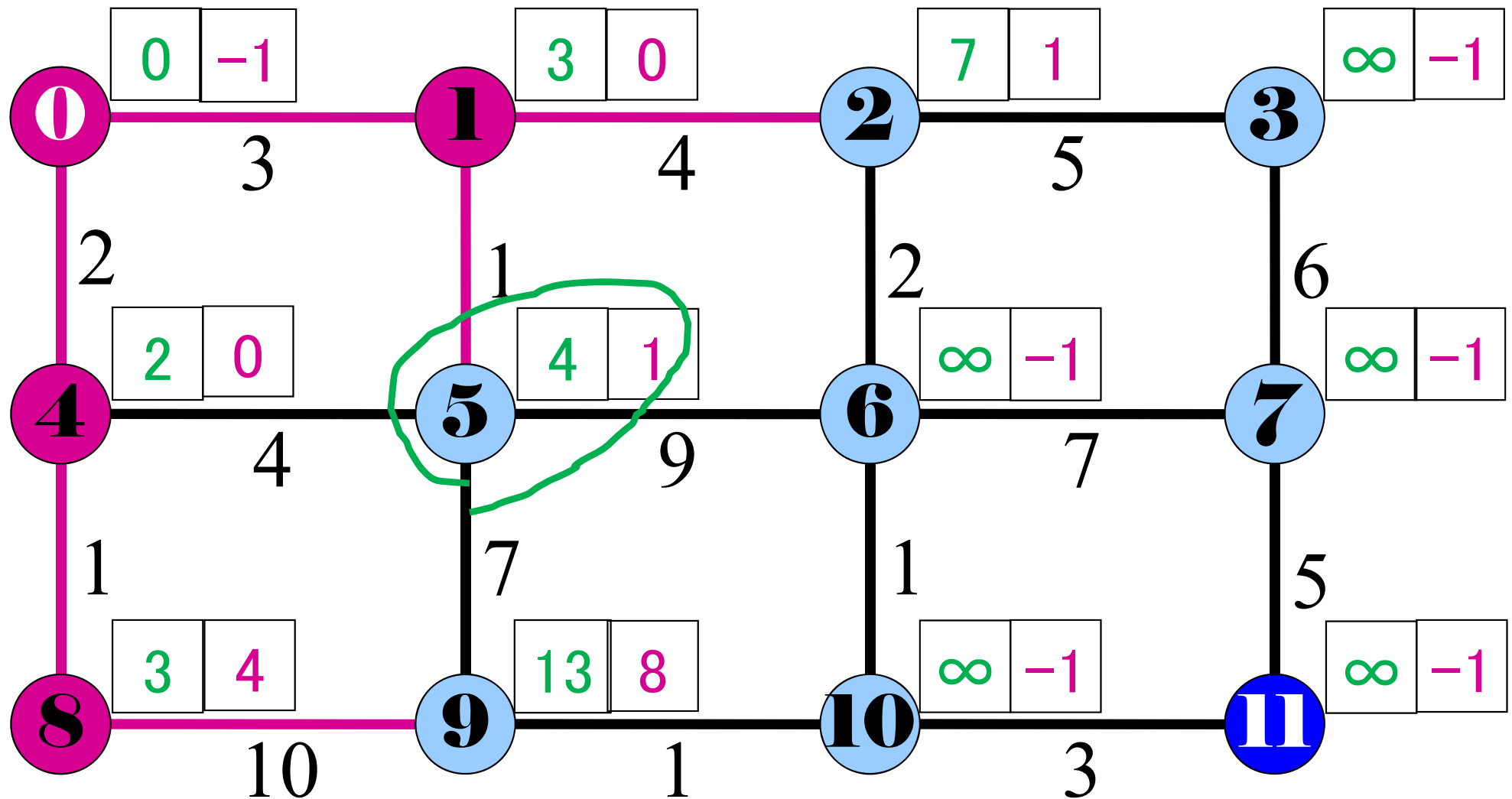
未確定点集合 $N=\{2,3,5,6,7,9,10,11\} \neq \emptyset$ より step1-1 へ戻る



Dijkstra法 (更新法)

step1-1: 未確定点 $v \in N$ について $d(v)$ が最小の点を見つける

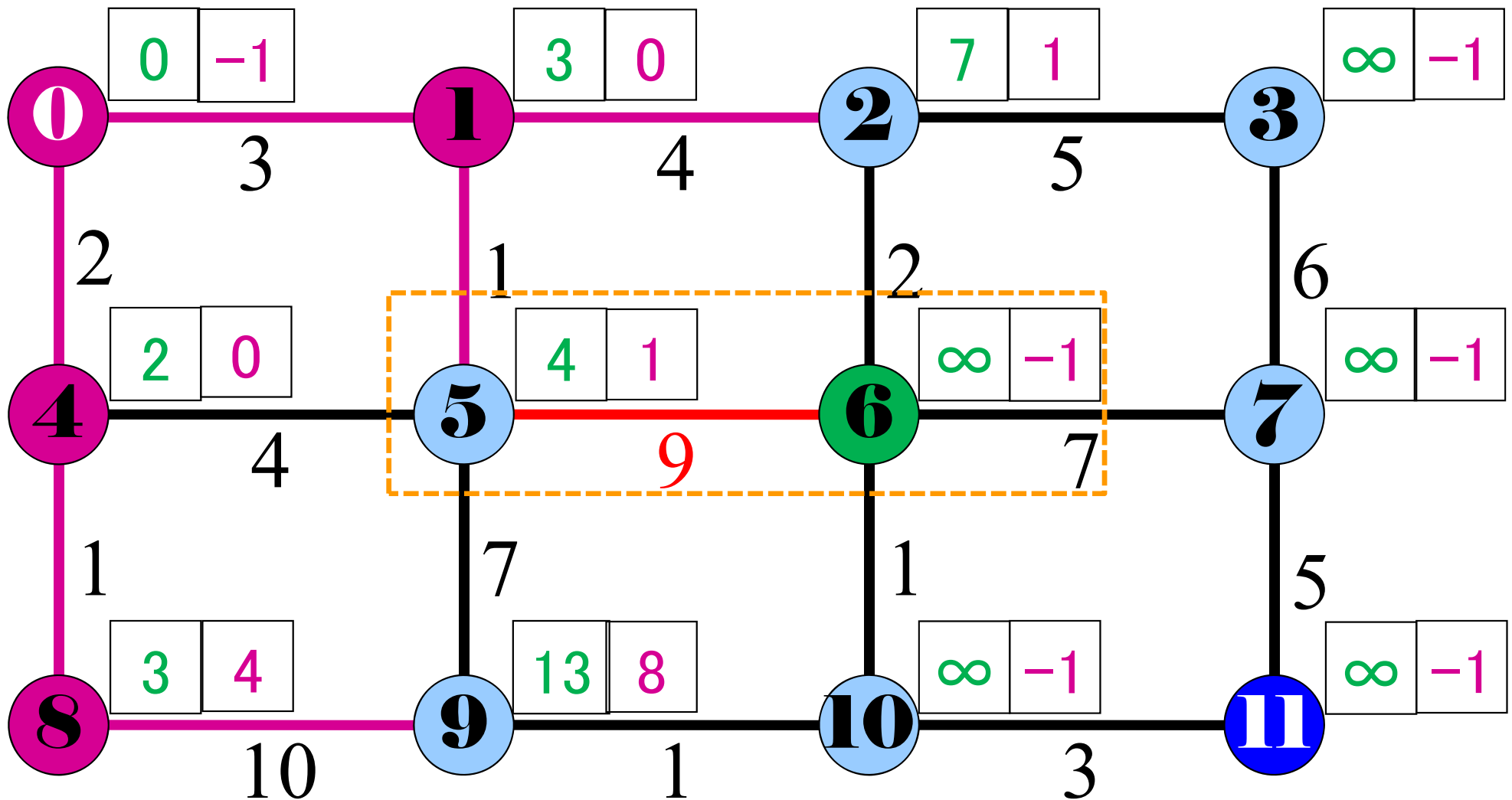
未確定点集合 $N = \{2, 3, 5, 6, 7, 9, 10, 11\}$ で $d(\cdot)$ 最小点 5 を見つけた



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$
+ 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、
もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

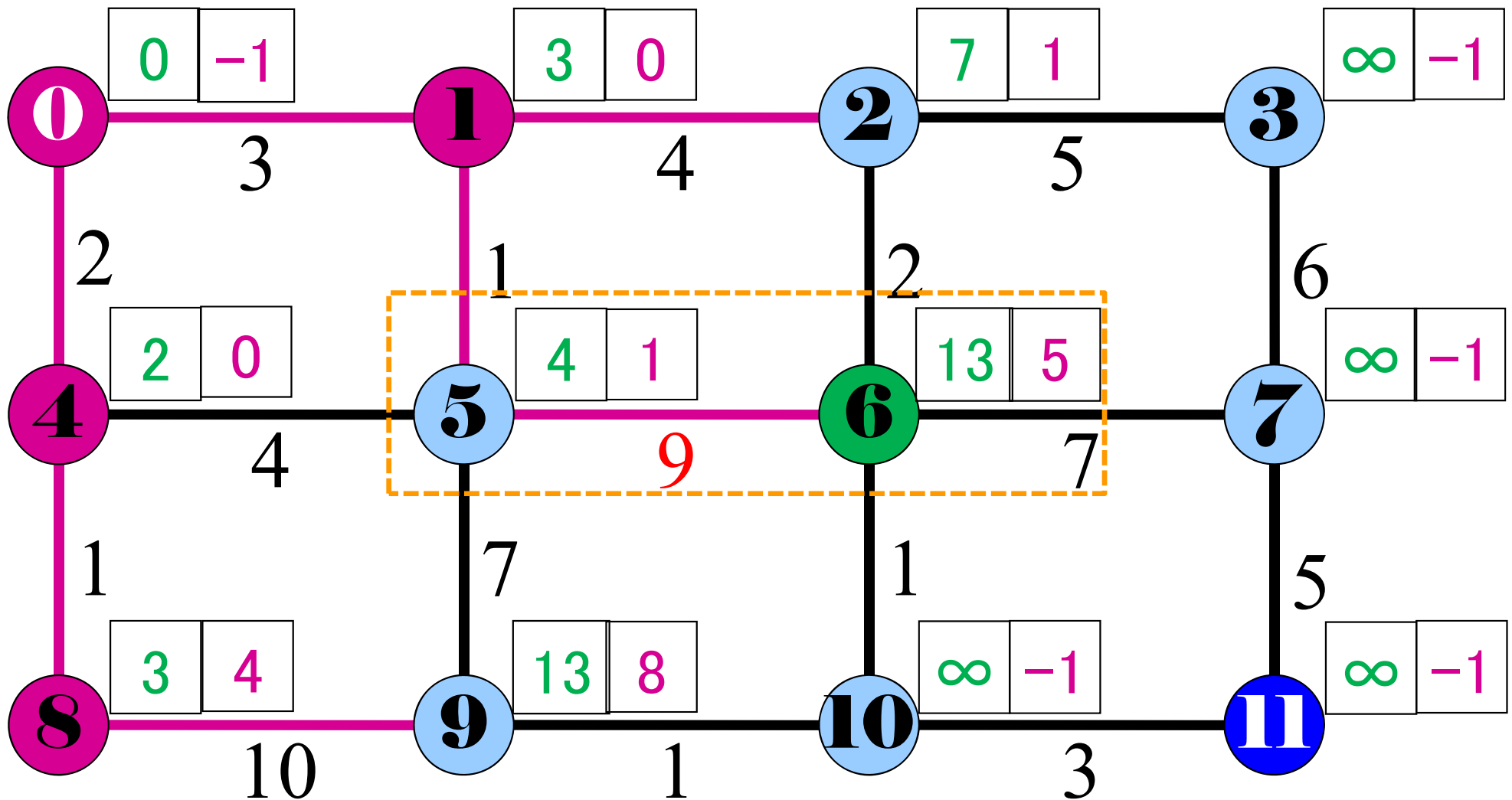
枝 e_{56} は $d(5)+c(e_{56}) = 4+9=13 < \infty = d(6)$ より、 $d(6) := 13$, $p(6) := 5$ に



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$
+ 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、
もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

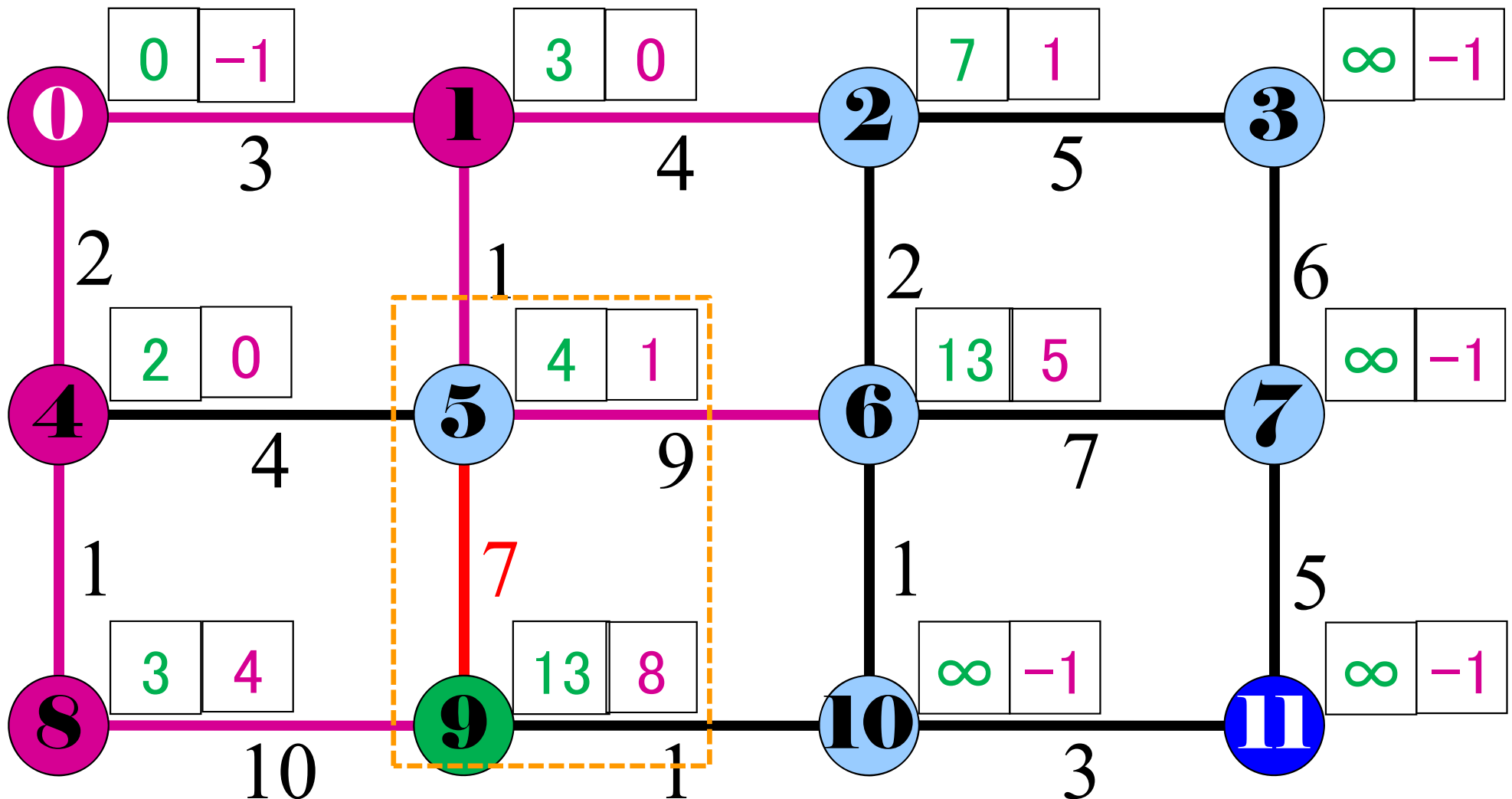
枝 e_{56} は $d(5)+c(e_{56}) = 4+9=13 < \infty = d(6)$ より、 $d(6) := 13$, $p(6) := 5$ に



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$ + 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

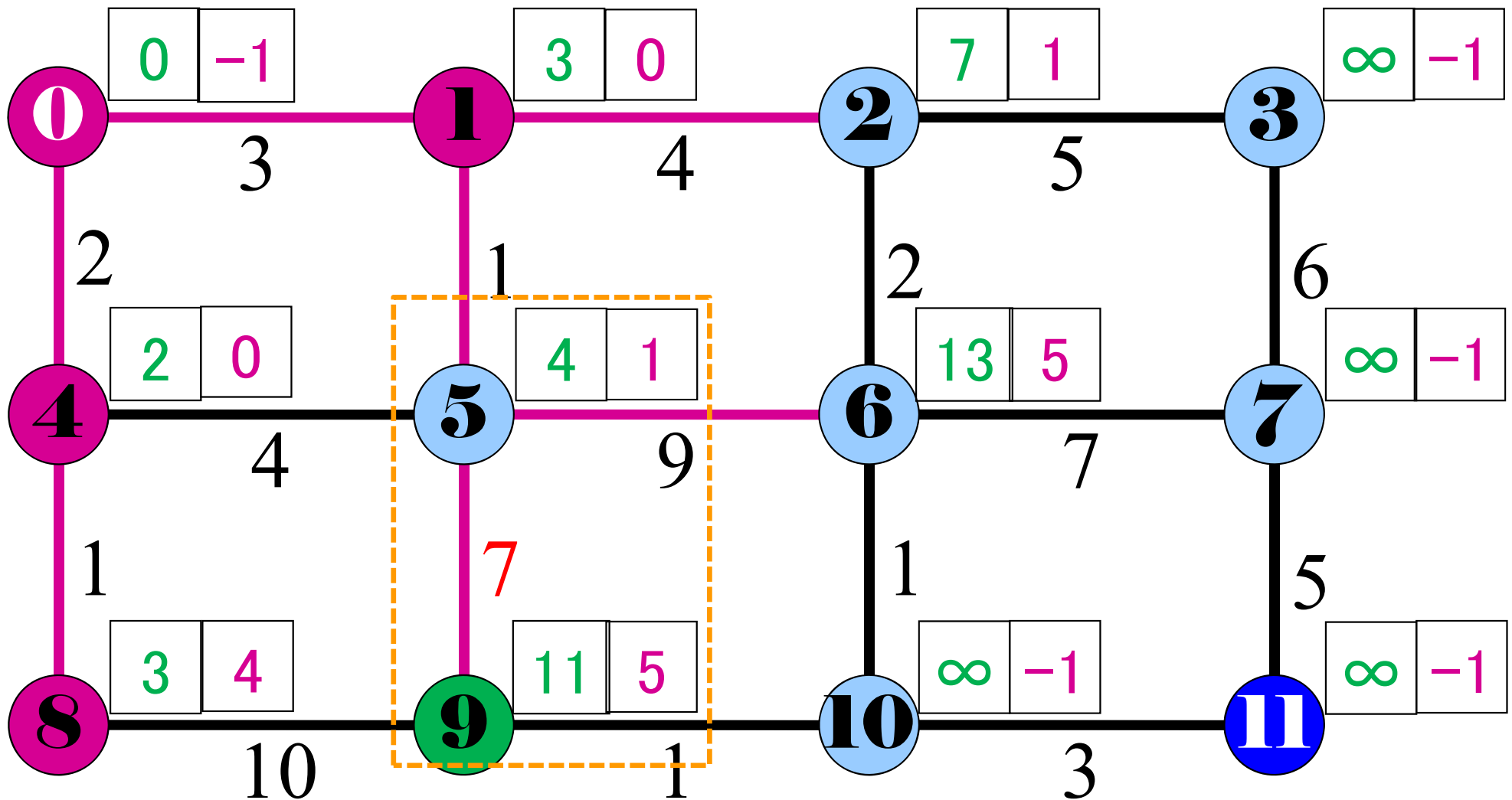
枝 e_{59} は $d(5)+c(e_{59}) = 4+7=11 < 13 = d(9)$ より, $d(9) := 11$, $p(9) := 5$ に



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$
+ 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、
もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

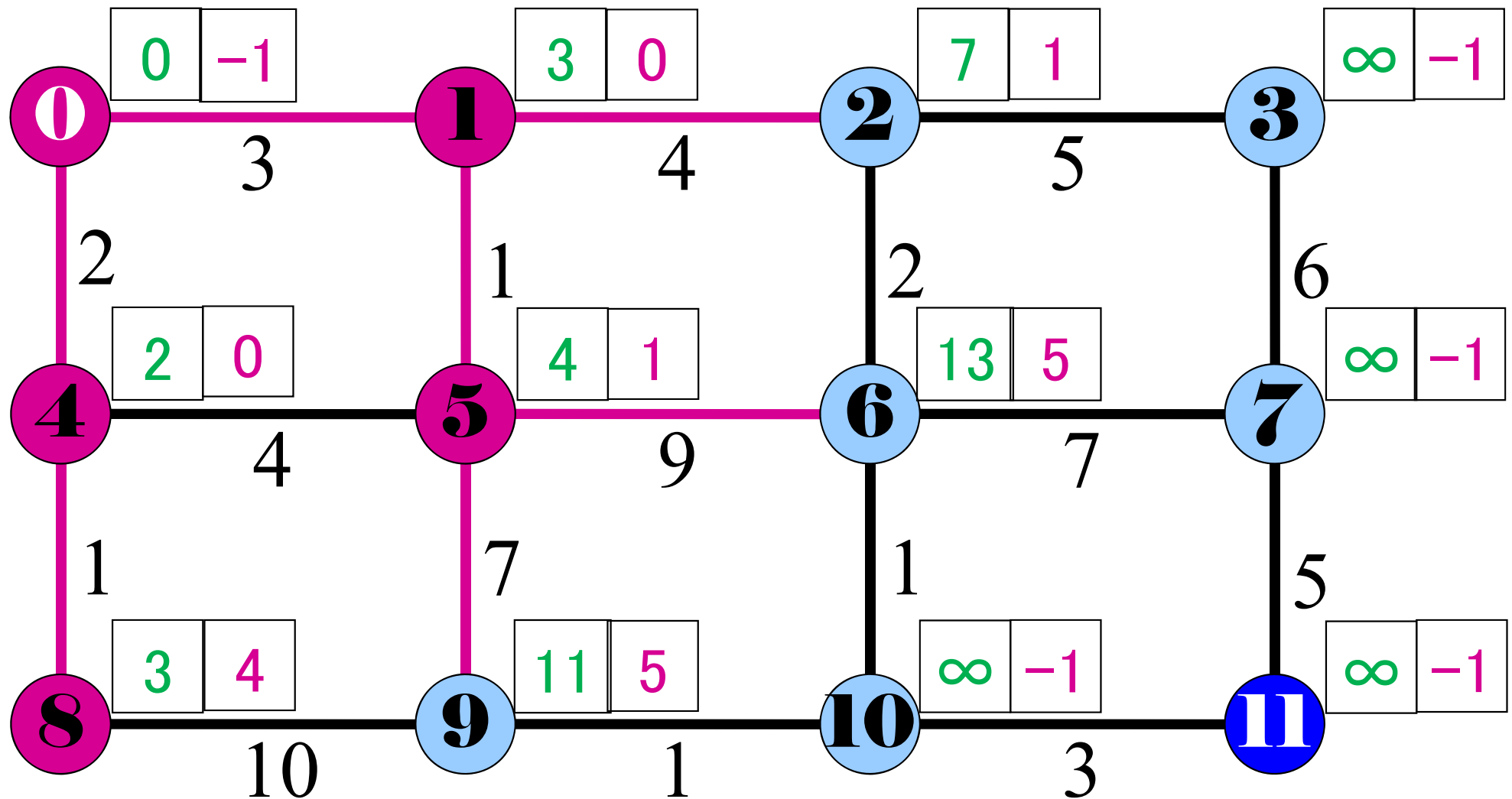
枝 e_{59} は $d(5)+c(e_{59}) = 4+7=11 < 13 = d(9)$ より, $d(9) := 11$, $p(9) := 5$ に



Dijkstra法 (更新法)

step1-3: その点 v から出る全枝 $e \in E$ の作業が全て終了したら,
その点 $v \in N$ を未確定点集合 N から除去する

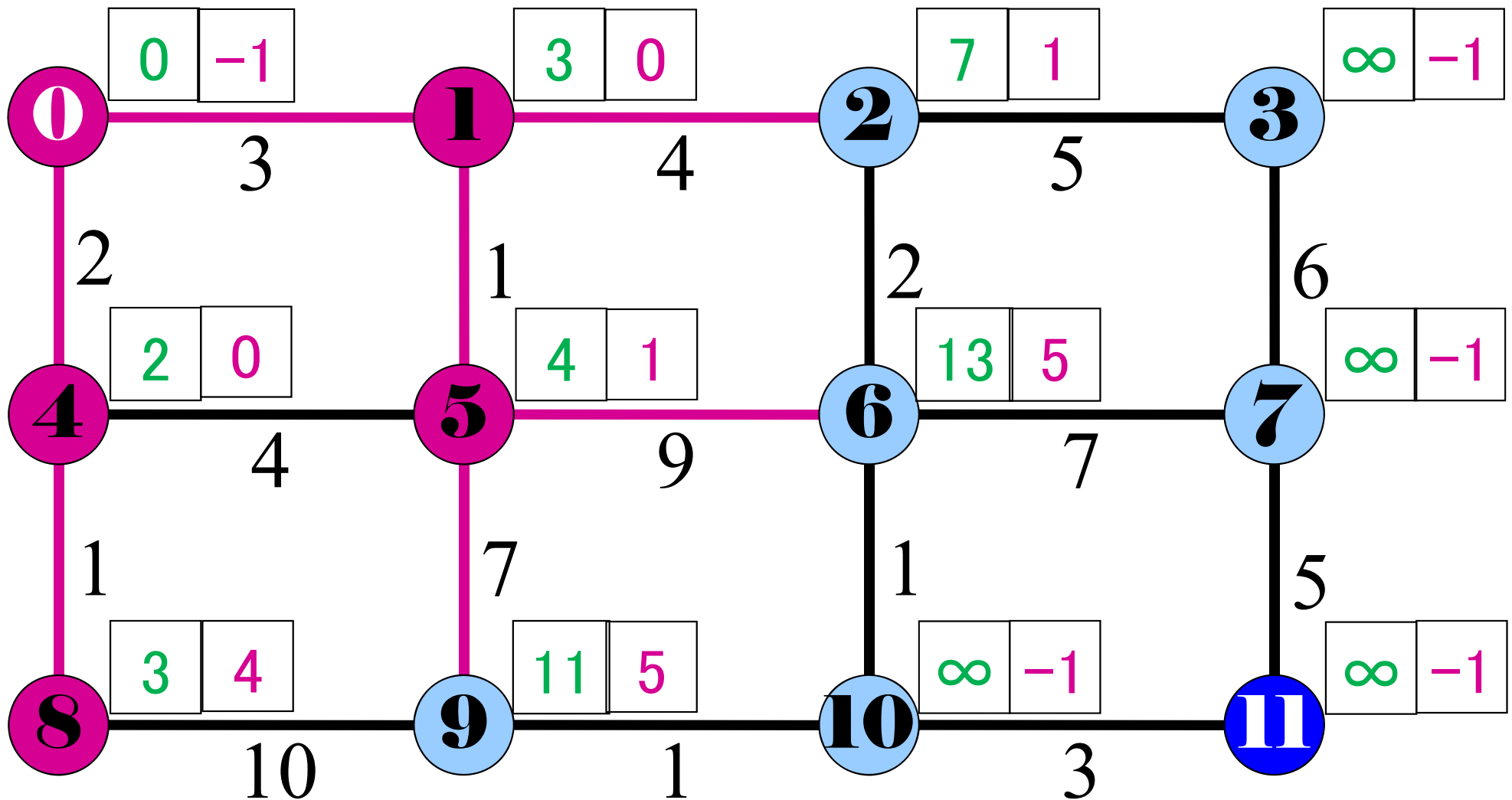
未確定点集合 $N = \{2, 3, 5, 6, 7, 9, 10, 11\}$
 $\rightarrow N = \{2, 3, 6, 7, 9, 10, 11\}$



Dijkstra法 (終了判定)

step2:未確定点集合 N が空(くう) ($N=\emptyset$)になったら終了. そうでなければ step1-1 へ戻る

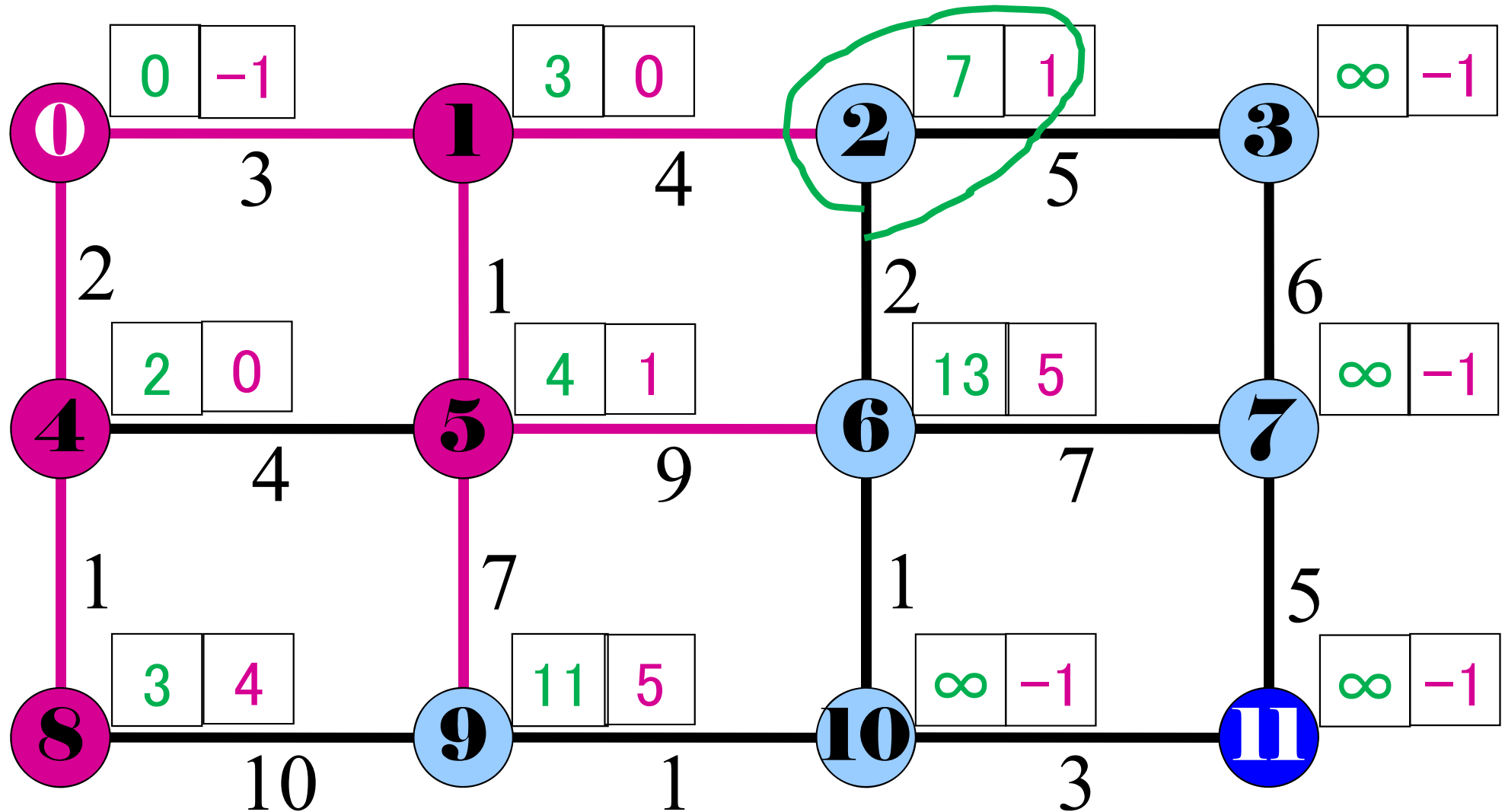
未確定点集合 $N=\{2,3,6,7,9,10,11\} \neq \emptyset$ より step1-1 へ戻る



Dijkstra法 (更新法)

step1-1: 未確定点 $v \in N$ について $d(v)$ が最小の点を見つける

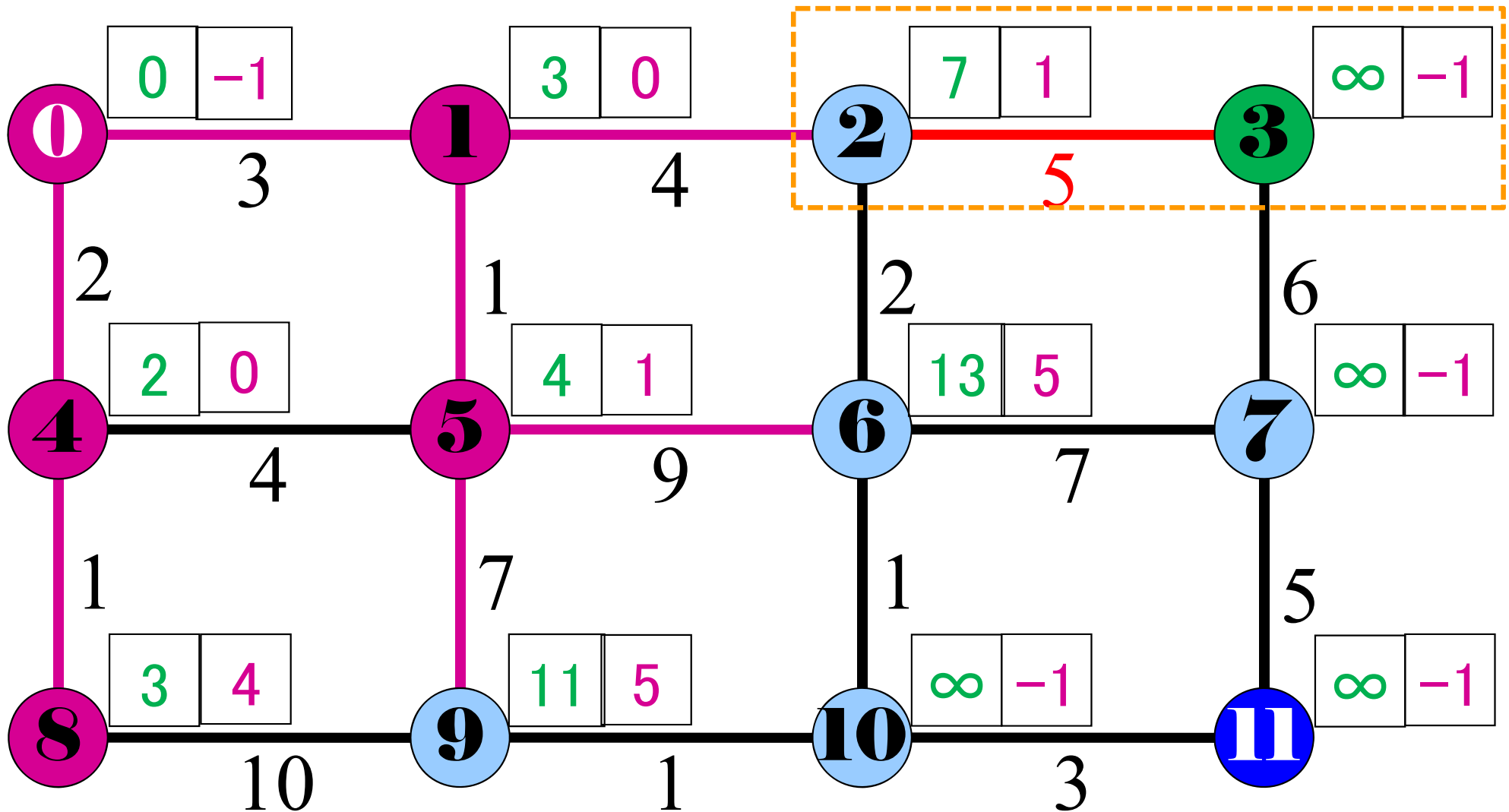
未確定点集合 $N = \{2, 3, 6, 7, 9, 10, 11\}$ で $d(\cdot)$ 最小点 2 を見つけた



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$
+ 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、
もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

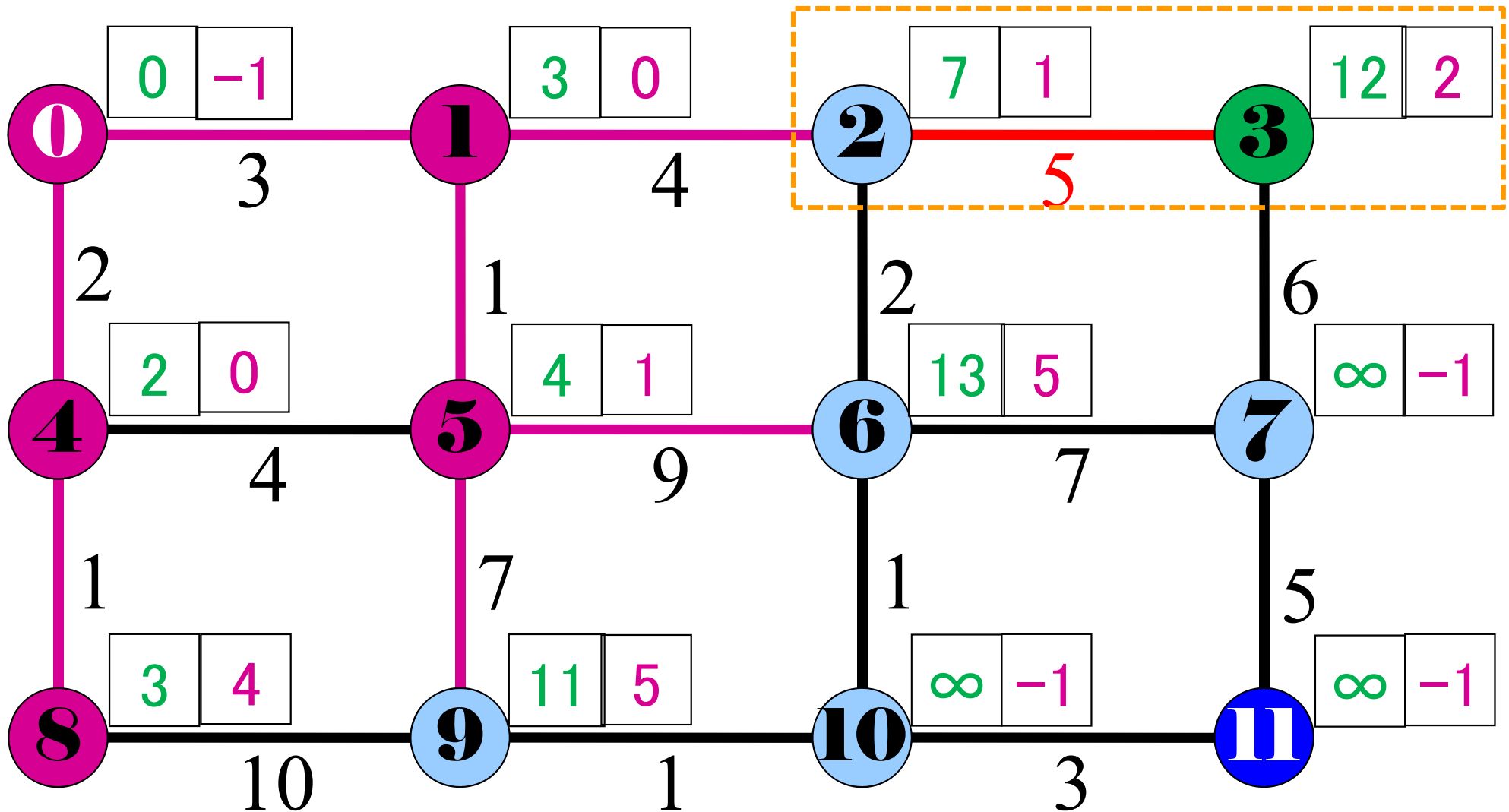
枝 e_{23} は $d(2)+c(e_{23}) = 7+5=12 < \infty=d(3)$ より、 $d(3) := 12$, $p(3) := 2$ に



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$
+ 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、
もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

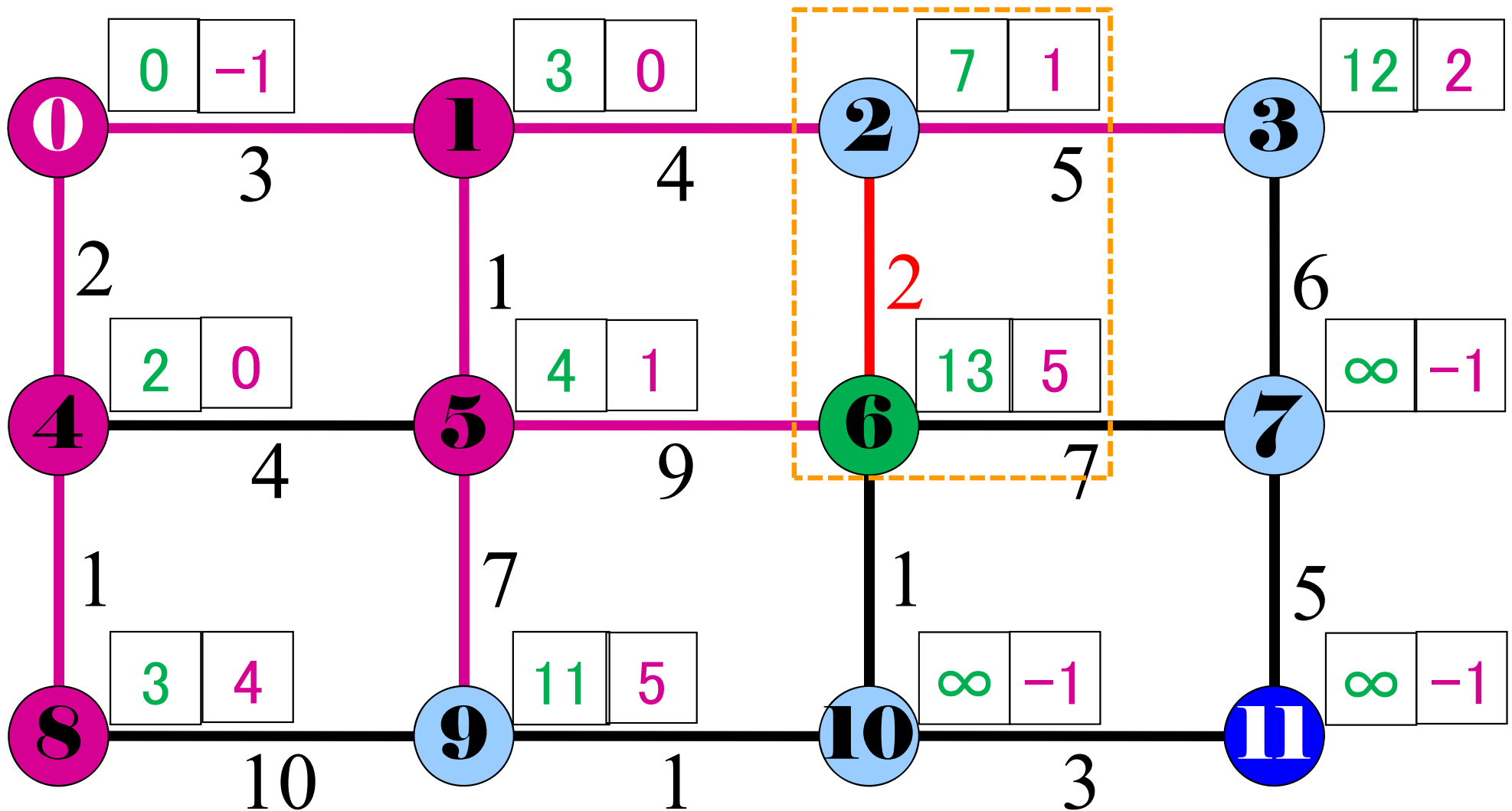
枝 e_{23} は $d(2)+c(e_{23}) = 7+5=12 < \infty=d(3)$ より, $d(3) := 12$, $p(3) := 2$ に



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$
+ 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、
もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

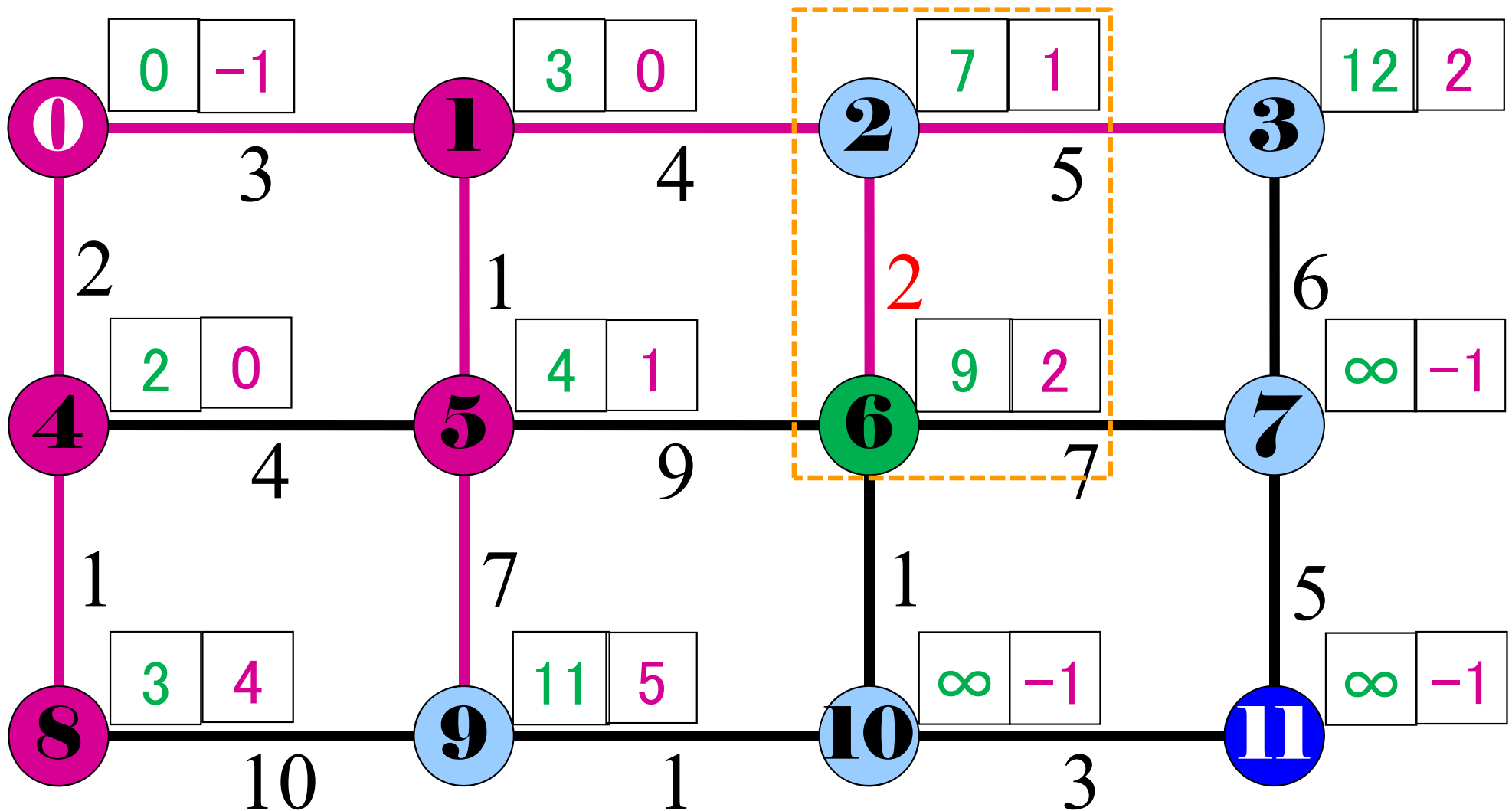
枝 e_{26} は $d(2)+c(e_{26}) = 7+2=9 < 13=d(6)$ より、 $d(6) := 9$, $p(6) := 2$ に



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$
+ 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、
もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

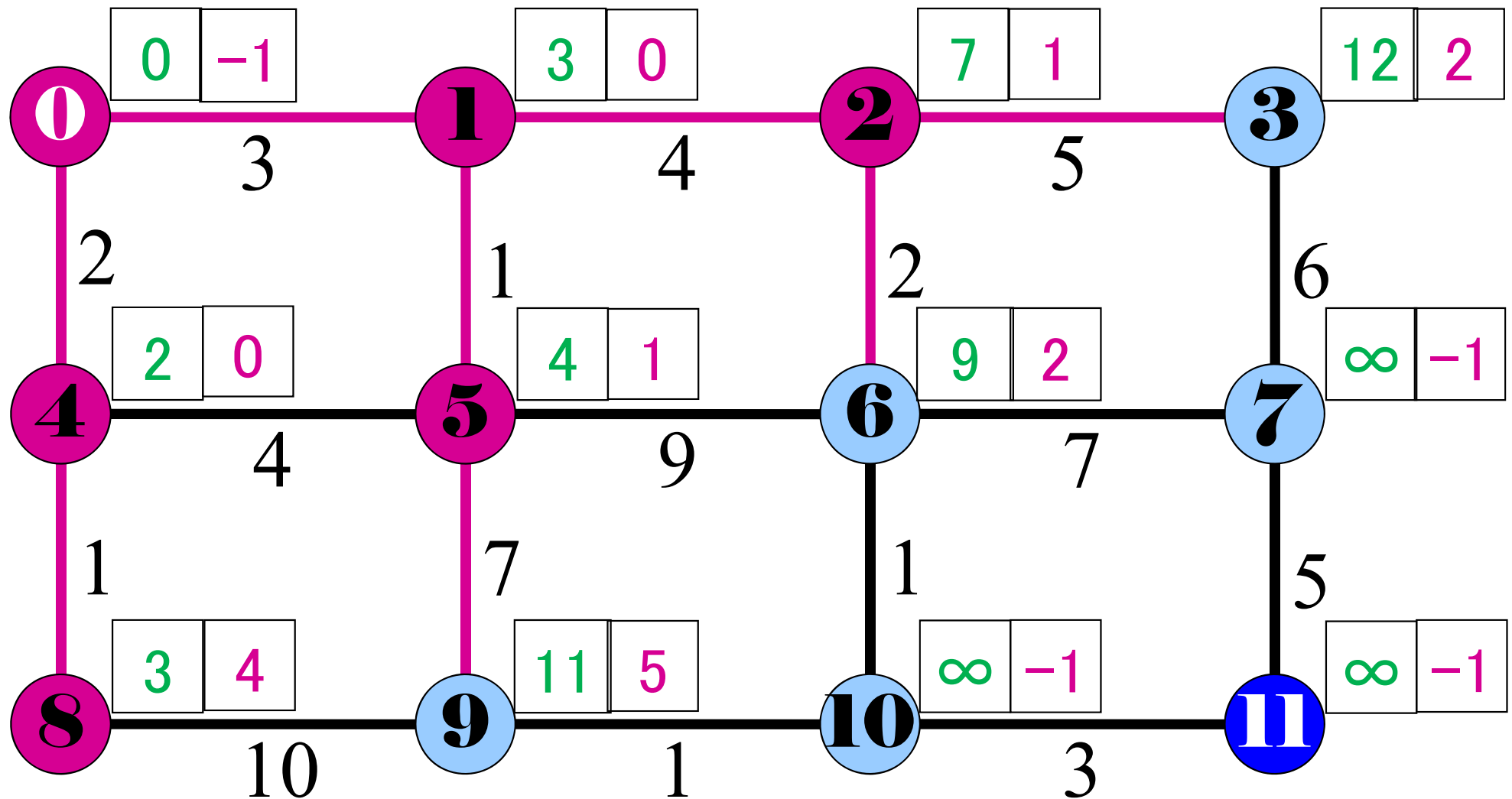
枝 e_{26} は $d(2)+c(e_{26}) = 7+2=9 < 13=d(6)$ より、 $d(6) := 9$, $p(6) := 2$ に



Dijkstra法 (更新法)

step1-3: その点 v から出る全枝 $e \in E$ の作業が全て終了したら、その点 $v \in N$ を**未確定点集合** N から**除去**する

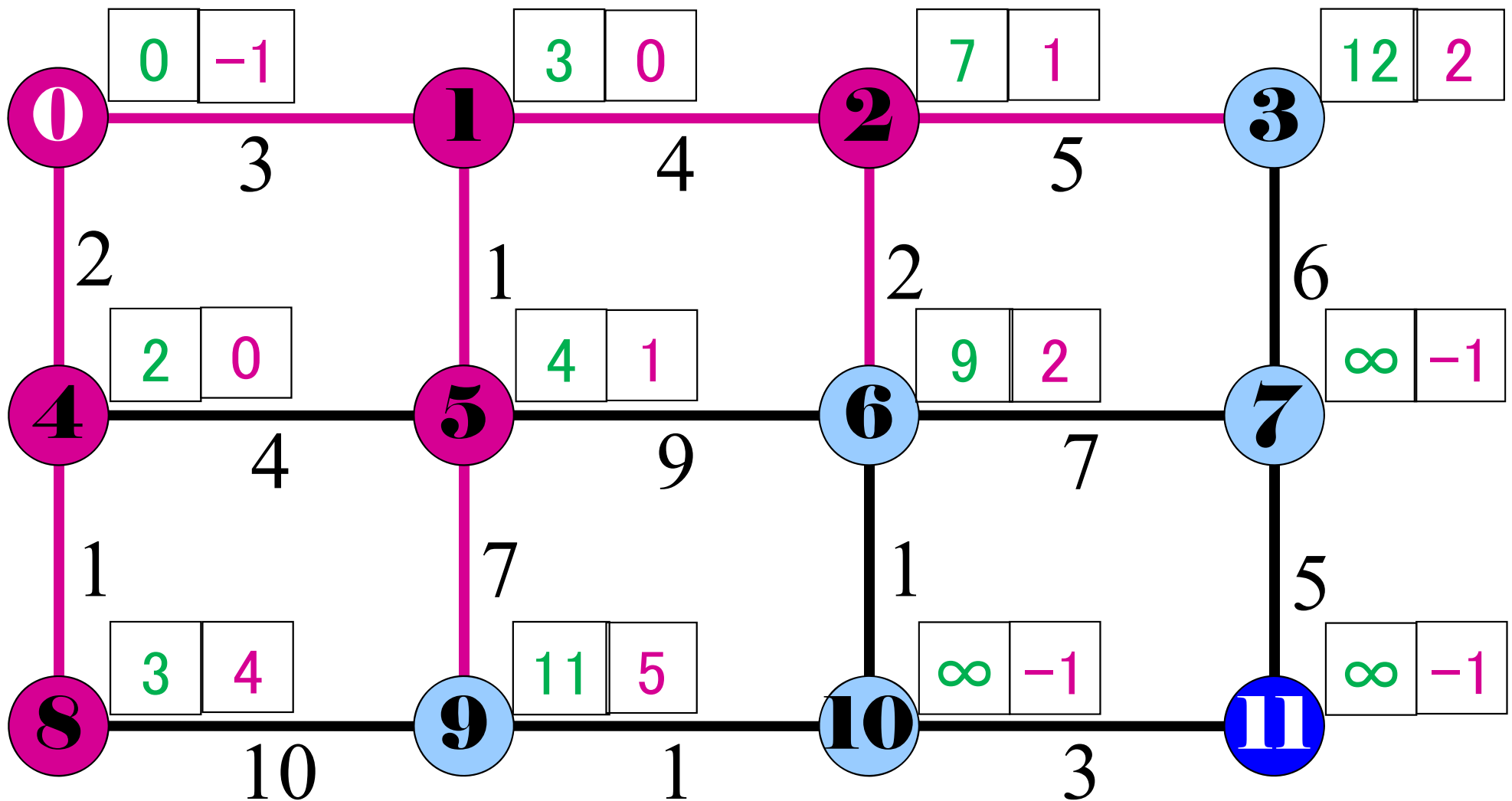
未確定点集合 $N=\{2,3,6,7,9,10,11\}$
 $\rightarrow N=\{3,6,7,9,10,11\}$



Dijkstra法 (終了判定)

step2:未確定点集合 N が空(くう)($N=\emptyset$)になったら終了. そうでなければ **step1-1** へ戻る

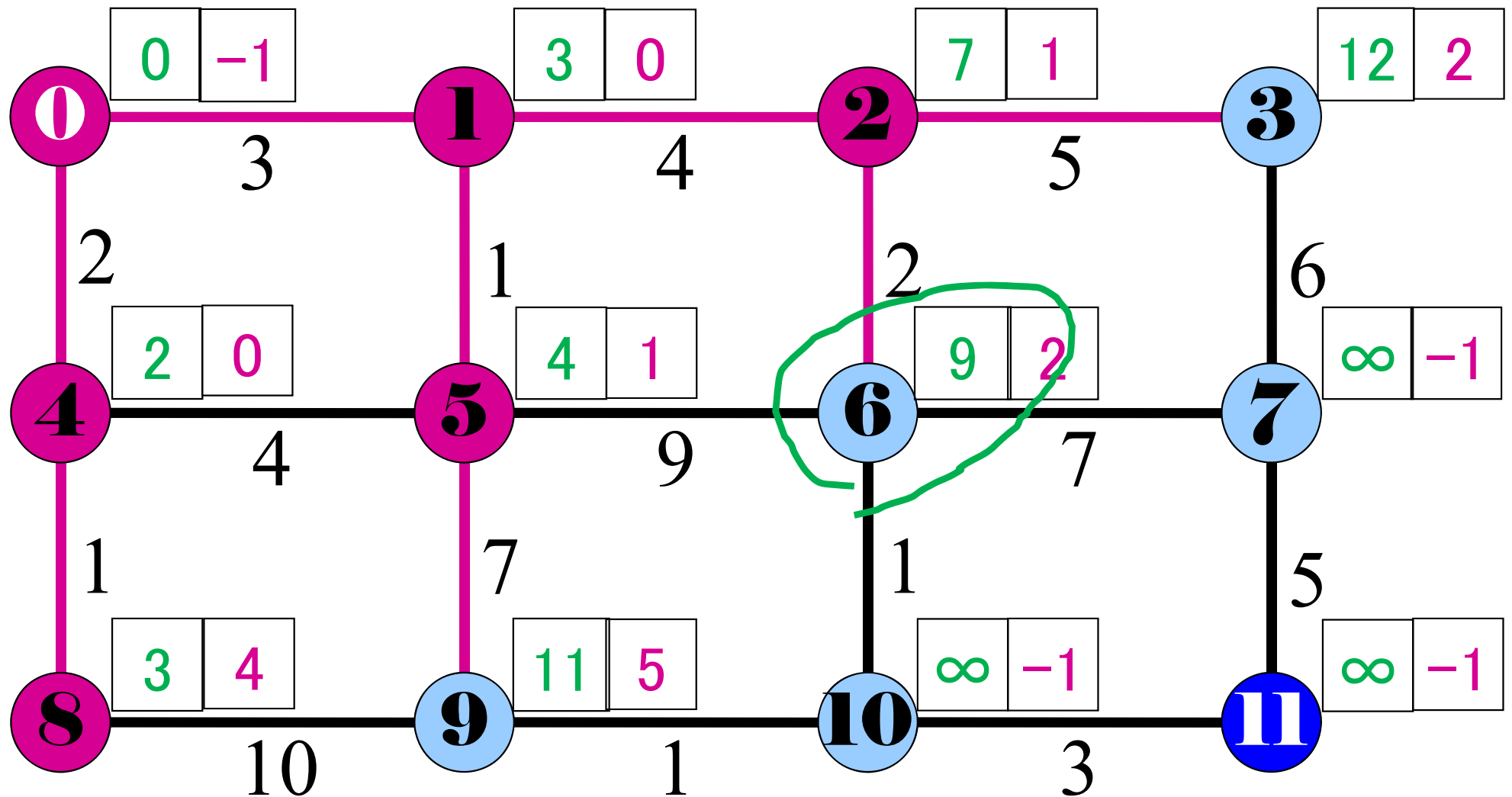
未確定点集合 $N=\{3,6,7,9,10,11\} \neq \emptyset$ より **step1-1** へ戻る



Dijkstra法 (更新法)

step1-1: 未確定点 $v \in N$ について $d(v)$ が最小の点を見つける

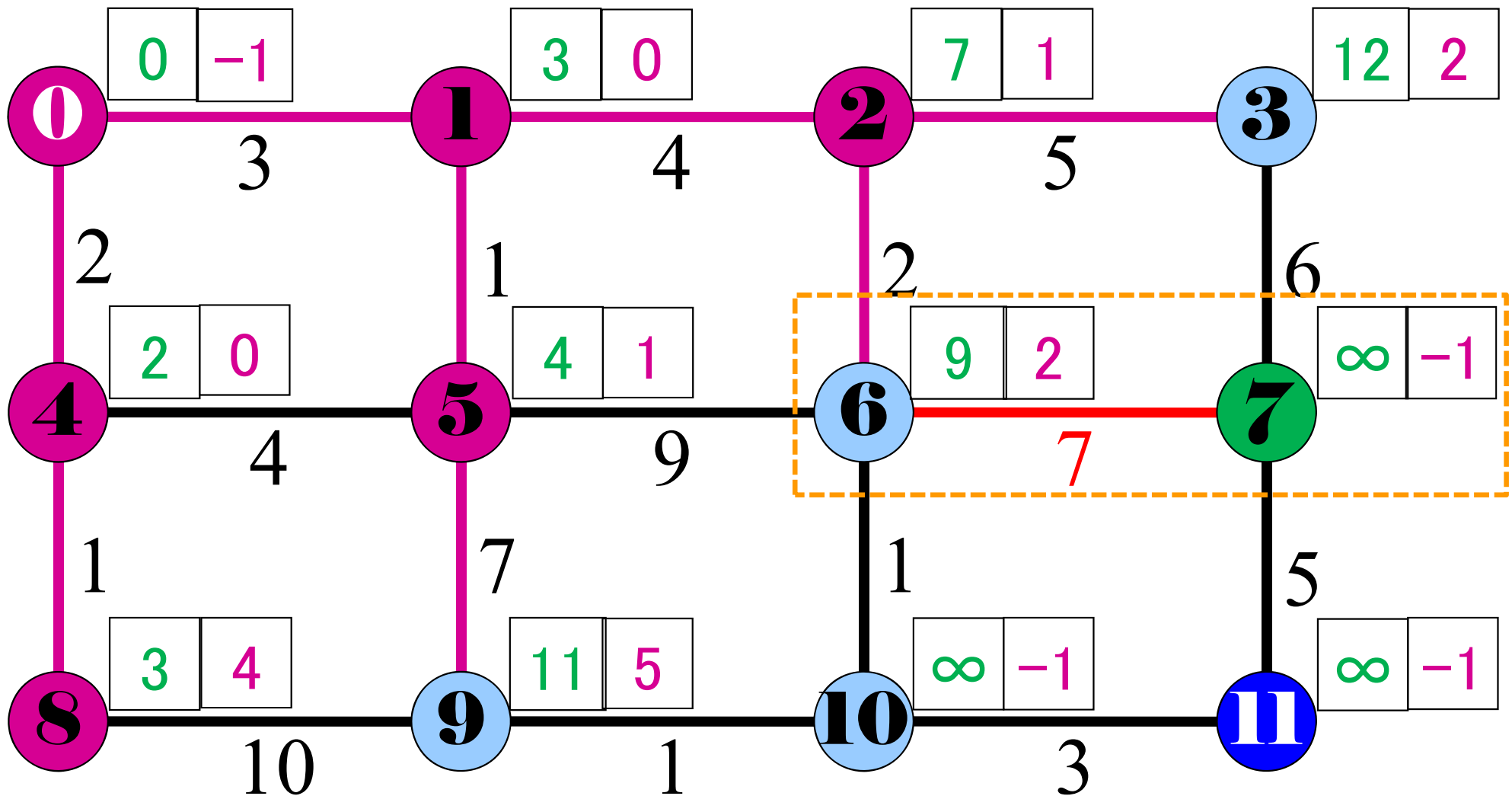
未確定点集合 $N = \{3, 6, 7, 9, 10, 11\}$ で $d(\cdot)$ 最小点 6 を見つけた



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$
+ 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、
もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

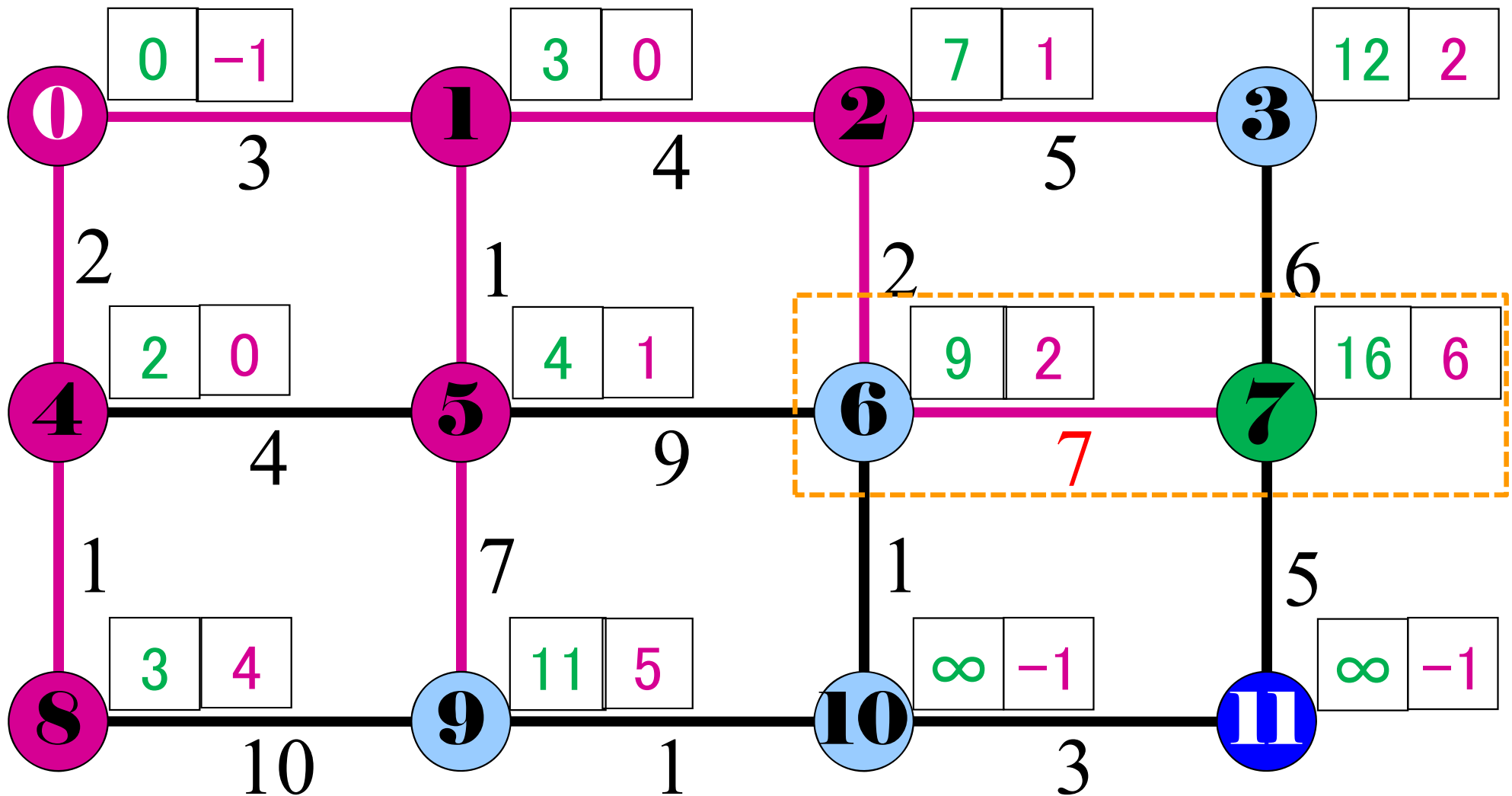
枝 e_{67} は $d(6)+c(e_{67}) = 9+7=16 < \infty=d(7)$ より, $d(7) := 16$, $p(7) := 6$ に



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$
+ 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、
もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

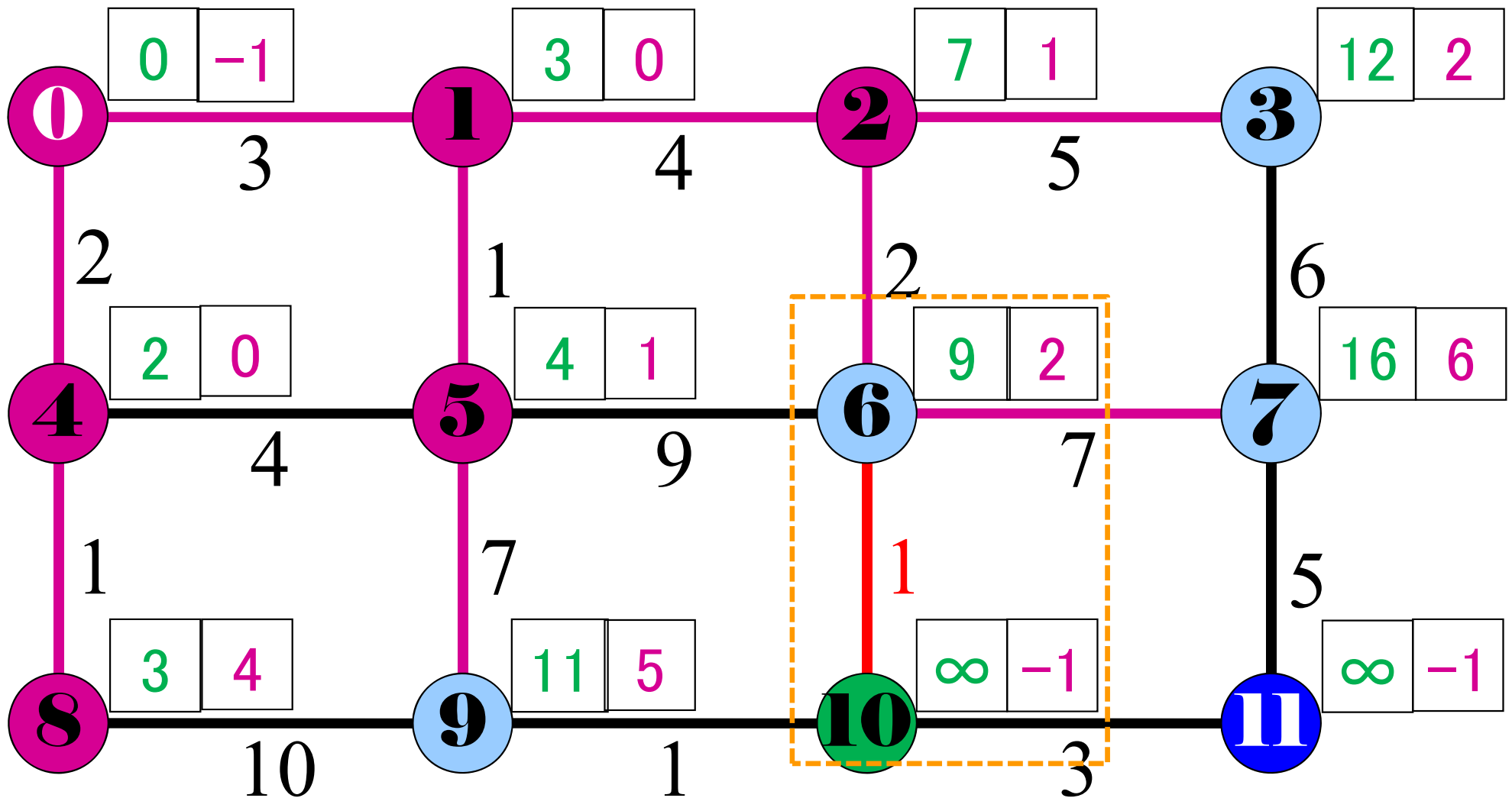
枝 e_{67} は $d(6)+c(e_{67}) = 9+7=16 < \infty=d(7)$ より, $d(7) := 16$, $p(7) := 6$ に



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$ + 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

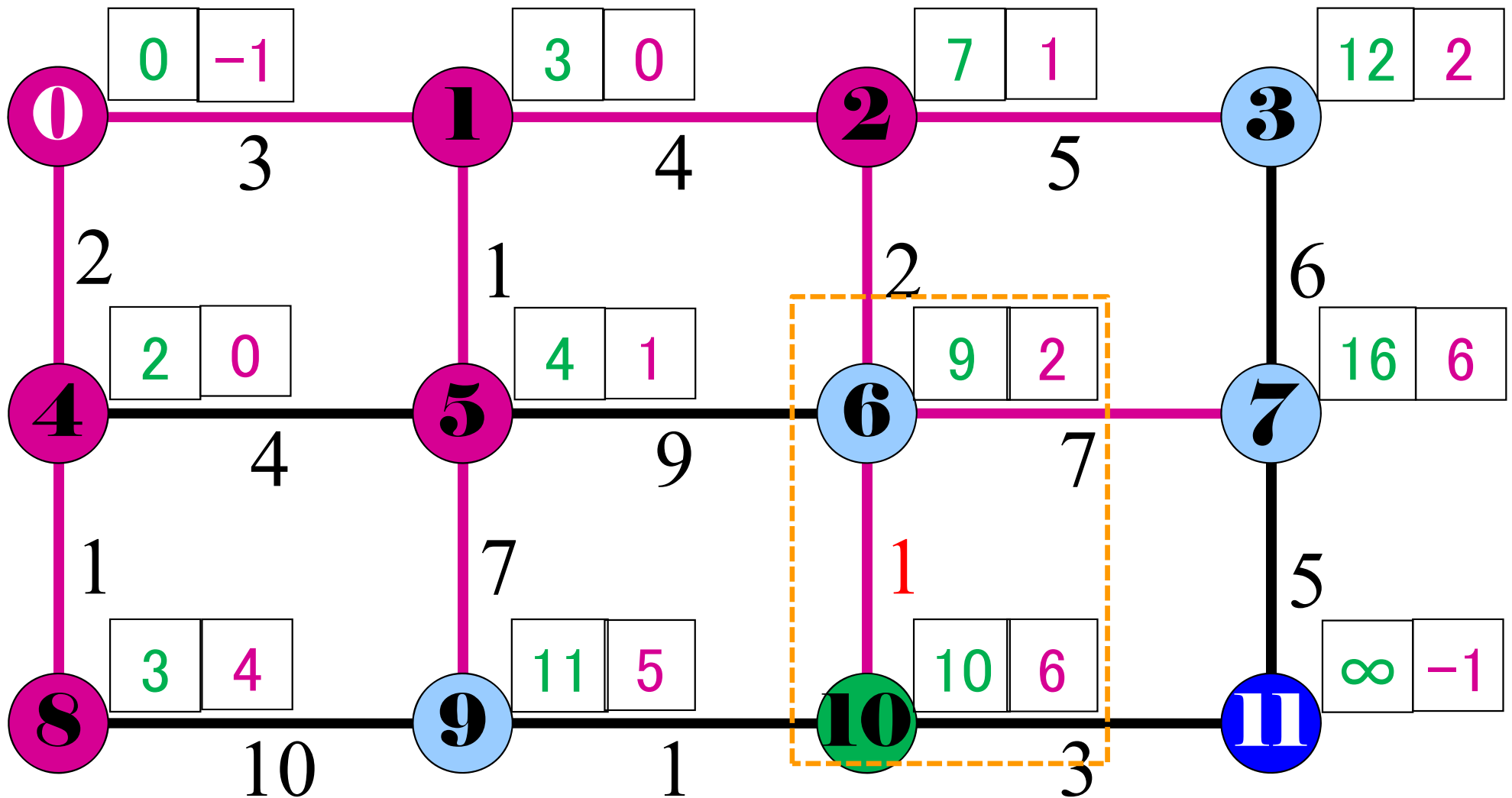
枝 e_{610} は $d(6)+c(e_{610})=9+1=10 < \infty=d(10)$ より, $d(10):=10$, $p(10):=6$ に



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$
+ 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、
もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

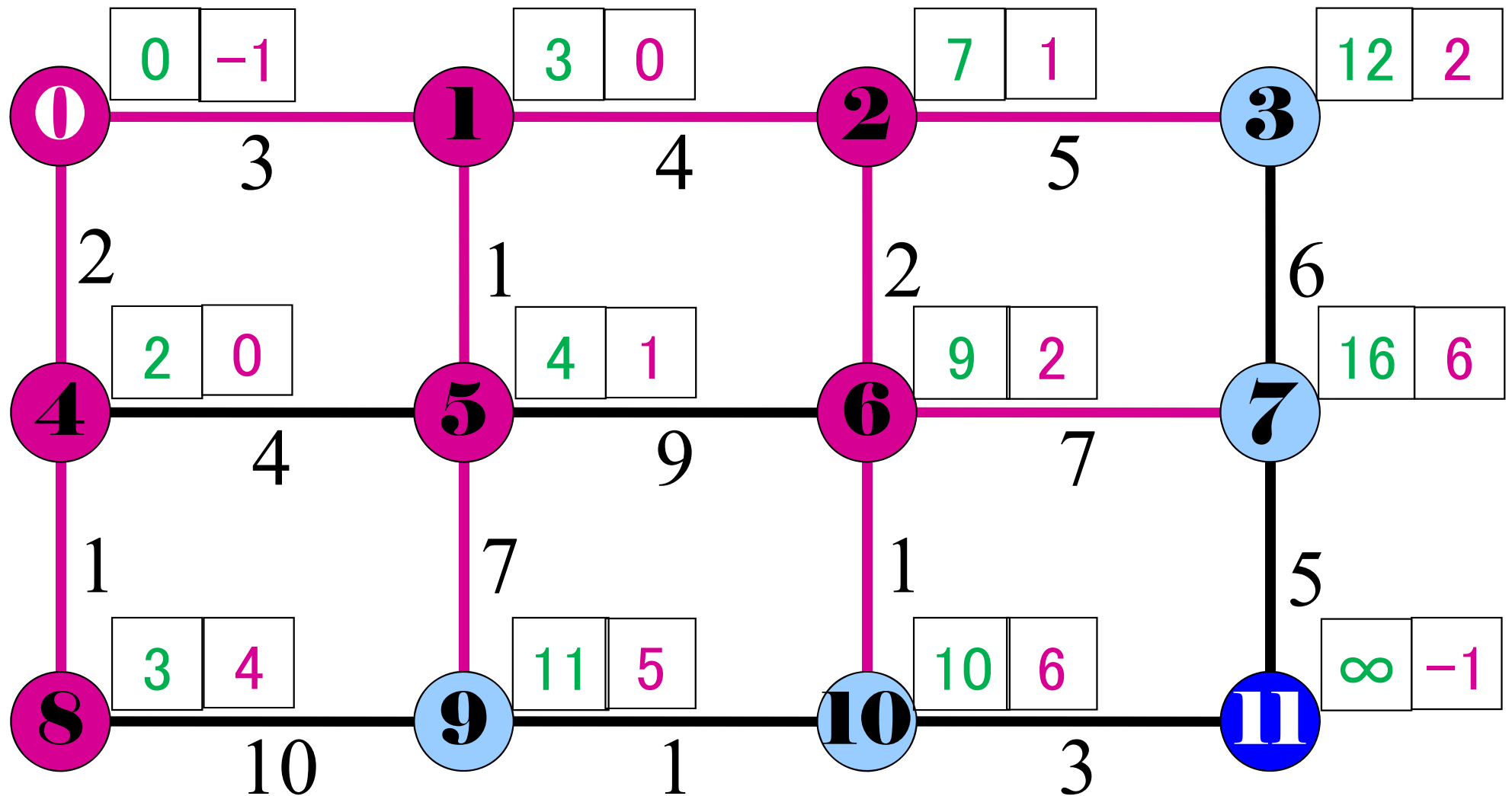
枝 e_{610} は $d(6)+c(e_{610})=9+1=10 < \infty=d(10)$ より, $d(10):=10$, $p(10):=6$ に



Dijkstra法 (更新法)

step1-3: その点 v から出る全枝 $e \in E$ の作業が全て終了したら,
その点 $v \in N$ を未確定点集合 N から除去する

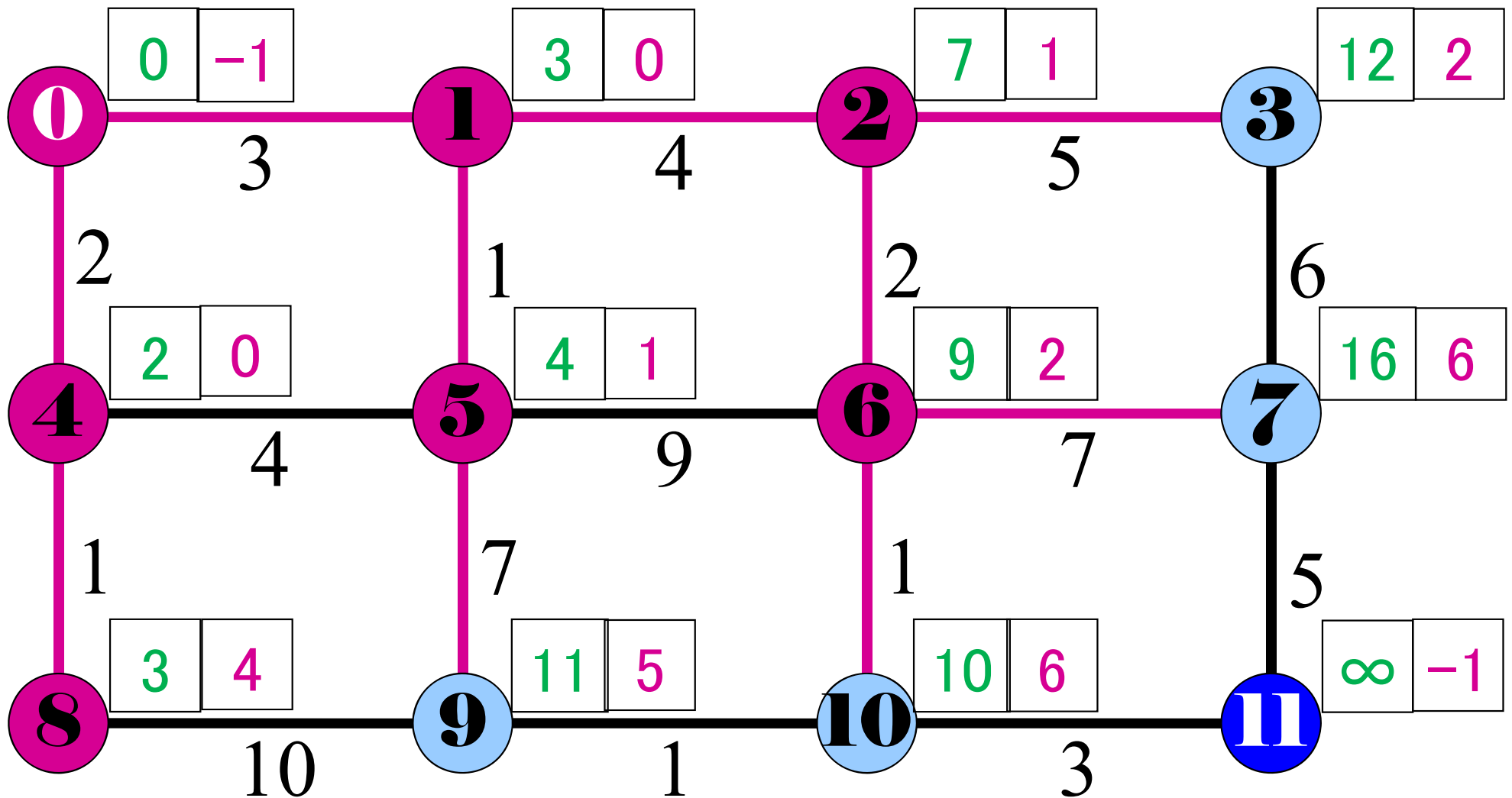
未確定点集合 $N = \{3, 6, 7, 9, 10, 11\}$
 $\rightarrow N = \{3, 7, 9, 10, 11\}$



Dijkstra法 (終了判定)

step2:未確定点集合 N が空(くう) ($N=\emptyset$)になったら終了. そうでなければ step1-1 へ戻る

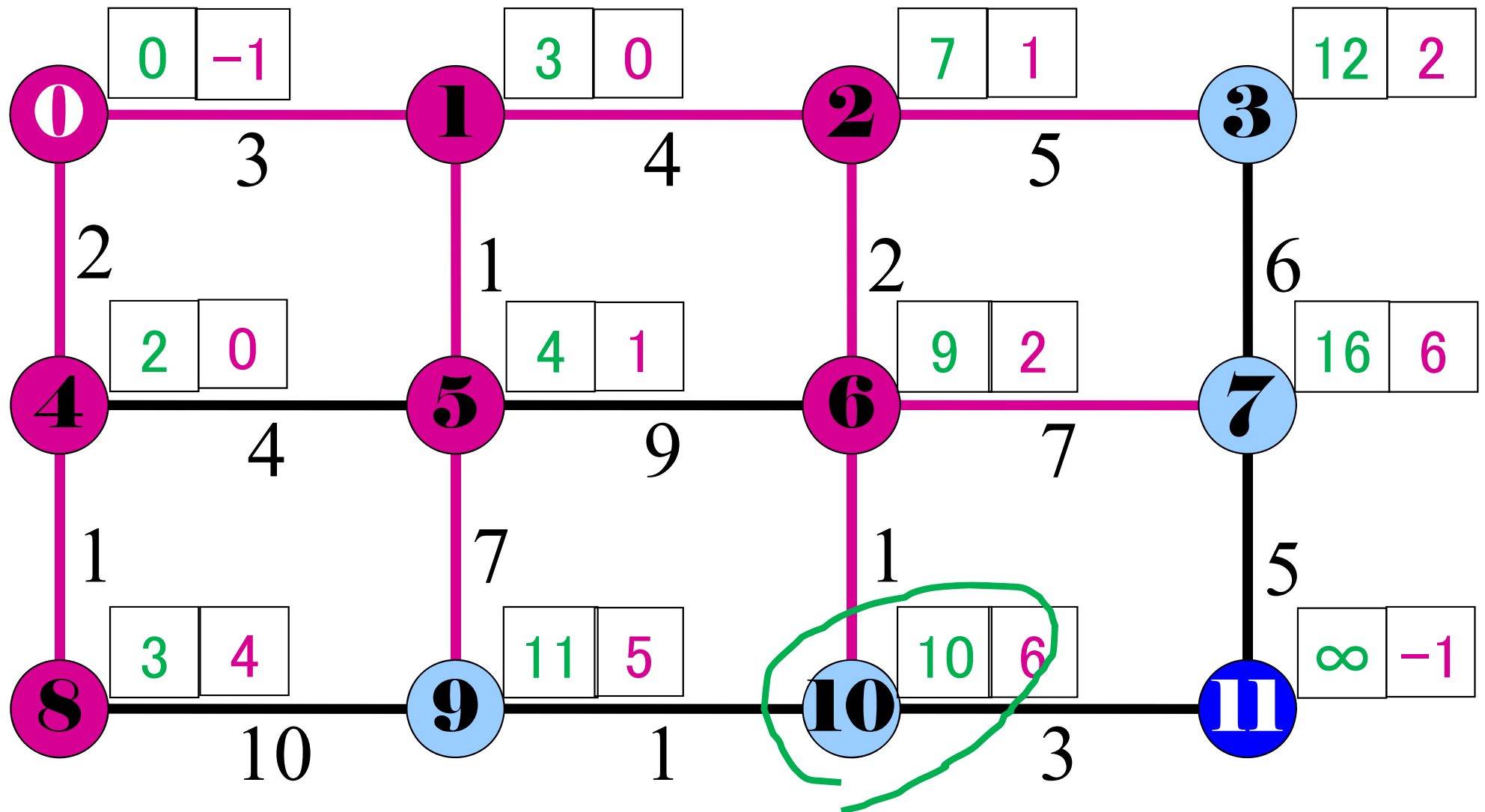
未確定点集合 $N=\{3,7,9,10,11\} \neq \emptyset$ より step1-1 へ戻る



Dijkstra法 (更新法)

step1-1: 未確定点 $v \in N$ について $d(v)$ が最小の点を見つける

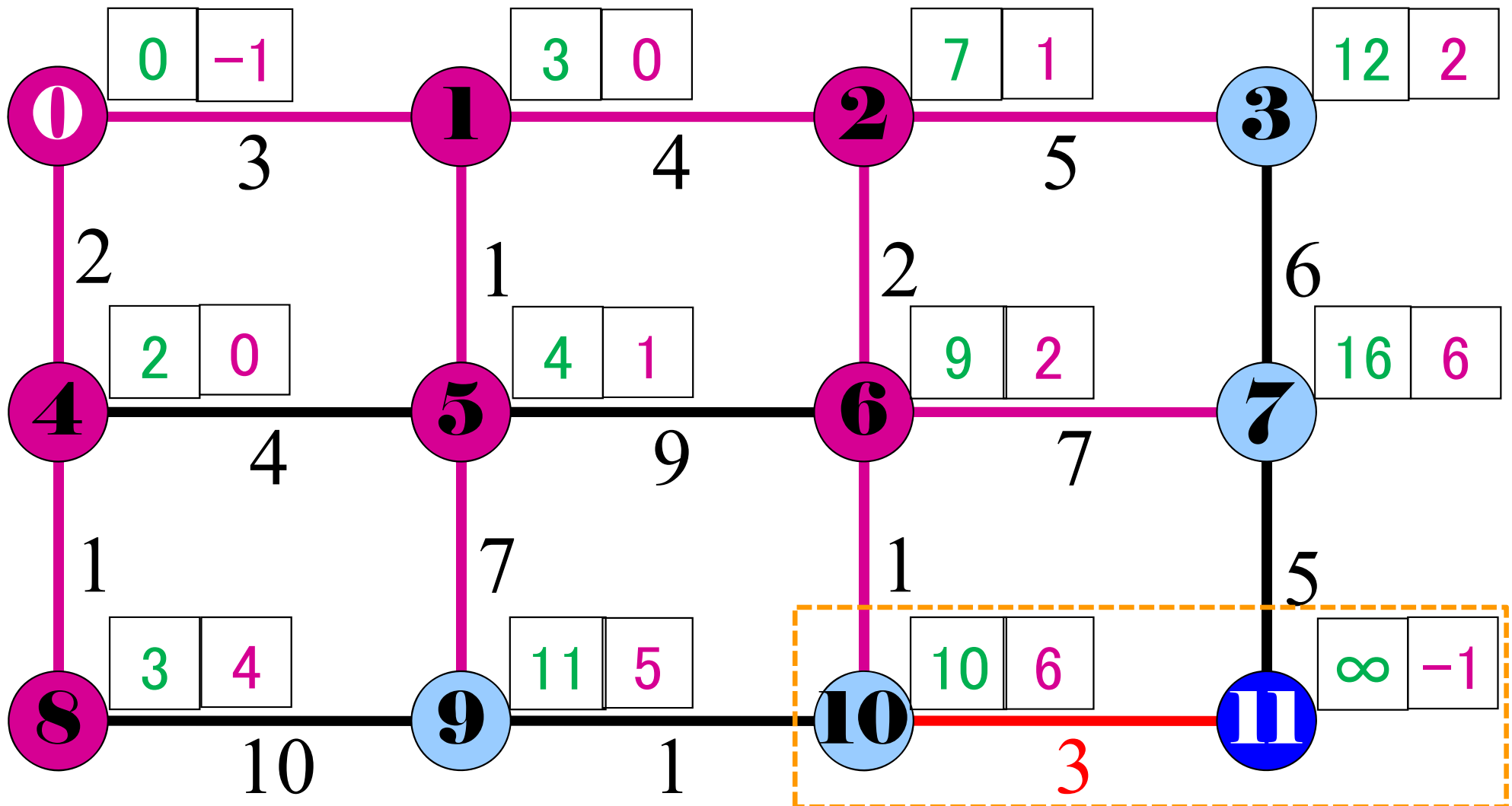
未確定点集合 $N = \{3, 7, 9, 10, 11\}$ で $d(\cdot)$ 最小点 10 を見つけた



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$
+ 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、
もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

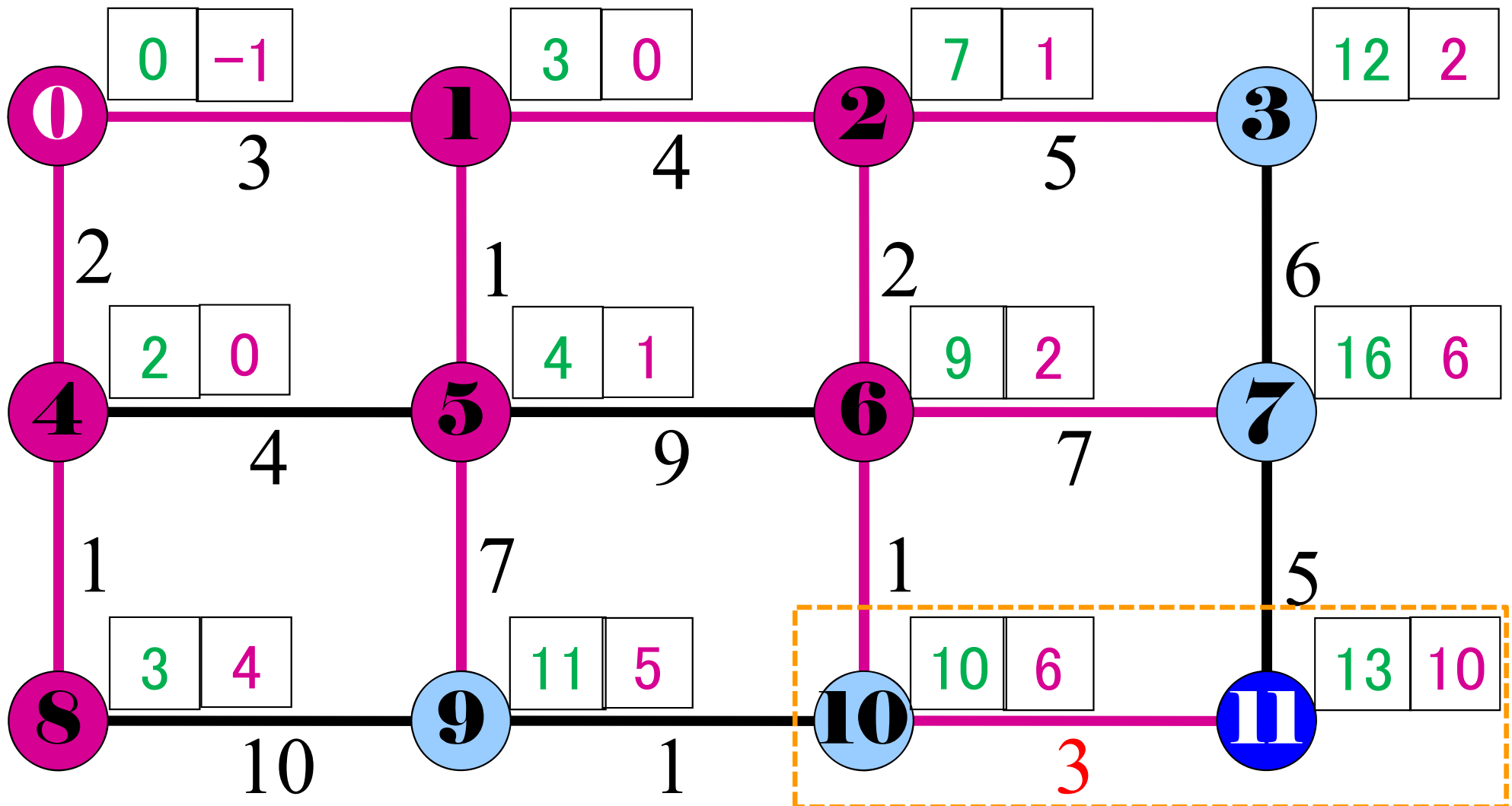
枝 e_{1011} は $d(10)+c(e_{1011})=10+3=13 < \infty = d(11)$ より $d(11):=13$, $p(11):=10$



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$
+ 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、
もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

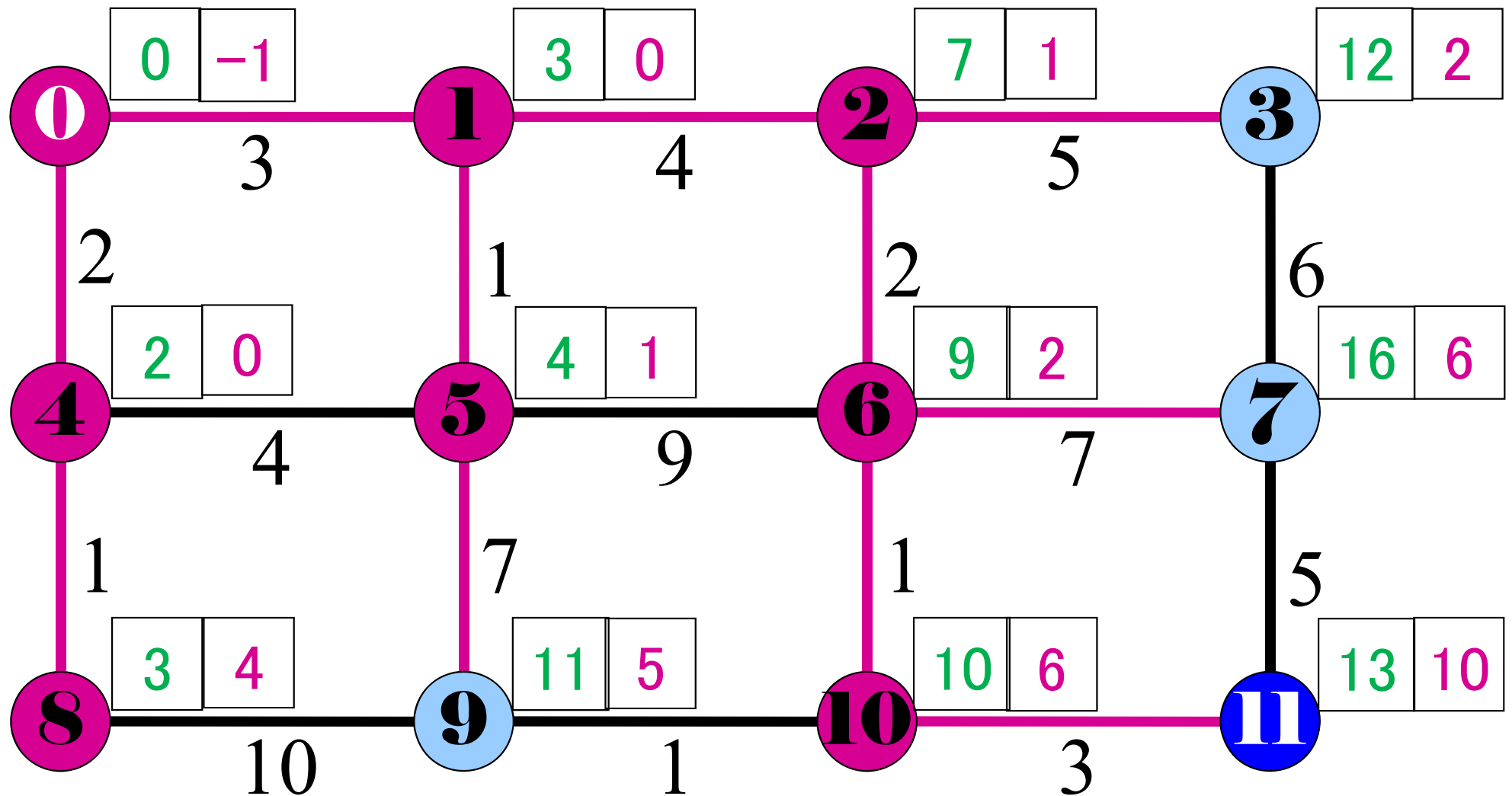
枝 e_{1011} は $d(10)+c(e_{1011})=10+3=13 < \infty = d(11)$ より $d(11):=13$, $p(11):=10$



Dijkstra法 (更新法)

step1-3: その点 v から出る全枝 $e \in E$ の作業が全て終了したら,
その点 $v \in N$ を未確定点集合 N から除去する

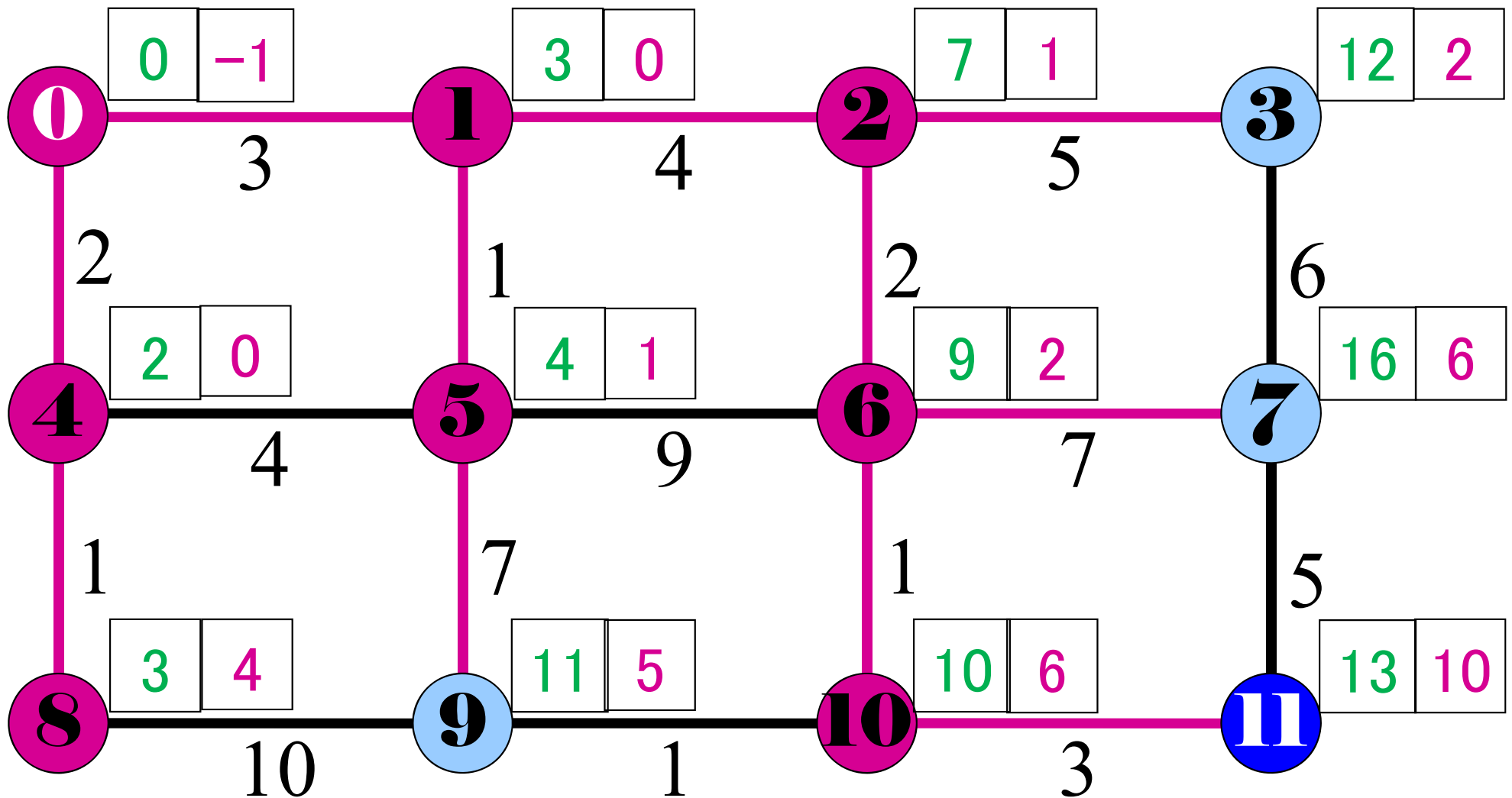
未確定点集合 $N = \{3, 7, 9, 10, 11\}$
 $\rightarrow N = \{3, 7, 9, 11\}$



Dijkstra法 (終了判定)

step2:未確定点集合 N が空(くう) ($N=\emptyset$)になったら終了. そうでなければ step1-1 へ戻る

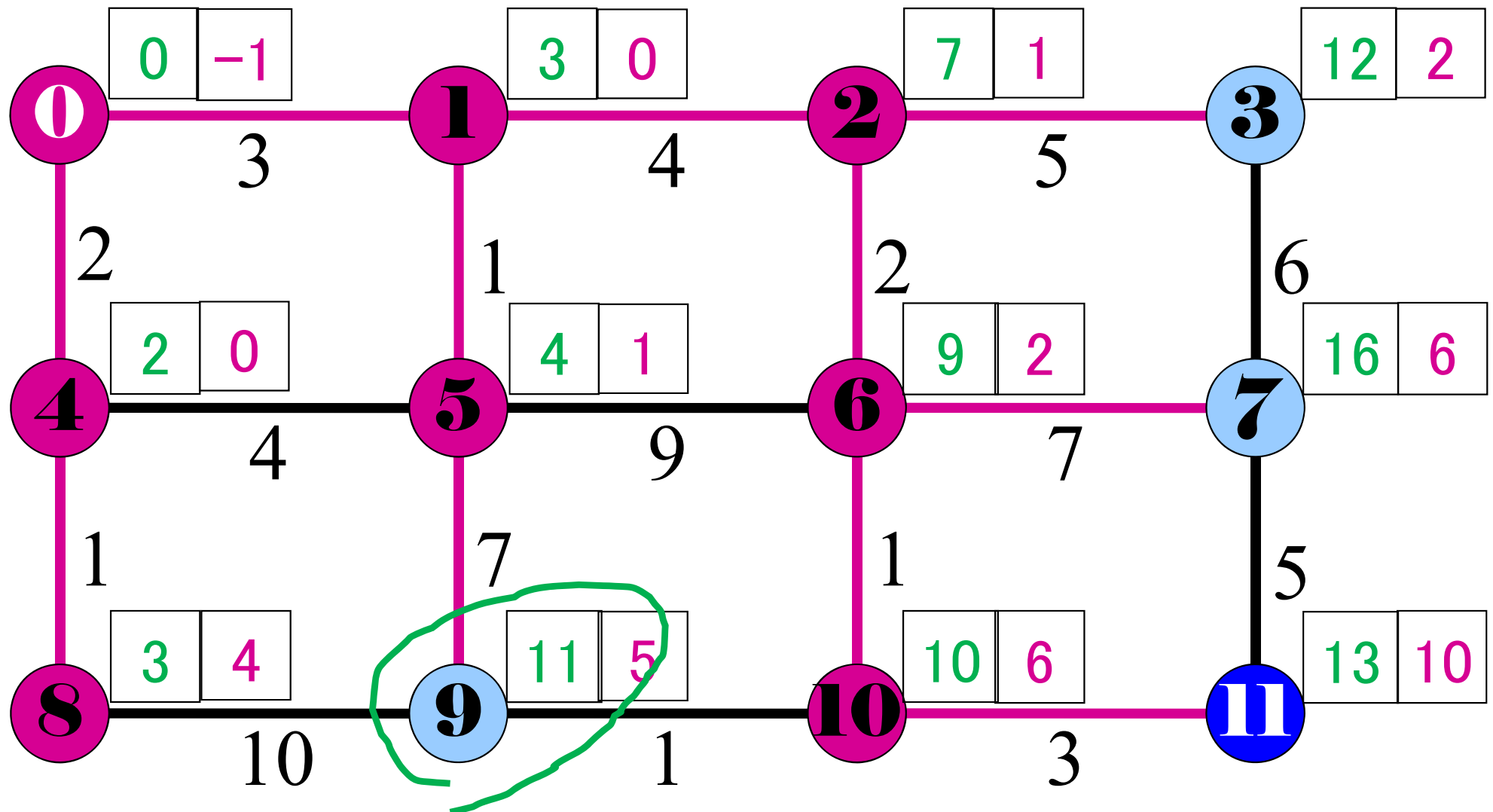
未確定点集合 $N=\{3,7,9,11\} \neq \emptyset$ より step1-1 へ戻る



Dijkstra法 (更新法)

step1-1: 未確定点 $v \in N$ について $d(v)$ が最小の点を見つける

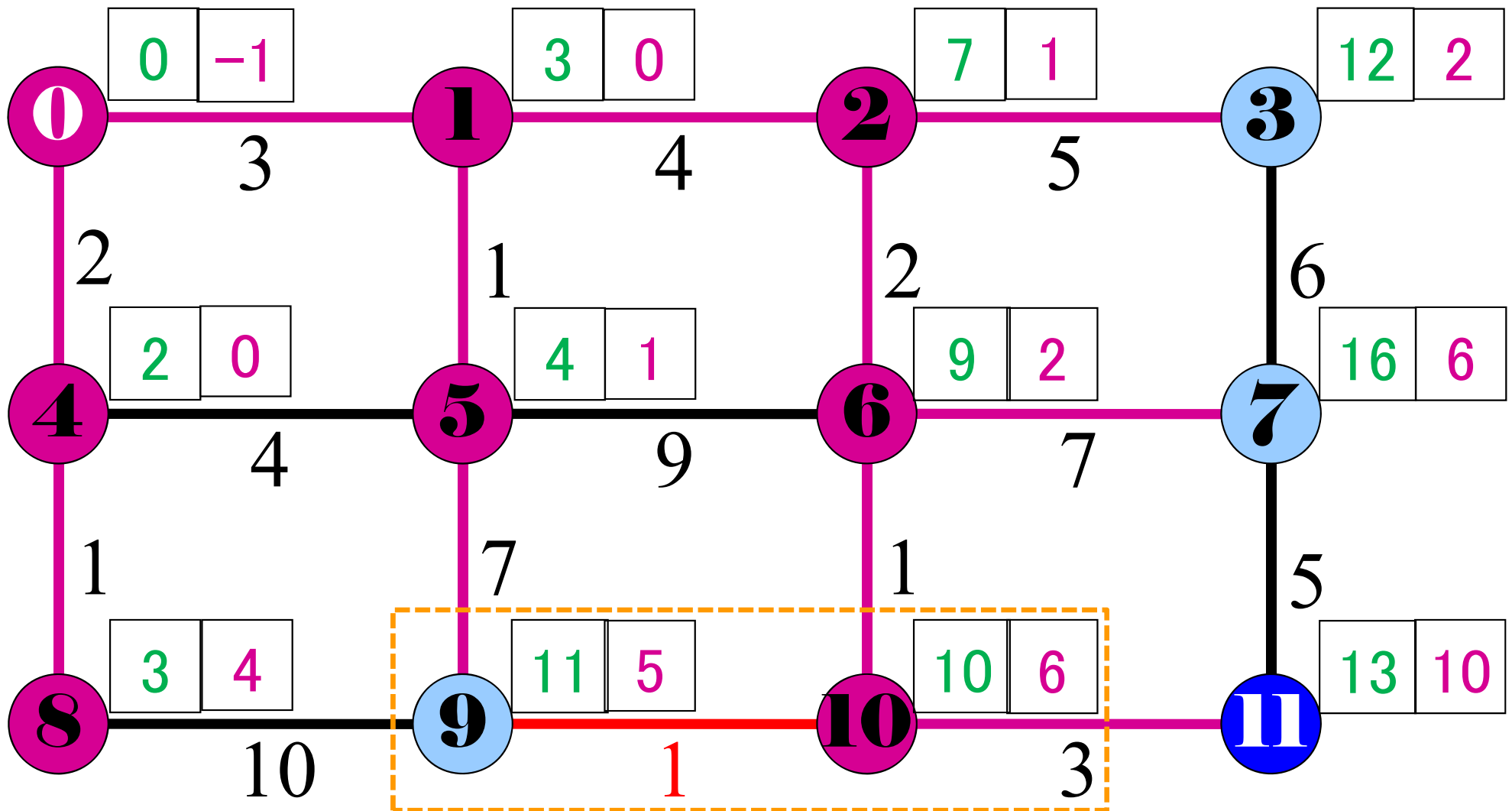
未確定点集合 $N = \{3, 7, 9, 11\}$ で $d(\cdot)$ 最小点 9 を見つけた



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$
+ 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、
もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

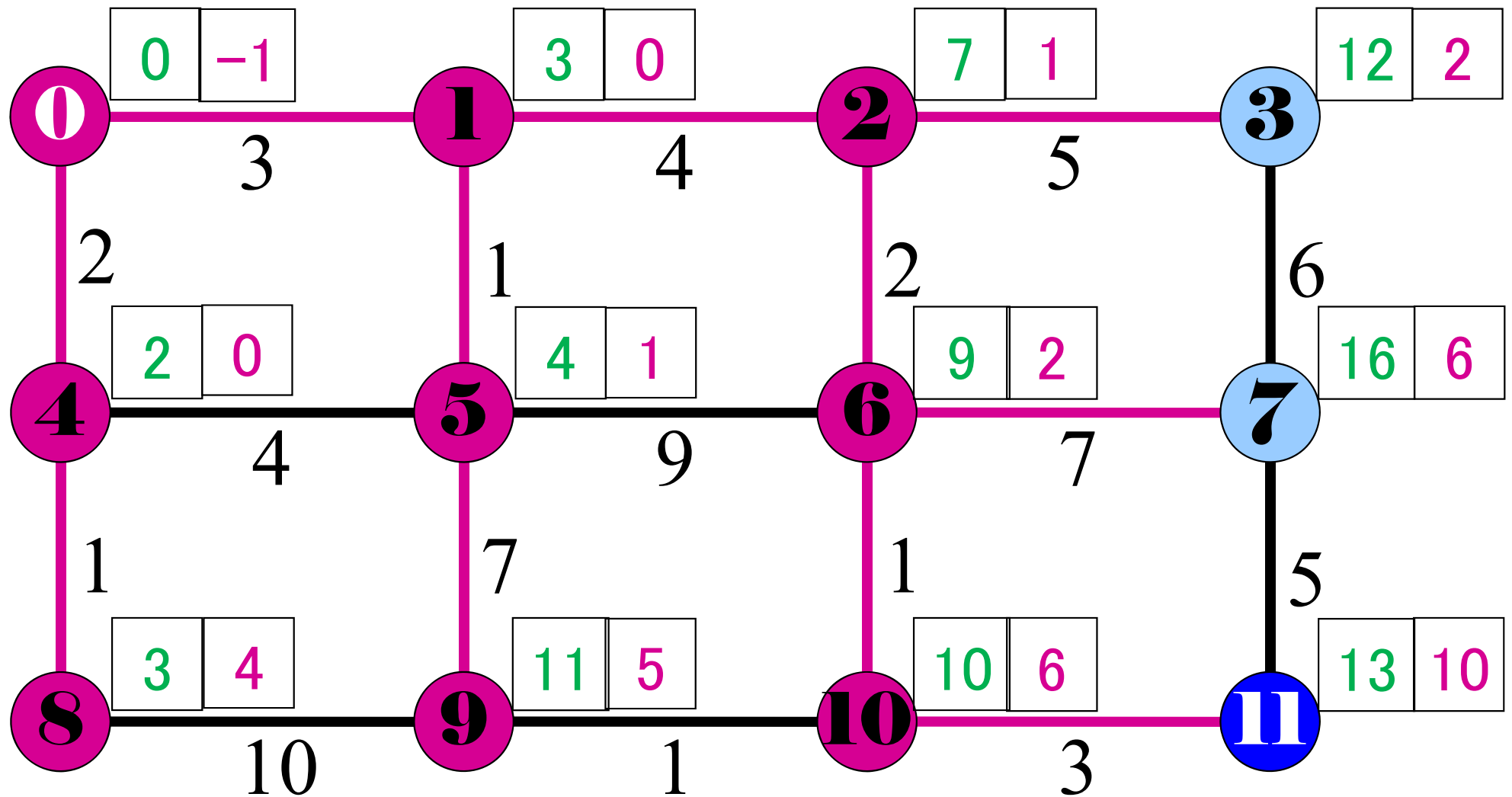
枝 e_{910} は $d(9)+c(e_{910})=11+1=12 > 10=d(10)$ より、更新しない



Dijkstra法 (更新法)

step1-3: その点 v から出る全枝 $e \in E$ の作業が全て終了したら,
その点 $v \in N$ を未確定点集合 N から除去する

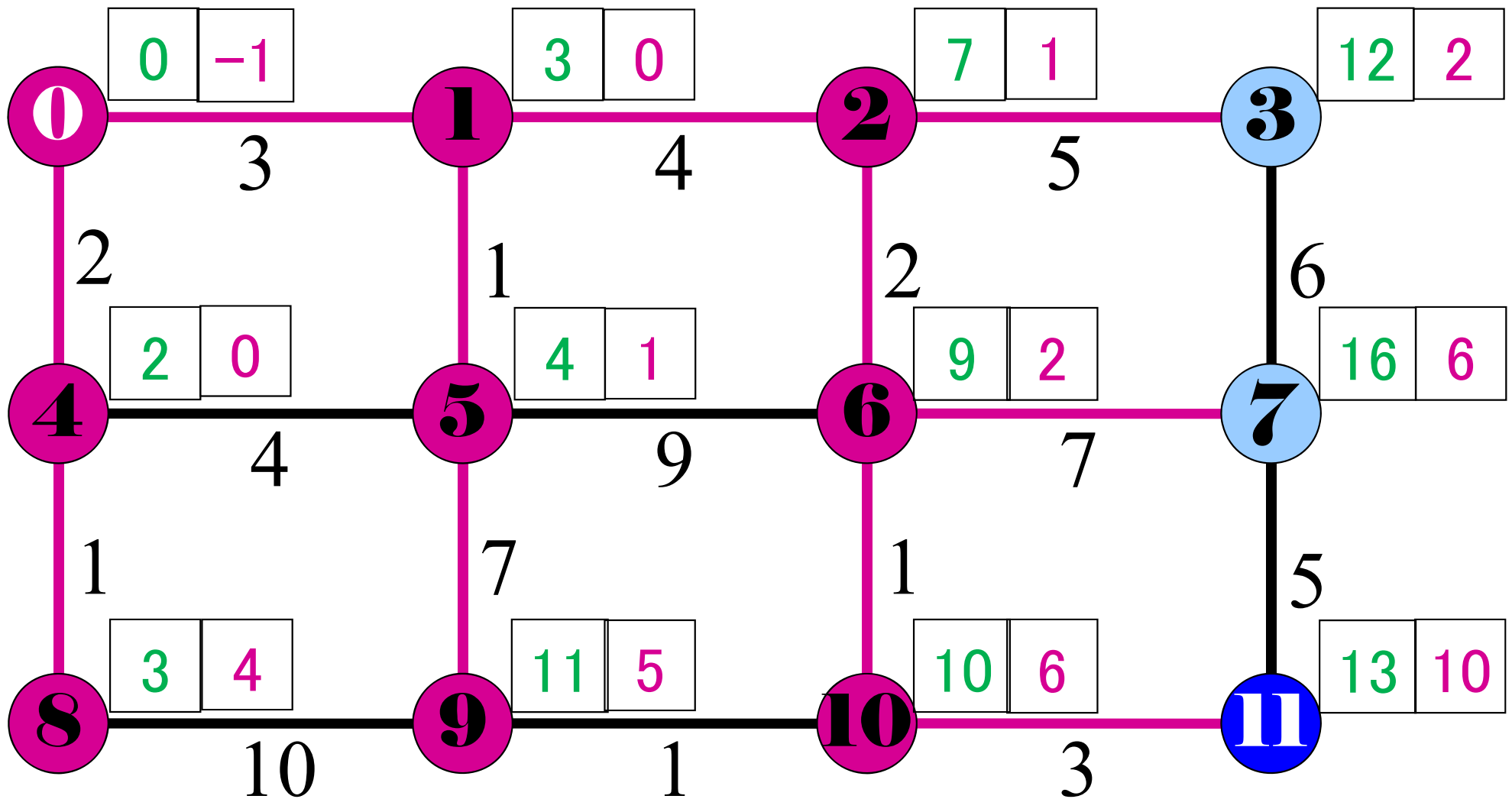
未確定点集合 $N = \{3, 7, 9, 11\}$
 $\rightarrow N = \{3, 7, 11\}$



Dijkstra法 (終了判定)

step2:未確定点集合 N が空(くう) ($N=\emptyset$)になったら終了. そうでなければ step1-1 へ戻る

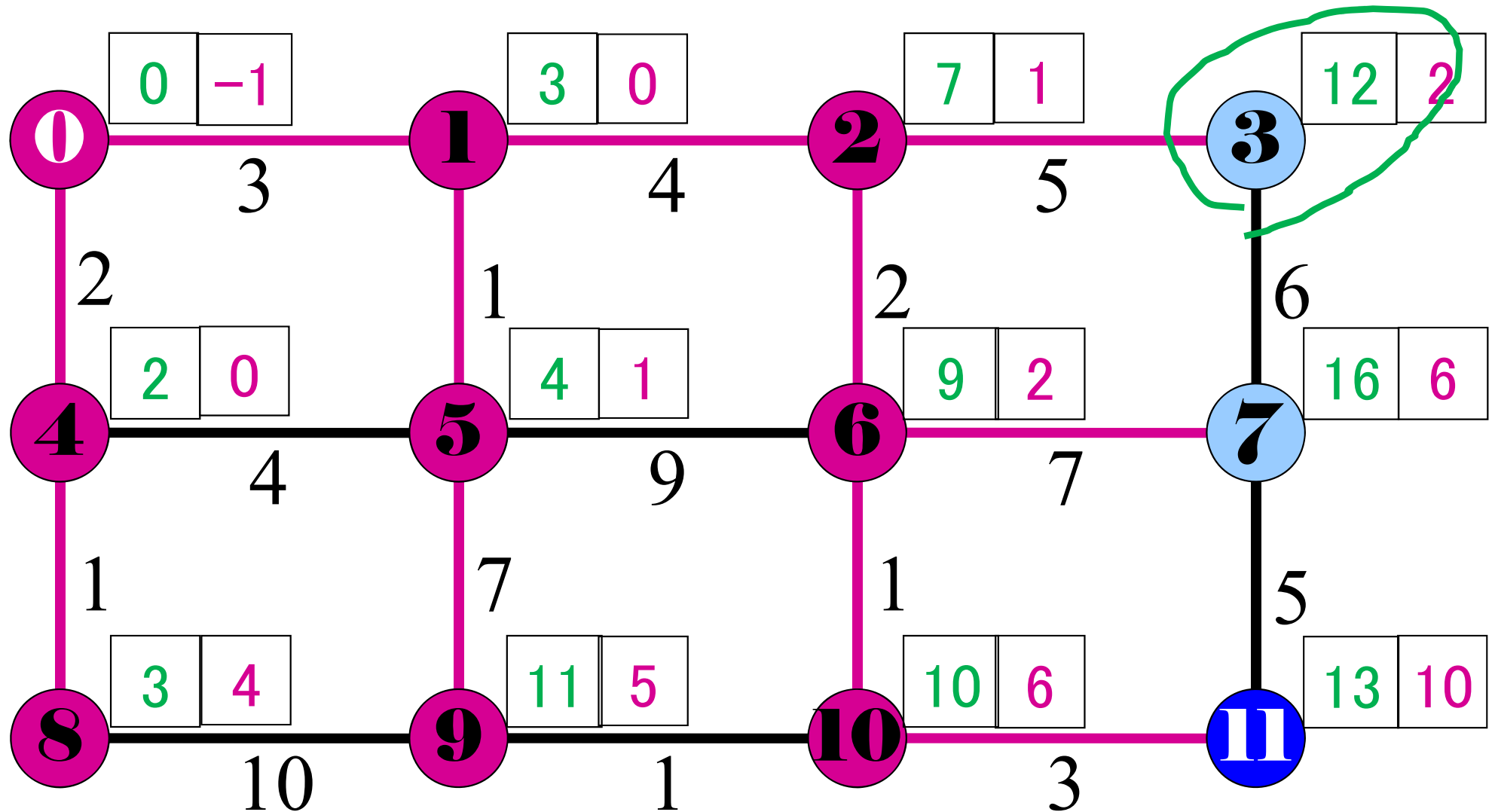
未確定点集合 $N=\{3,7,11\} \neq \emptyset$ より step1-1 へ戻る



Dijkstra法 (更新法)

step1-1: 未確定点 $v \in N$ について $d(v)$ が最小の点を見つける

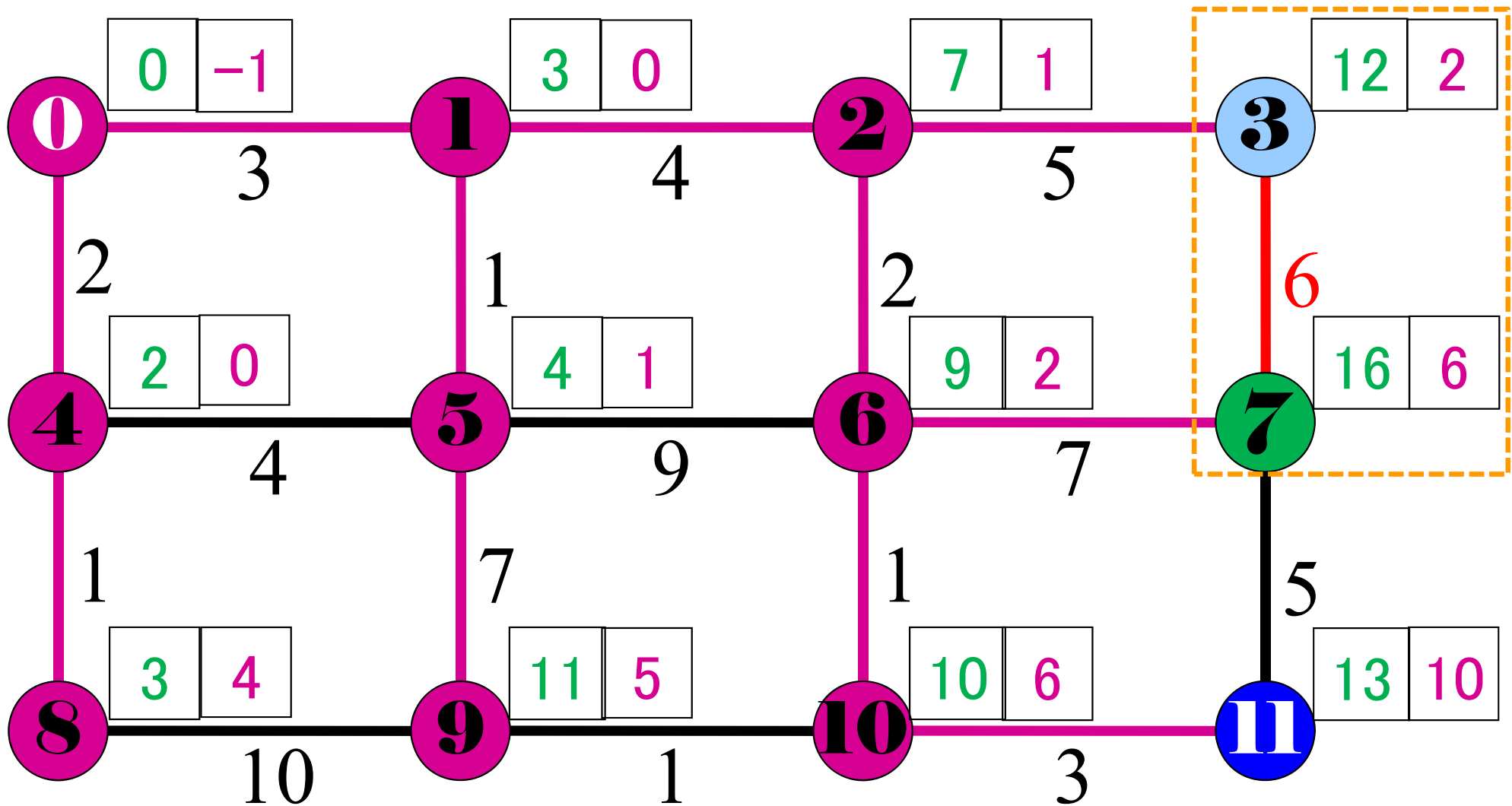
未確定点集合 $N = \{3, 7, 11\}$ で $d(\cdot)$ 最小点 3 を見つけた



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$
+ 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、
もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

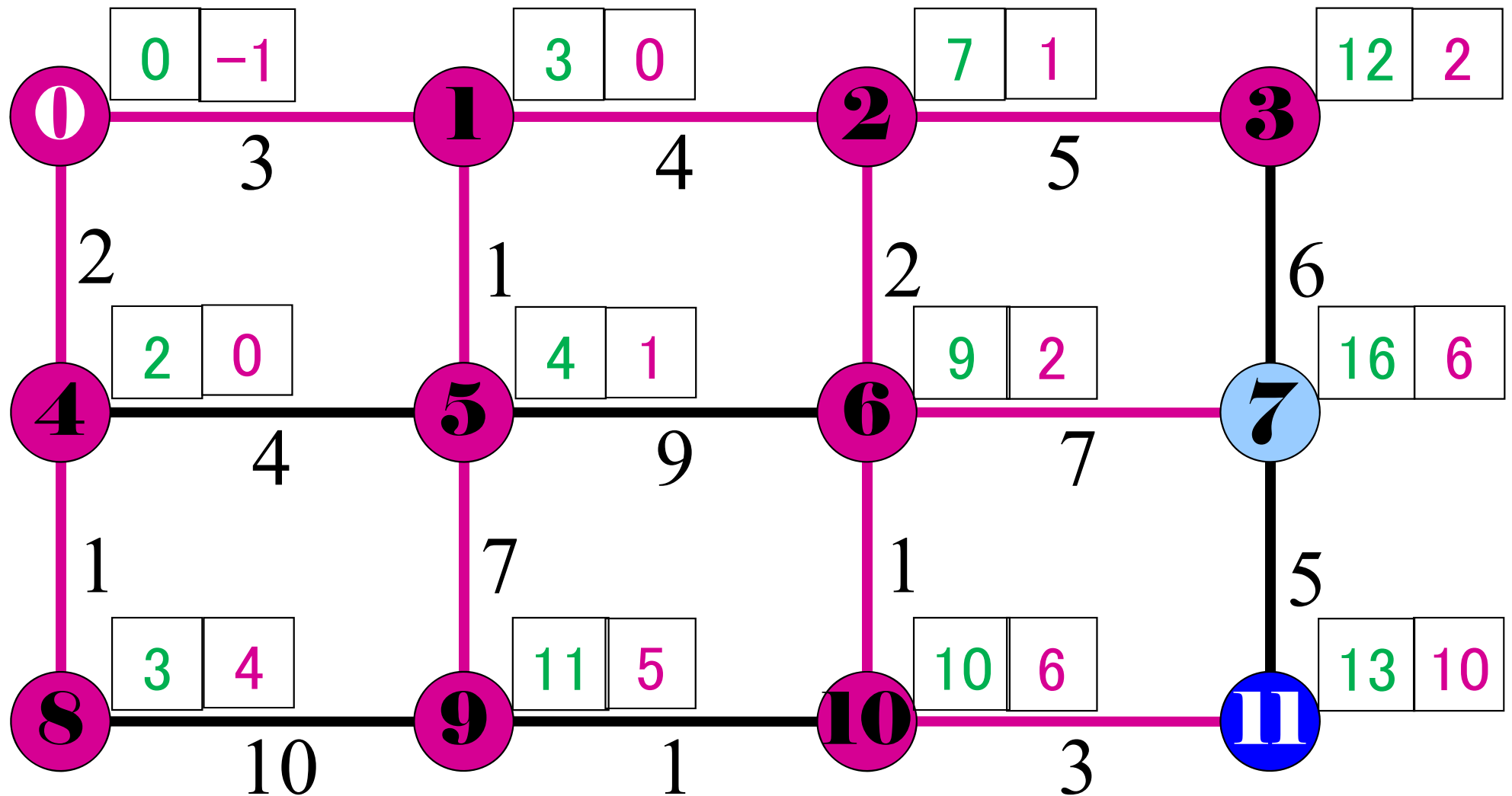
枝 e_{37} は $d(3)+c(e_{37})=12+6=18 > 16=d(7)$ より、更新しない



Dijkstra法 (更新法)

step1-3: その点 v から出る全枝 $e \in E$ の作業が全て終了したら,
その点 $v \in N$ を未確定点集合 N から除去する

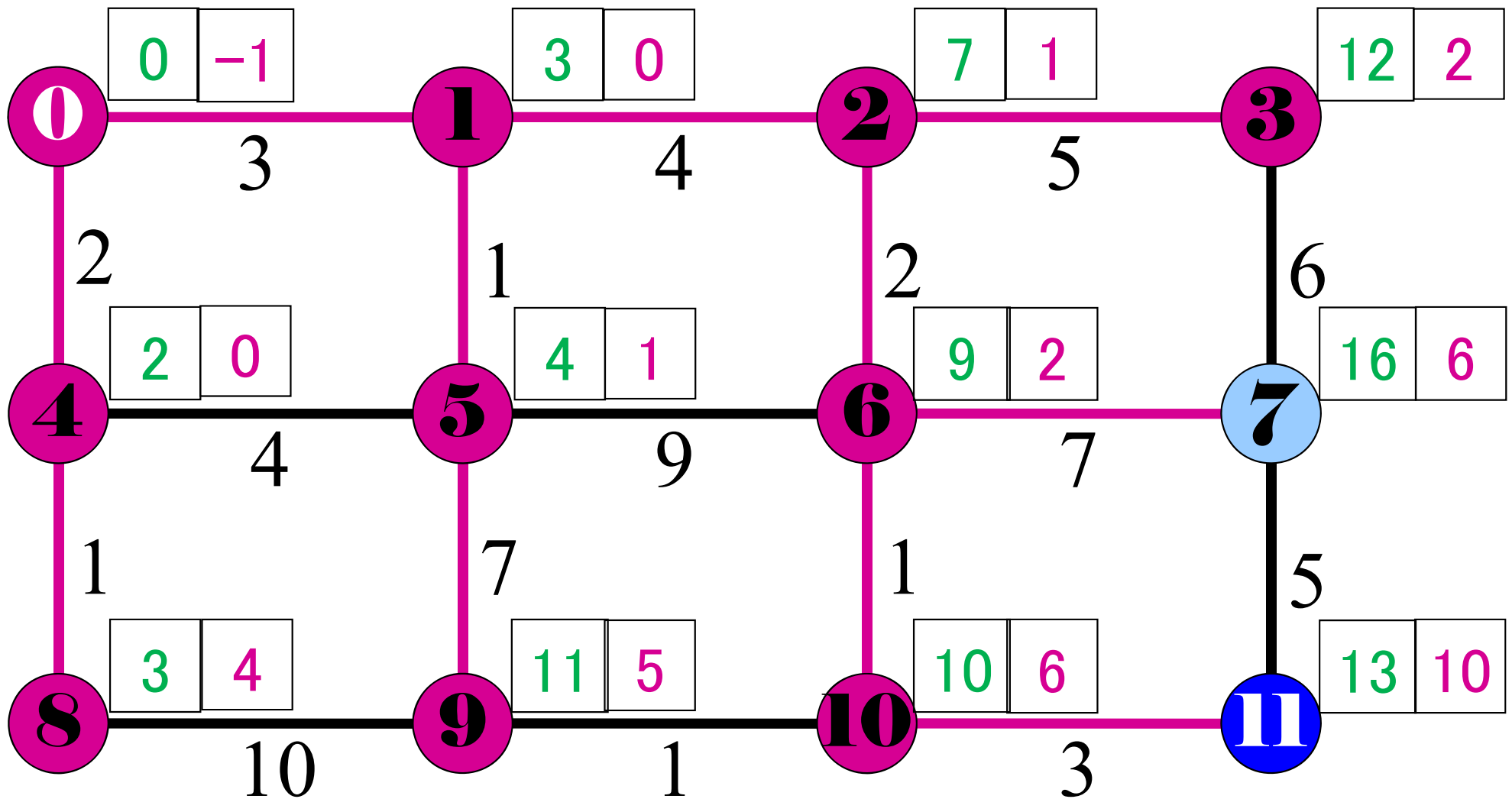
未確定点集合 $N = \{3, 7, 11\}$
 $\rightarrow N = \{7, 11\}$



Dijkstra法 (終了判定)

step2:未確定点集合 N が空(くう) ($N=\emptyset$)になったら終了. そうでなければ step1-1 へ戻る

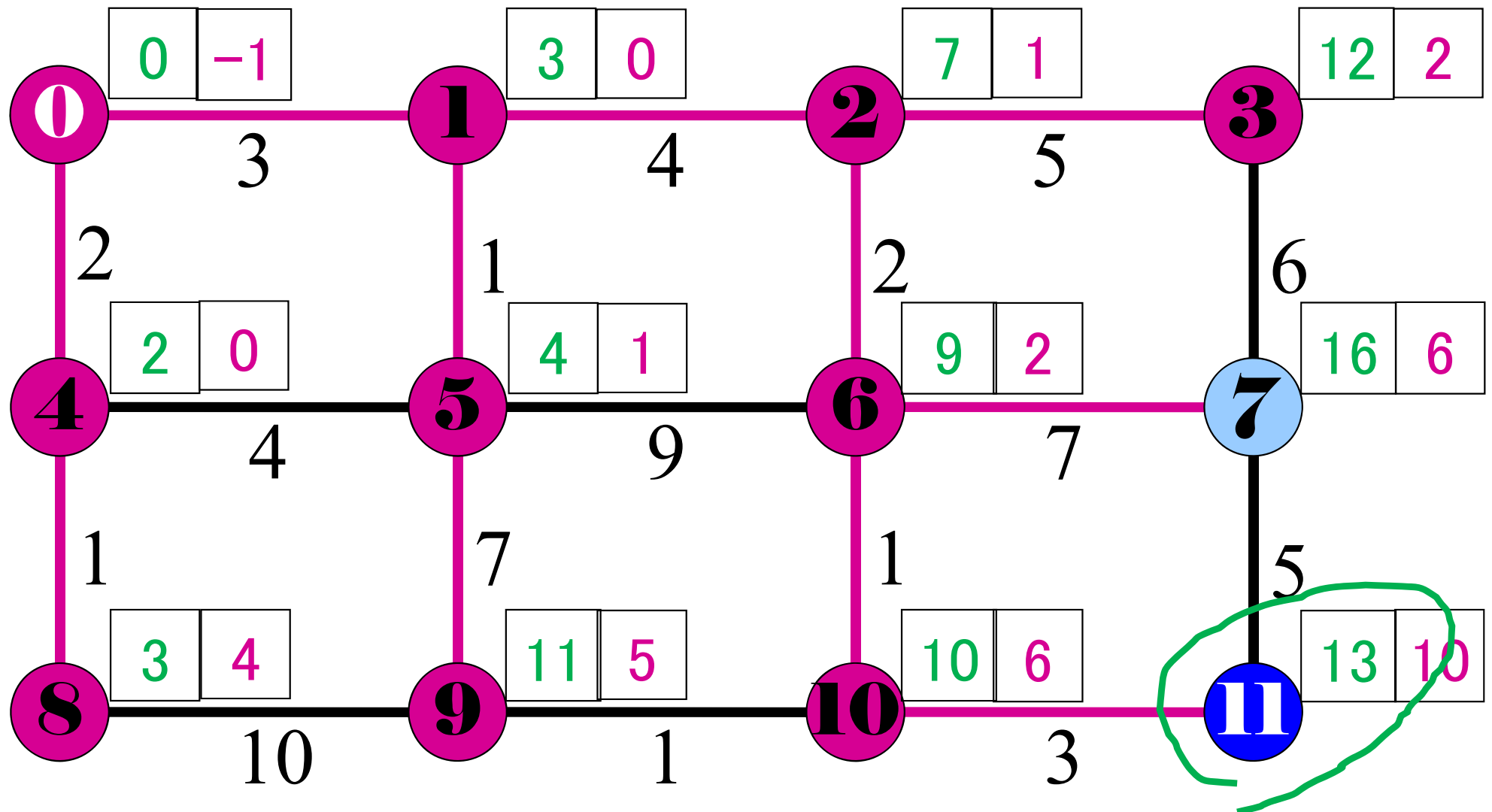
未確定点集合 $N=\{7,11\} \neq \emptyset$ より step1-1 へ戻る



Dijkstra法 (更新法)

step1-1: 未確定点 $v \in N$ について $d(v)$ が最小の点を見つける

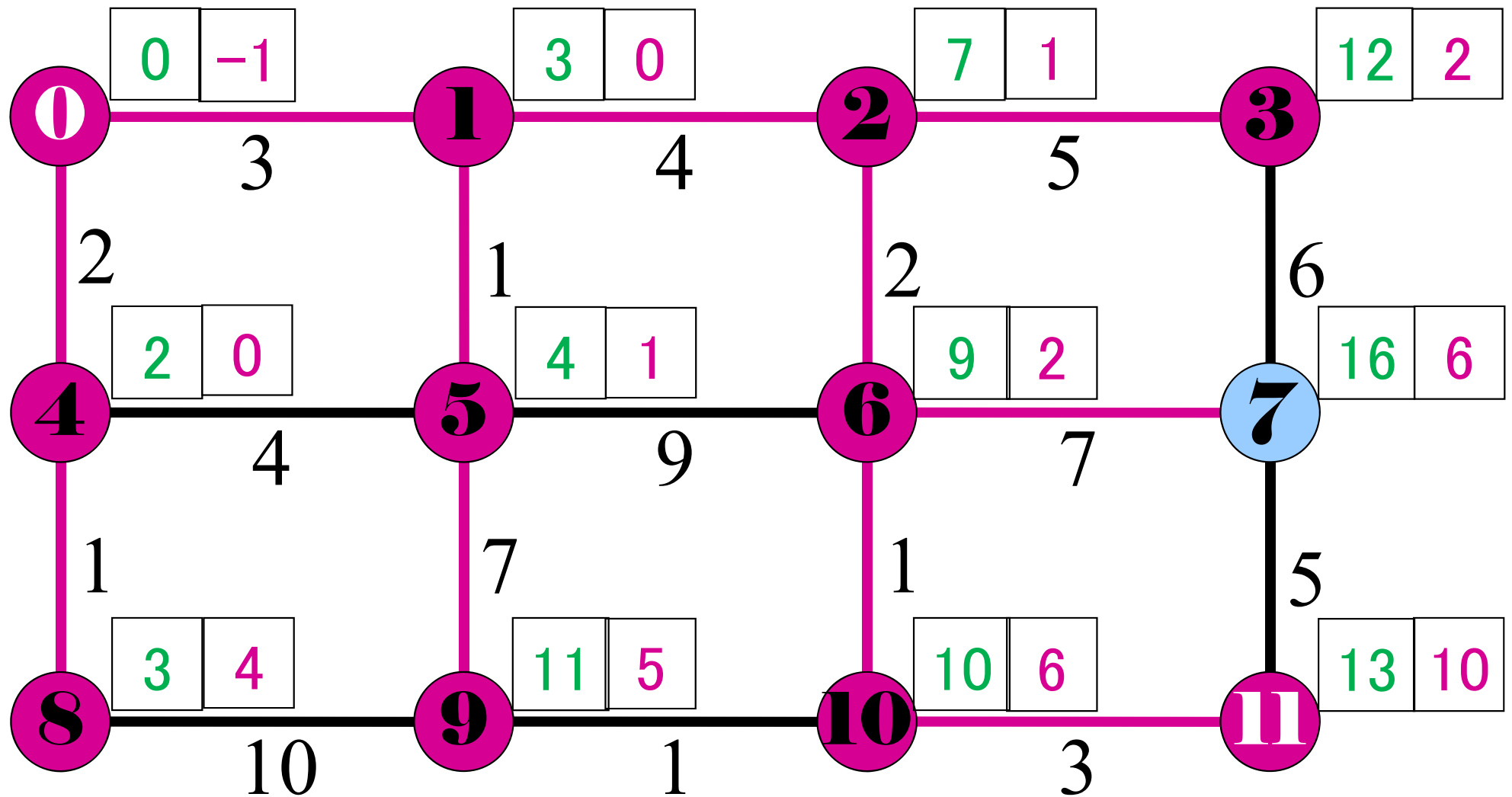
未確定点集合 $N=\{7,11\}$ で $d(\cdot)$ 最小点 11 を見つけた



Dijkstra法 (更新法)

step1-3: その点 v から出る全枝 $e \in E$ の作業が全て終了したら,
その点 $v \in N$ を未確定点集合 N から除去する

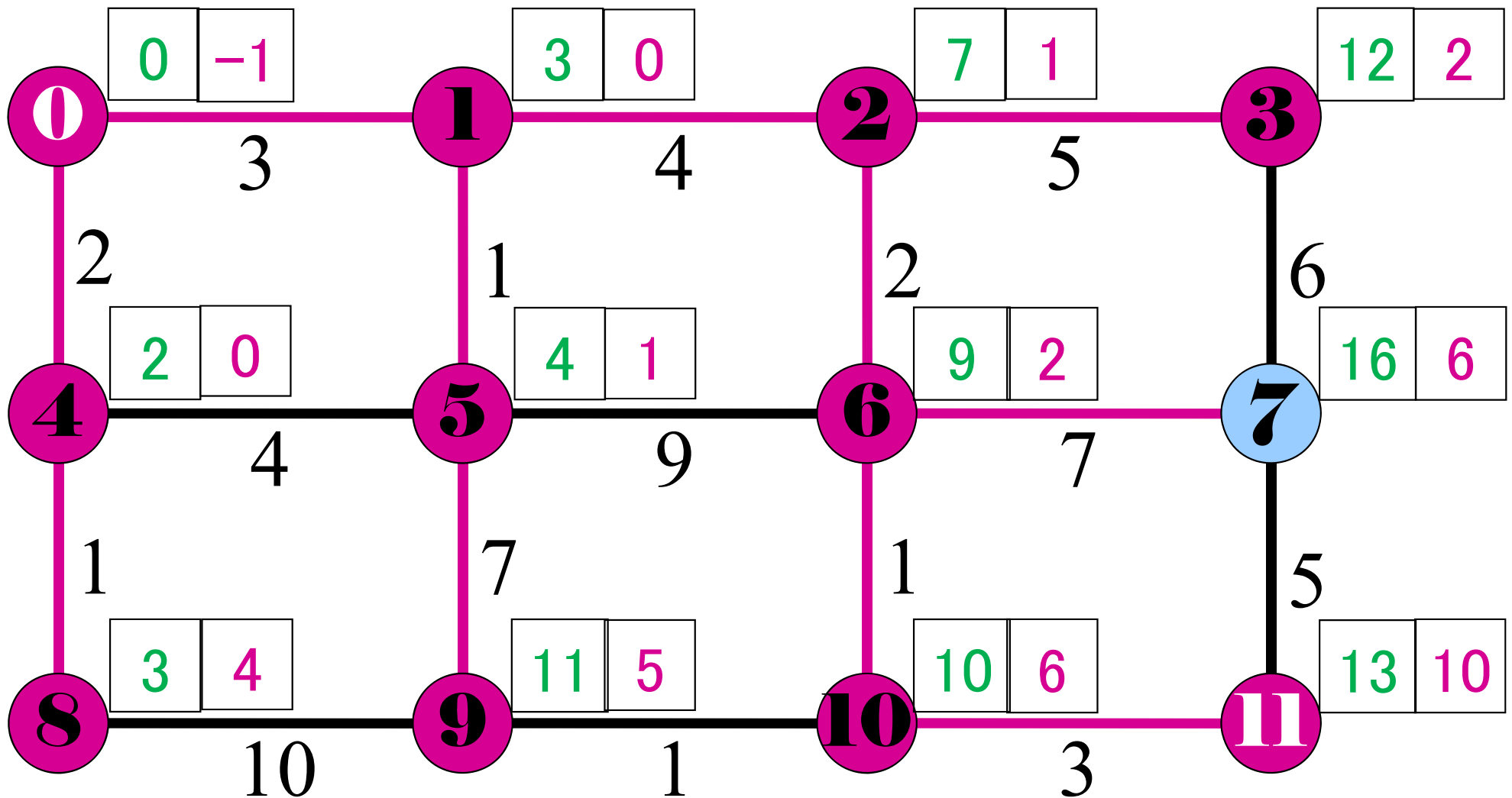
未確定点集合 $N = \{7, 11\}$
 $\rightarrow N = \{7\}$



Dijkstra法 (終了判定)

step2:未確定点集合 N が空(くう) ($N=\emptyset$)になったら終了. そうでなければ step1-1 へ戻る

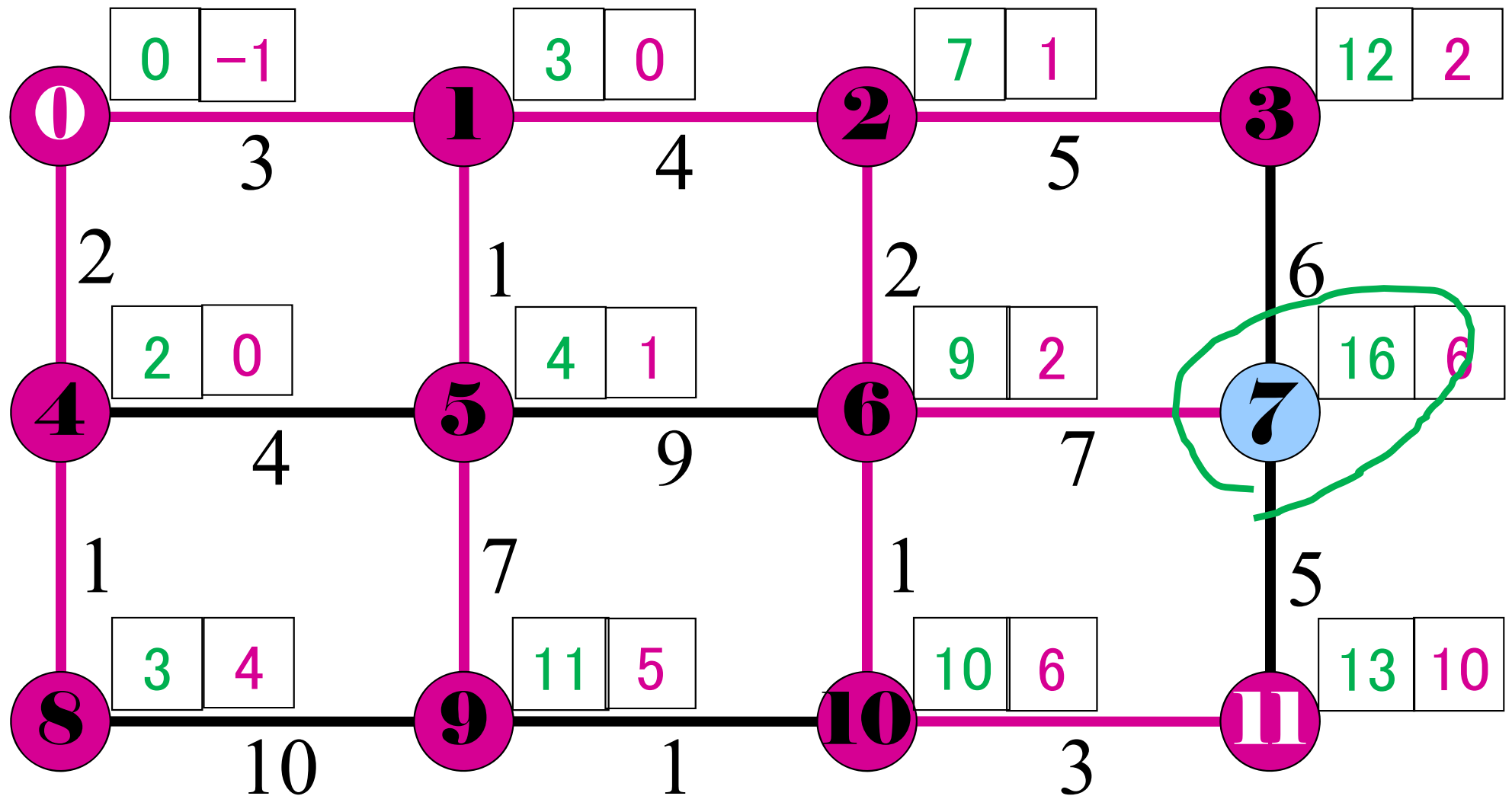
未確定点集合 $N=\{7\} \neq \emptyset$ より step1-1 へ戻る



Dijkstra法 (更新法)

step1-1: 未確定点 $v \in N$ について $d(v)$ が最小の点を見つける

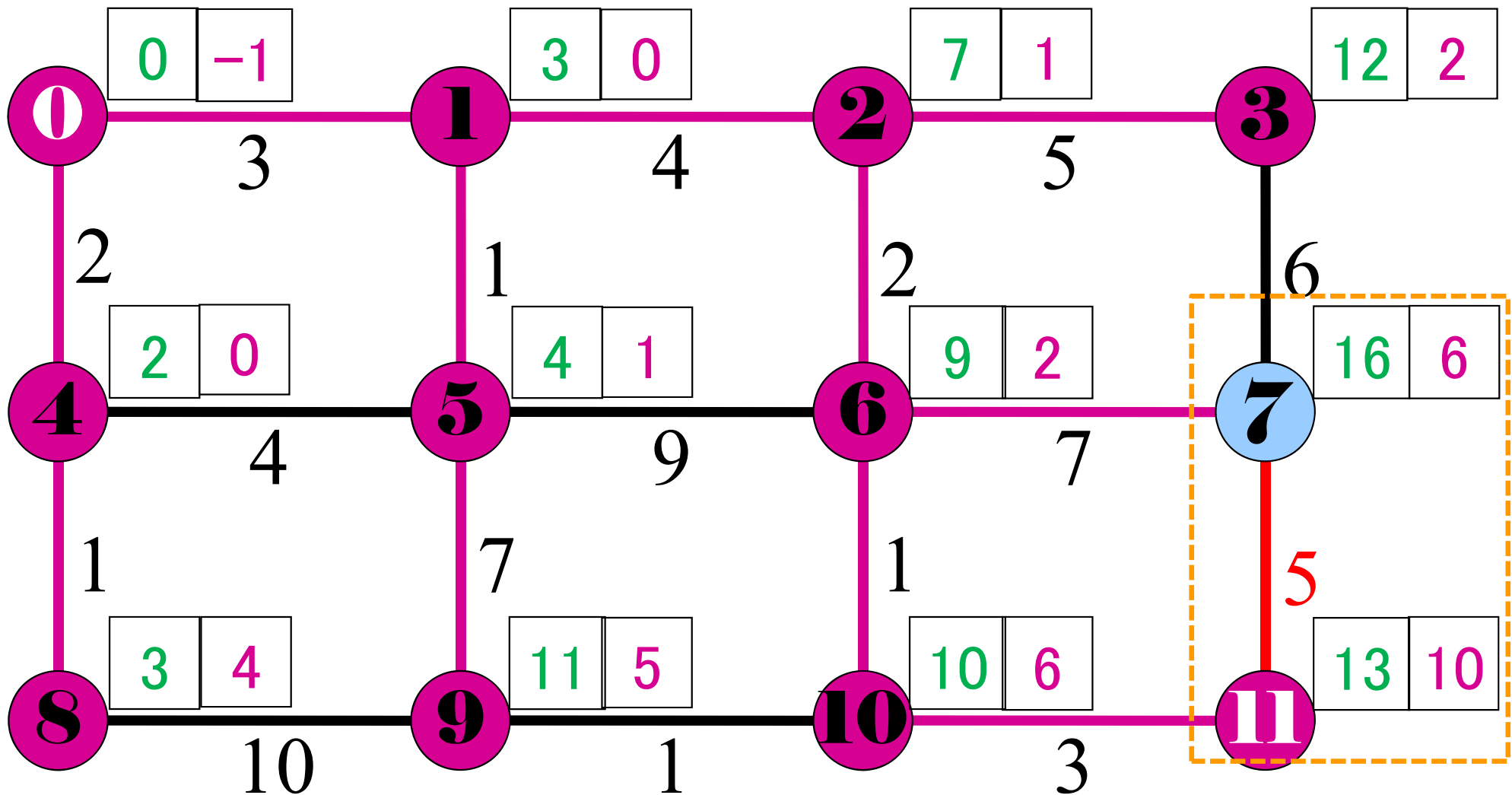
未確定点集合 $N=\{7\}$ で $d(\cdot)$ 最小点 7 を見つけた



Dijkstra法 (更新法)

step1-2: その点 v から出る全枝 $e \in E$ について「距離ラベル $d(v)$
+ 枝コスト $c(e)$ 」を計算し、枝先点 u の距離ラベル $d(u)$ と比較し、
もし $d(v)+c(e) < d(u)$ なら、 $d(u) := d(v)+c(e)$, $p(u) := v$ とする

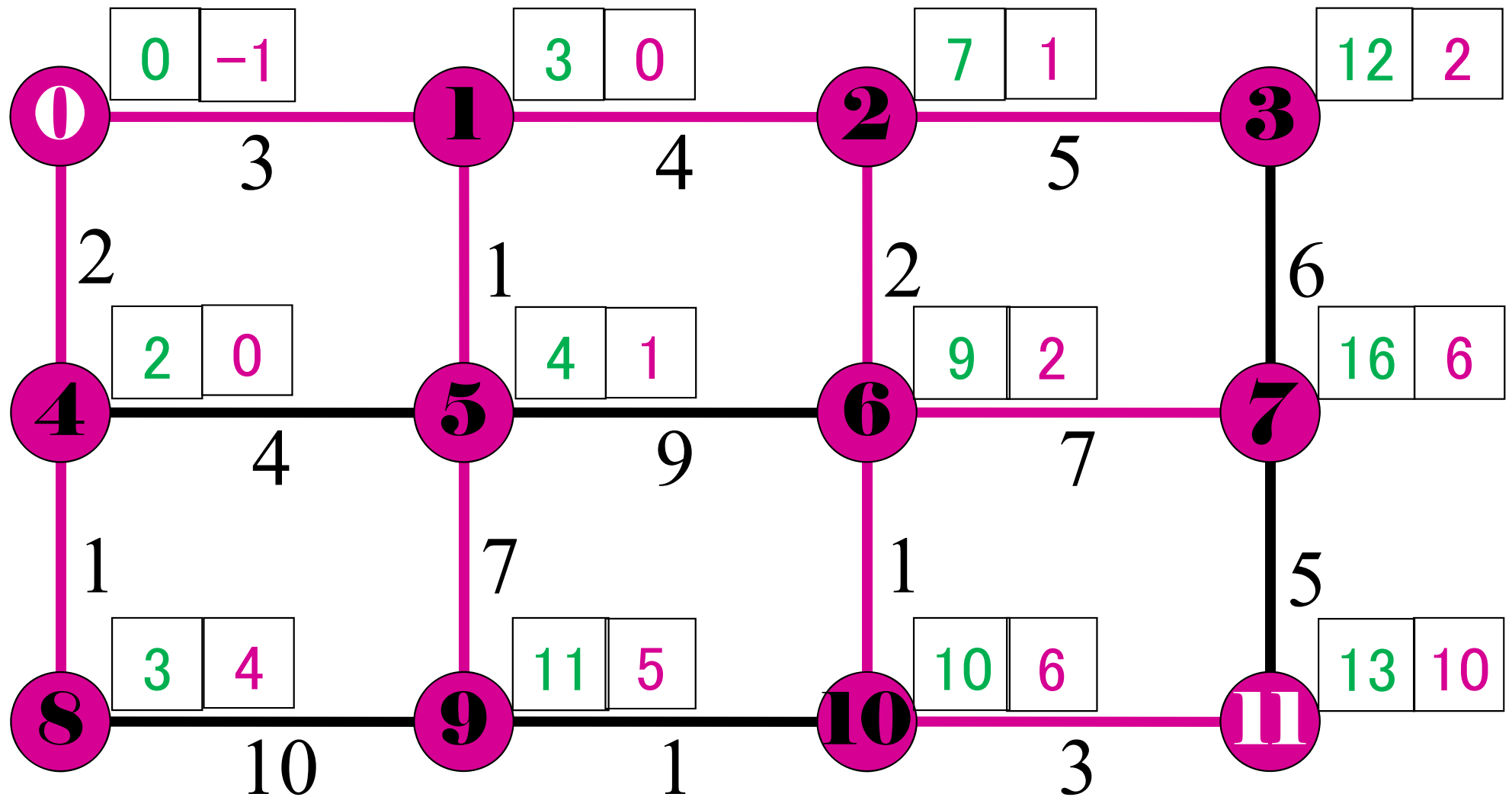
枝 e_{711} は $d(7)+c(e_{711})=16+5=21 > 13=d(11)$ より、更新しない



Dijkstra法 (更新法)

step1-3: その点 v から出る全枝 $e \in E$ の作業が全て終了したら,
その点 $v \in N$ を未確定点集合 N から除去する

未確定点集合 $N = \{7\}$
 $\rightarrow N = \emptyset$

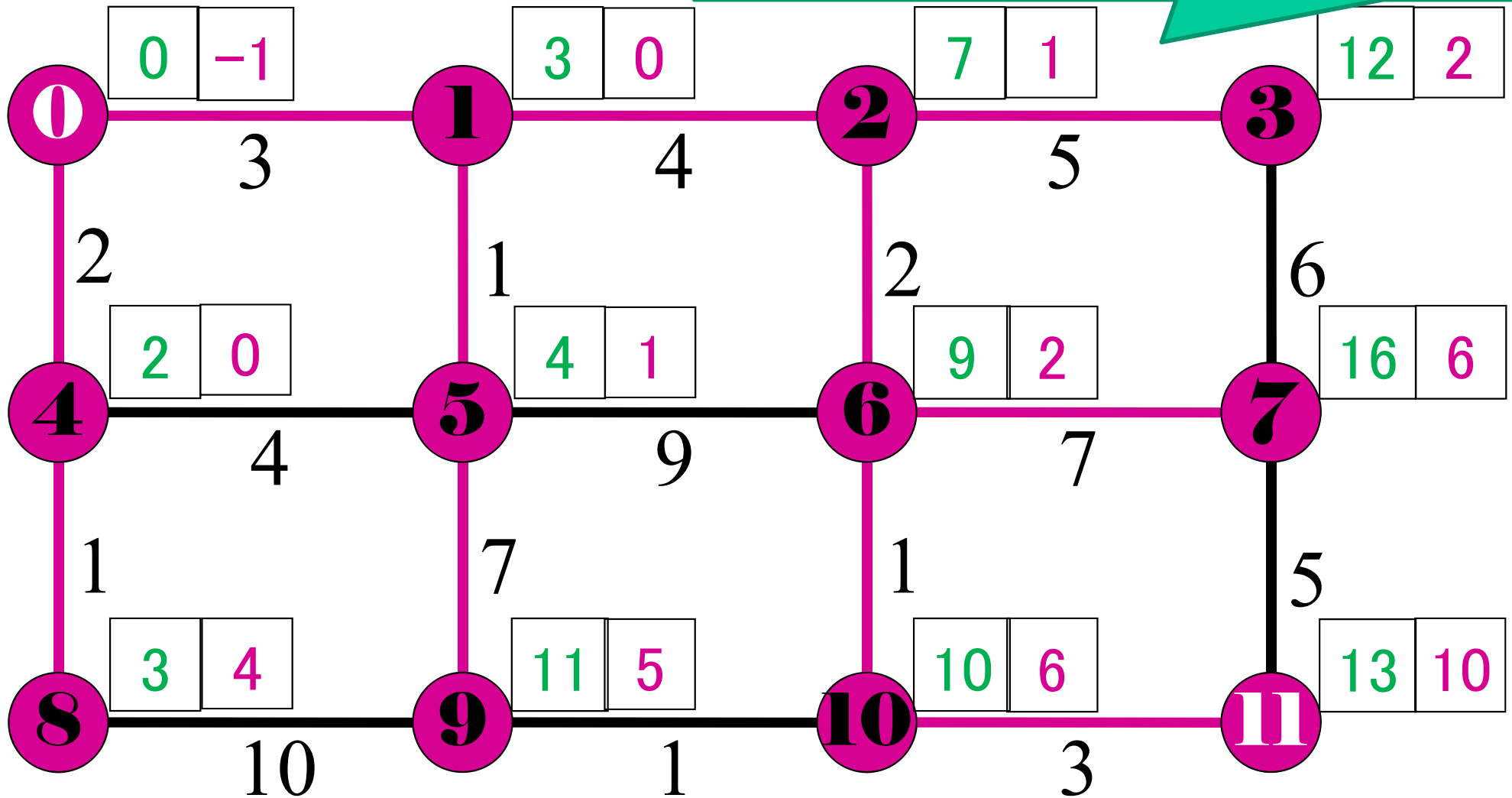


Dijkstra法 (終了判定)

step2:未確定点集合 N が空(くう) ($N=\emptyset$)になったら終了. そうでなければ step1-1 へ戻る

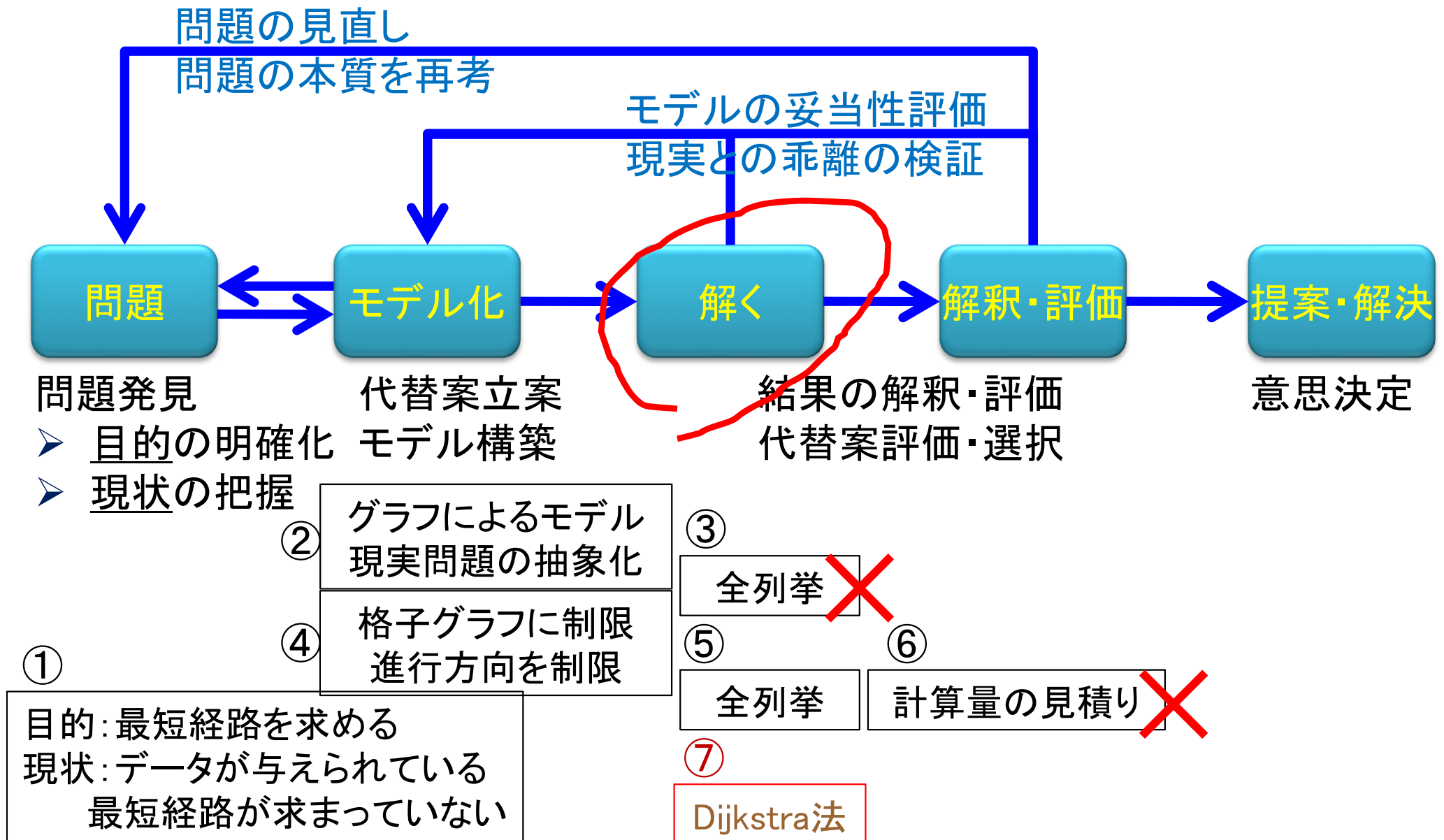
未確定点集合 $N=\emptyset$ より終了

始点から全点への最短路が求まっている
各点の距離ラベルは始点からその点までの最小コストを表す
各点から親ラベルをたどる路が最短路となる



問題解決とは？

➤ 問題発見・問題解決から意思決定まで



評価: Dijkstra法って速いのか？



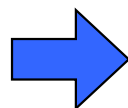
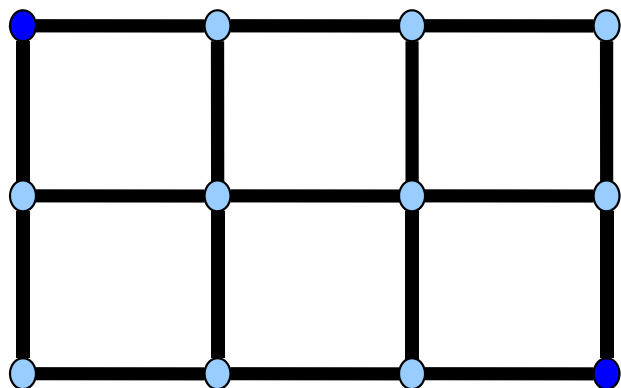
- 点の数を n とすると, 大雑把な見積もりで,

$$O(n^2) \quad \text{多項式オーダー}$$

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

- 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？

415.53 PFLOPS

51.2GFLOPS

R(横) D(縦)	全経路	富岳&しらみつぶし	Core i7 & Dijkstra
3 2	10	0.0000000000秒	0.0000000014秒
6 4	210	0.0000000000秒	0.0000000120秒
10 5	3,003	0.0000000000秒	0.0000000425秒
20 10	30,045,015	0.0000000002秒	0.0000005211秒
25 25	1.3×10^{14}	0.015210768秒	0.000044627秒
30 30	1.2×10^{17}	17.076684942秒	0.000090188秒
40 40	1.1×10^{23}	239.56 日	0.000275953秒
50 50	1.0×10^{29}	769,918.79 年	0.000660664秒
100 100	9.1×10^{58}	1.00E+26 宙齡	0.010162149秒
500 500	2.7×10^{299}	1.49E+267 宙齡	6.152490430秒

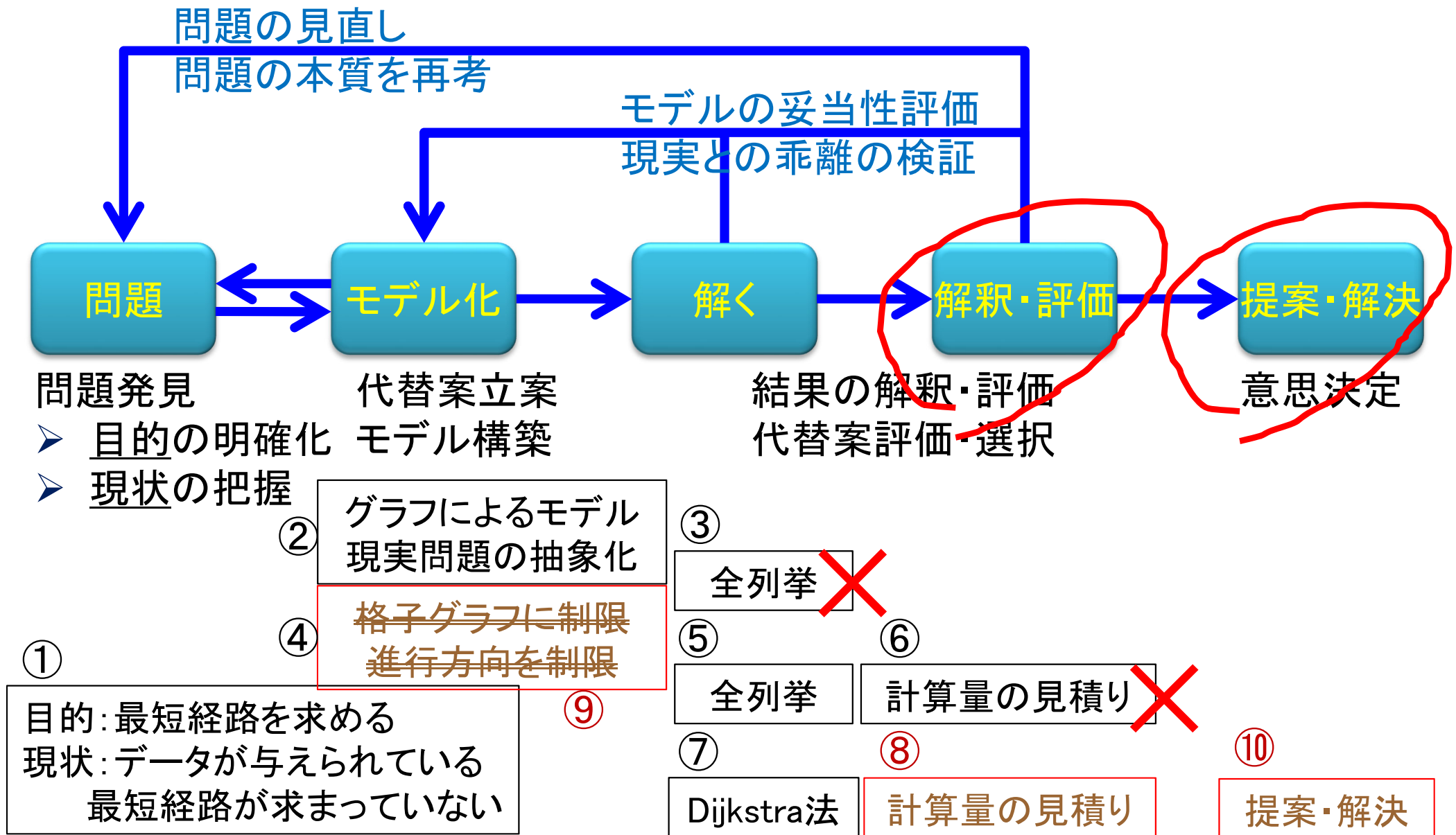
世界最速SuperComp
+ 力技 (しょぼい方法)



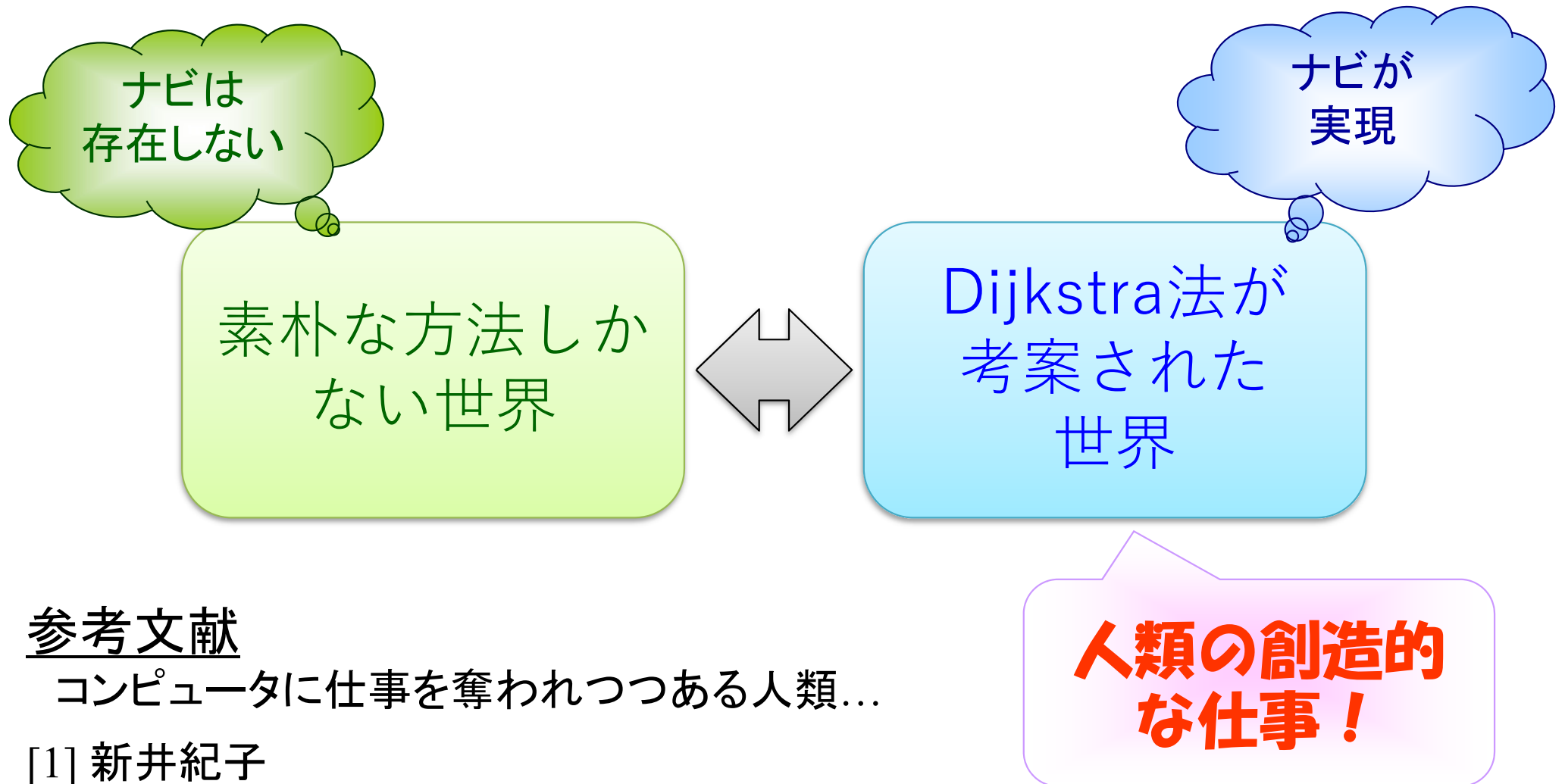
そこらのPC
+ 人間の知恵

問題解決とは？

➤ 問題発見・問題解決から意思決定まで



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)

Google Colaboratory で dijkstra法実践

- 最短路問題を作成し, pythonで解く
 - Google Colaboratory を利用し, python, networkx, etc. を使う
- 利用方法(初回)
 - (1) google アカウントにログインし, google drive へ移動
 - (2) 「新規」－「その他」－「アプリを追加」を選択
 - (3) 「Google Colaboratory」を追加
- 利用方法(2回目以降)
 - (1) google アカウントにログインし, google drive へ移動
 - (2) 「新規」－「その他」－「Google Colaboratory」を選択
- ファイルは google drive に自動保存される. 一度作成したら, 次回以降は, google drive 内のファイル [***.ipynb] を選択して, 開くことができる
- Jupyter Notebook と同様に使える
- networkx, matplotlib などの pythonライブラリは default で使用可能

Google Colaboratory で dijkstra法実践

➤ 例1) ランダム格子グラフの作成と描画

```
%matplotlib inline
import matplotlib.pyplot as plt
import networkx as nx
import random

G = nx.grid_2d_graph(5,6)
pos = {(i,j):(i,j) for (i,j) in G.nodes() }
for (i,j) in G.edges():
    G[i][j]["weight"] = random.randint(1,10)

edge_labels = { (i,j):str(G[i][j]["weight"]) for (i,j) in G.edges() }
plt.figure()
nx.draw(G, pos=pos,with_labels=False, node_size=100)
nx.draw_networkx_edge_labels(G,pos,edge_labels=edge_labels)
plt.show()
```

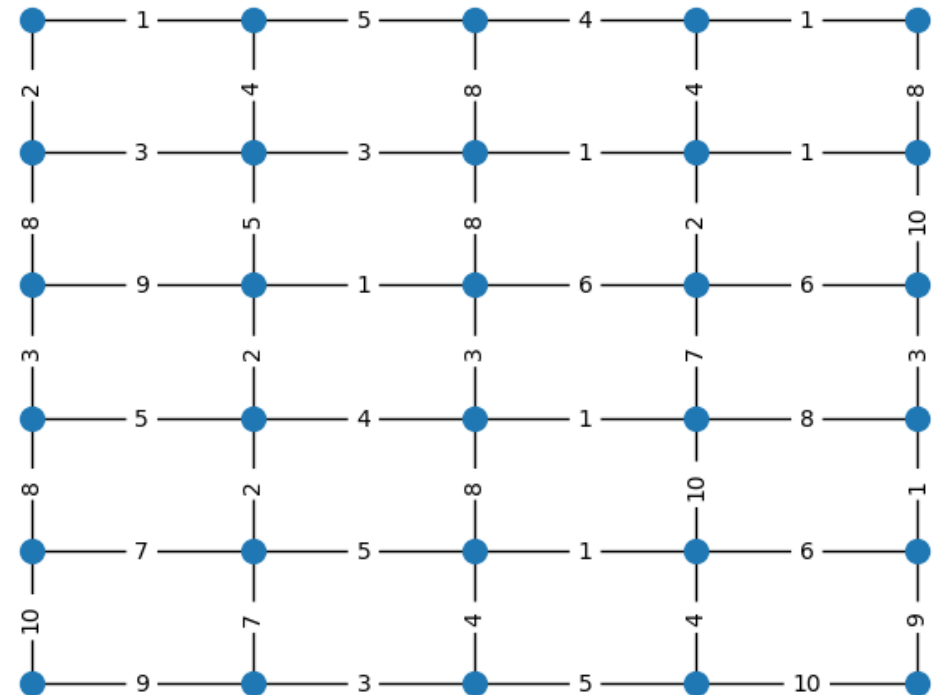
Google colaboratory の
[コード] 内に記述

全て記述出来たら、左
側の三角ボタンを押す
と、コードが実行される

コードの下に
実行結果が表示される

実行結果

コードの下に実行結果が表示されずに、
エラーメッセージが出たら、内容をよく読
み、コードの間違っている箇所を探して
修正する。修正し終わったら再度実行ボタ
ンを押す(正しく出来るまでこの繰り返し)



Google Colaboratory で dijkstra法実践

➤ 例1) dijkstra法の実行と結果の描画

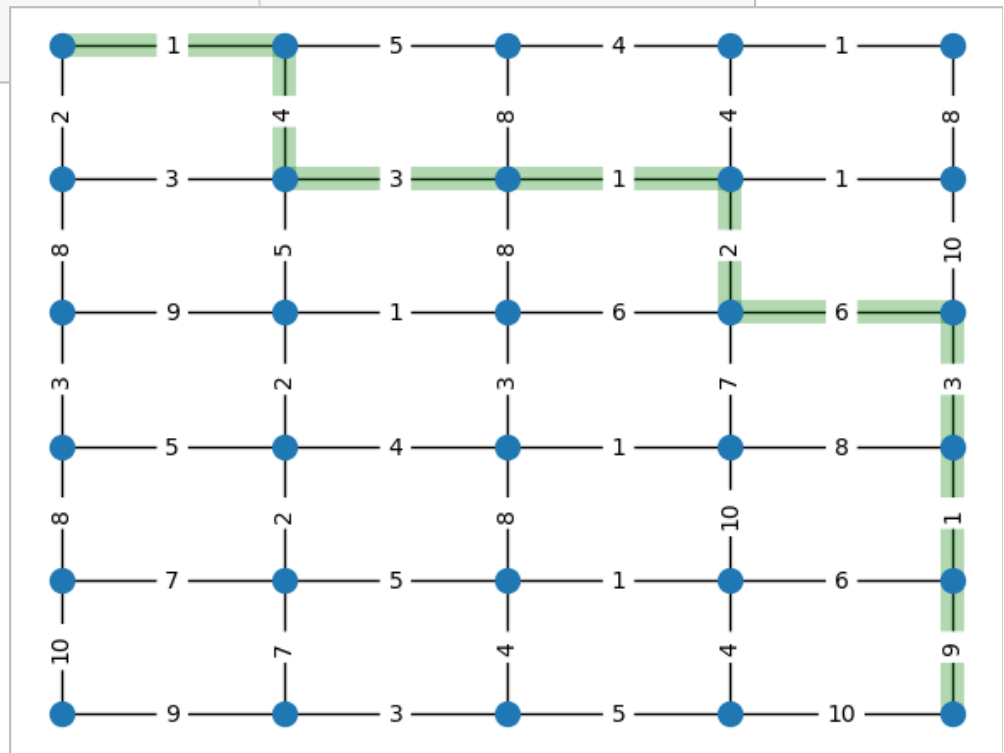
```
▶ path = nx.dijkstra_path(G, (0,5), (4,0), weight="weight")

edge_list = []
i = path[0]
for count in range(1, len(path)):
    j = path[count]
    edge_list.append( (i,j) )
    i = j
plt.figure()
nx.draw(G, pos=pos, with_labels=False, node_size=100)
nx.draw(G, pos=pos, with_labels=False, node_size=100, edgelist=edge_list, edge_color="green", width=10, alpha=0.3)
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels)
plt.show()
```

上部にある[+コード]ボタンを押す. すると, 前ページの結果の下に新たな[コード]画面が開くので, そこにこれを記述.
[コード]画面は右側の上下矢印[↑ / ↓]で移動(入替)可能

現在点座標(0,5): 左上
目的点座標(4,0): 右下
として
最短路を
dijkstra法で探索

実行結果



Google Colaboratory で dijkstra法実践

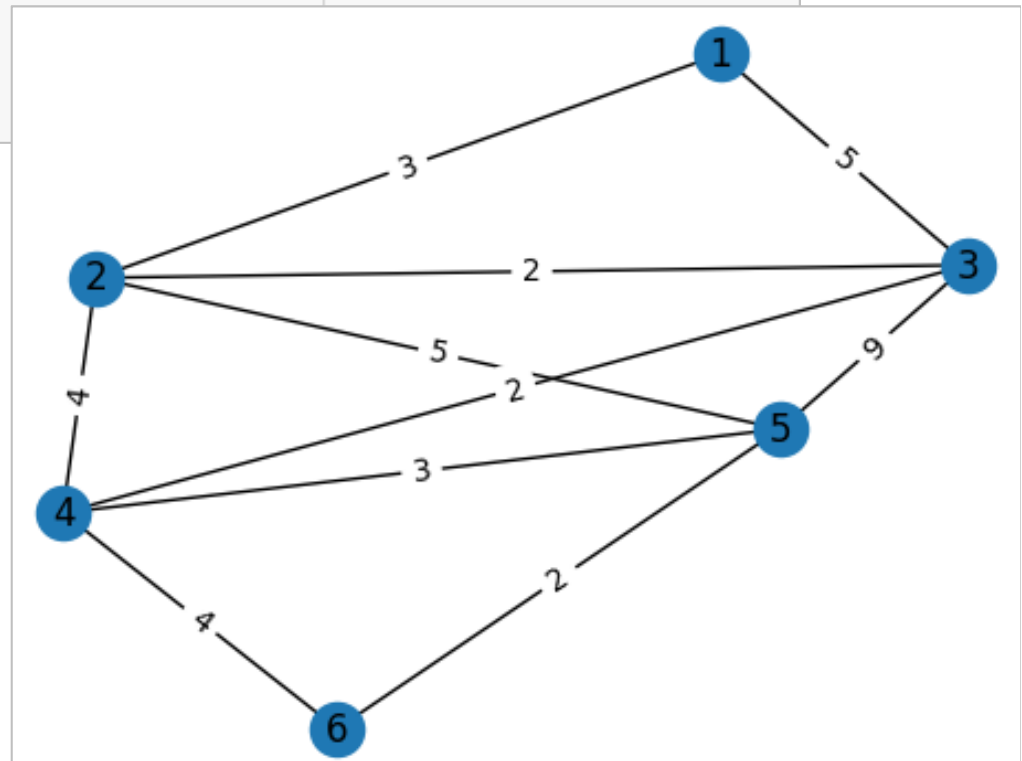
➤ 例2) (格子ではない) 一般のグラフの作成と描画

```
%matplotlib inline
import matplotlib.pyplot as plt
import networkx as nx

G = nx.Graph()
G.add_weighted_edges_from([(1,2,3),(1,3,5),(2,3,2),(2,4,4),(2,5,5),(3,4,2),(3,5,9),(4,5,3),(4,6,4),(5,6,2)])

pos = nx.spring_layout(G)
nx.draw_networkx_nodes(G, pos)
nx.draw_networkx_edges(G, pos)
nx.draw_networkx_labels(G, pos)
edge_labels = nx.get_edge_attributes(G, "weight")
nx.draw_networkx_edge_labels(G, pos, edge_labels)
plt.show()
```

実行結果



Google Colaboratory で dijkstra法実践

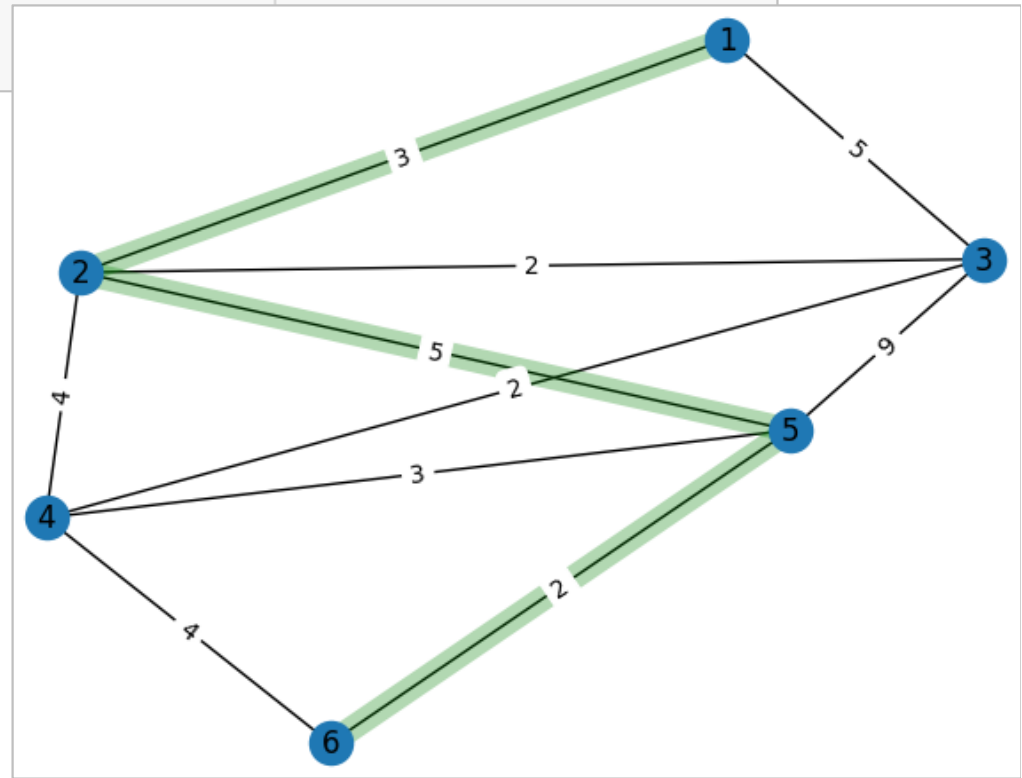
➤ 例2) dijkstra法の実行と結果の描画

```
▶ path = nx.dijkstra_path(G, 1, 6, weight="weight")

edge_list = []
i = path[0]
for count in range(1, len(path)):
    j = path[count]
    edge_list.append( (i,j) )
    i = j
plt.figure()
nx.draw(G, pos=pos, with_labels=True, node_size=300)
nx.draw(G, pos=pos, with_labels=False, node_size=100, edgelist=edge_list, edge_color="green", width=10, alpha=0.3)
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels)
plt.show()
```

現在点①
目的点⑥
として
最短路を
dijkstra法で探索

実行結果



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)
- 久保「**Pythonによる実務で役立つ最適化問題100+**」 朝倉書店(2022)

- 関連する授業

- 「**ネットワークモデル分析A / B**」(3, 4セメ)
- 「**最適化モデル分析**」(5セメ)
- 「**プログラミング**」(3, 4セメ) etc...