アルゴリズムとデータ構造

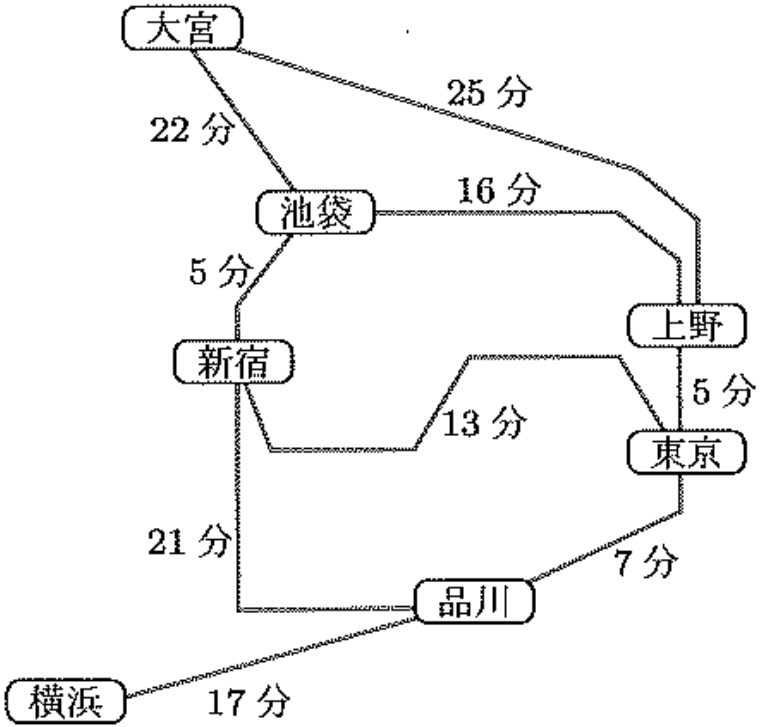
第19週目

担当　情報システム部門　徳光政弘
2025年10月28日

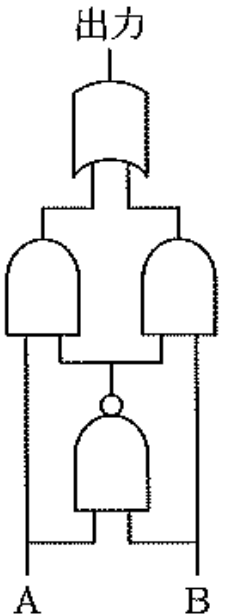
今日の内容

- グラフのデータ表現
- ダイクストラ法
  - ある頂点からの最短経路を求める

グラフ



（a）鉄道の路線図



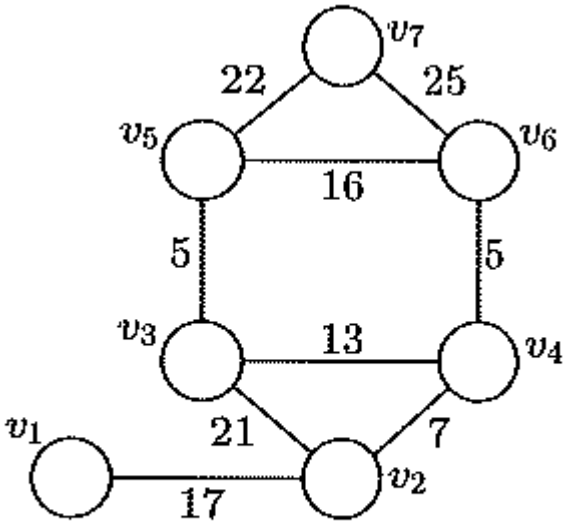
（b）回路図

図 10.1　一般生活におけるグラフ

最短経路問題

例えば、v_1からv_7までの最短となるコスト(重み)を求めたい。

最小のコストと経路
経路
コスト　54　$(v_1, v_2, v_4, v_6, v_7)$



（a）路線図を表すグラフ

最短経路問題

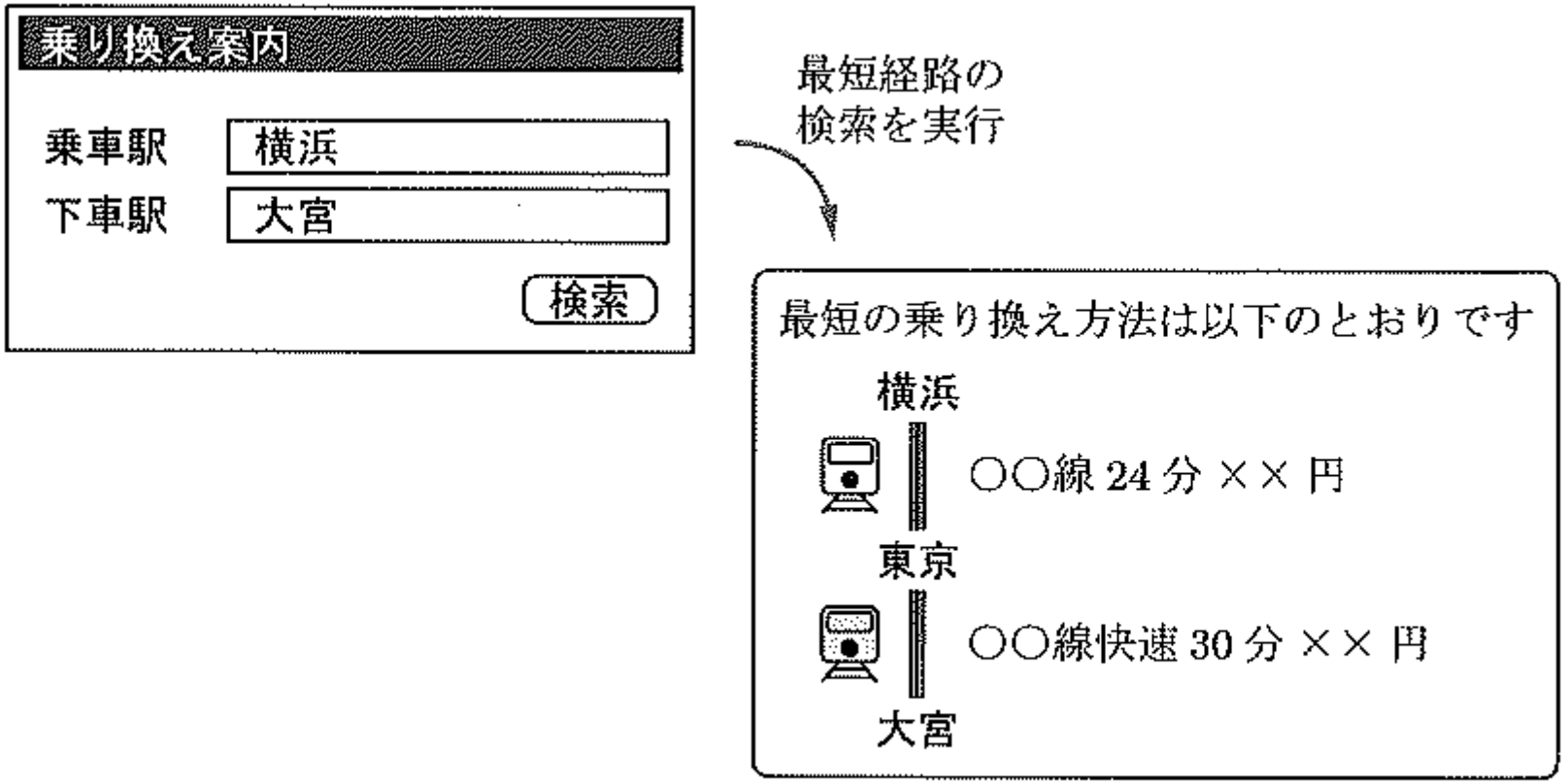


図 10.7　最短経路問題の実用例

最短経路問題（到着時刻順）

最短経路問題（乗換回数順）

ダイクストラ法

① 集合 S に含まれない頂点のうち，始点からの距離を表す変数 d_i の値がもっとも小さい頂点 v_k を選ぶ．

② 頂点 v_k を S に加える．

③ 頂点 v_k に隣接する頂点のうち S に含まれないすべての頂点について，始点から距離を格納する変数 d_i を再計算する．

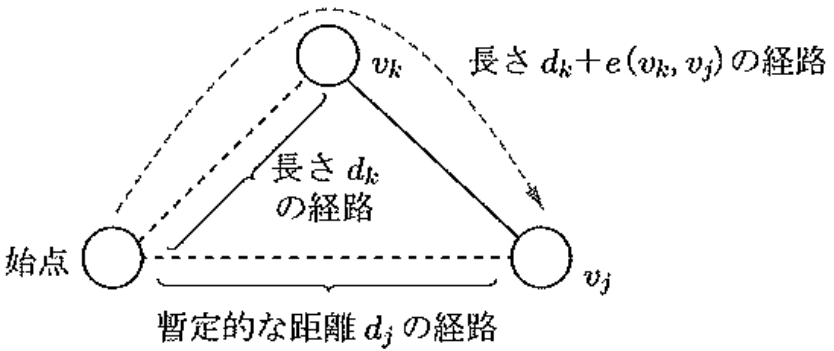


図 10.8　変数 d_j の再計算

ダイクストラ法

初期値： $S = \phi$,

d_1	d_2	d_3	d_4	d_5	d_6	d_7
0	∞	∞	∞	∞	∞	∞

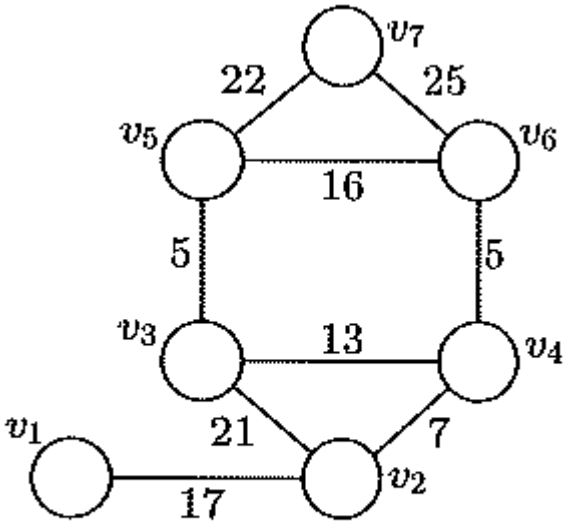
1 回目： $S = \{v_1\}$,

d_1	d_2	d_3	d_4	d_5	d_6	d_7
0	17	∞	∞	∞	∞	∞

2 回目： $S = \{v_1, v_2\}$,

d_1	d_2	d_3	d_4	d_5	d_6	d_7
0	17	38	24	∞	∞	∞

$d_3 = \min\{\infty, 17 + 21\} = 38,\ d_4 = \min\{\infty,\ 17 + 7\} = 24$



（a）路線図を表すグラフ

ダイクストラ法

3 回目： $S = \{v_1, v_2, v_4\}$,

d_1	d_2	d_3	d_4	d_5	d_6	d_7
0	17	37	24	∞	29	∞

4 回目： $S = \{v_1, v_2, v_4, v_6\}$,

d_1	d_2	d_3	d_4	d_5	d_6	d_7
0	17	37	24	45	29	54

5 回目： $S = \{v_1, v_2, v_3, v_4, v_6\}$,

d_1	d_2	d_3	d_4	d_5	d_6	d_7
0	17	37	24	42	29	54

6 回目： $S = \{v_1, v_2, v_3, v_4, v_5, v_6\}$,

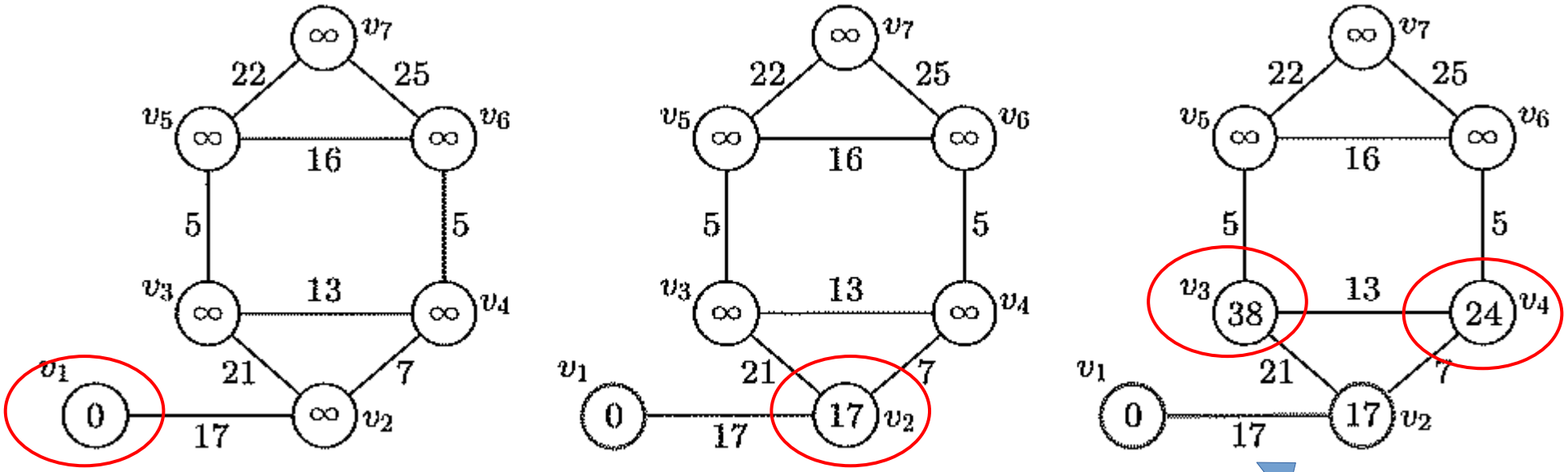
d_1	d_2	d_3	d_4	d_5	d_6	d_7
0	17	37	24	42	29	54

7 回目： $S = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$,

d_1	d_2	d_3	d_4	d_5	d_6	d_7
0	17	37	24	42	29	54

ダイクストラ法

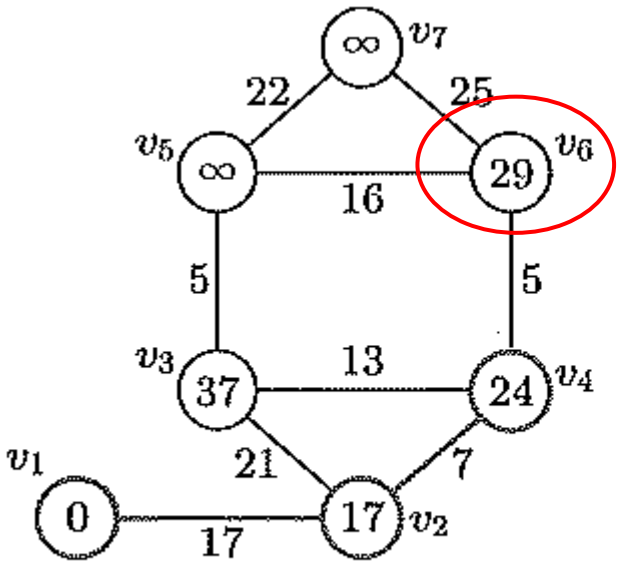
図10.9は色付きでわかりやすい



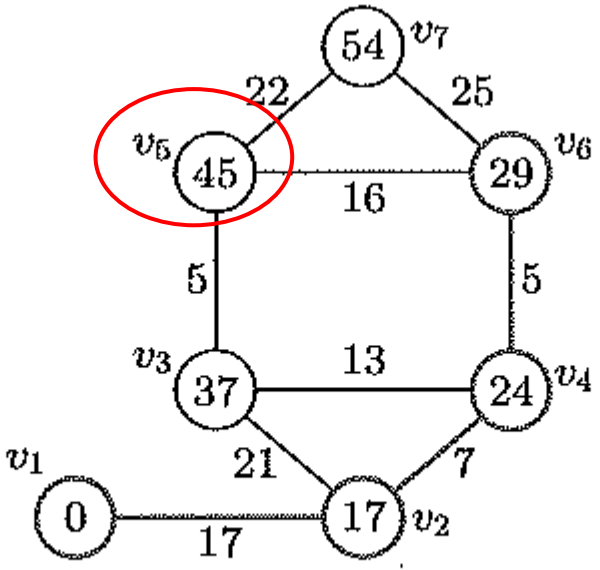
$$d_3 = \min\{\infty, 17 + 21\} = 38,\ d_4 = \min\{\infty,\ 17 + 7\} = 24$$

最小コストを比較している演算になっている。

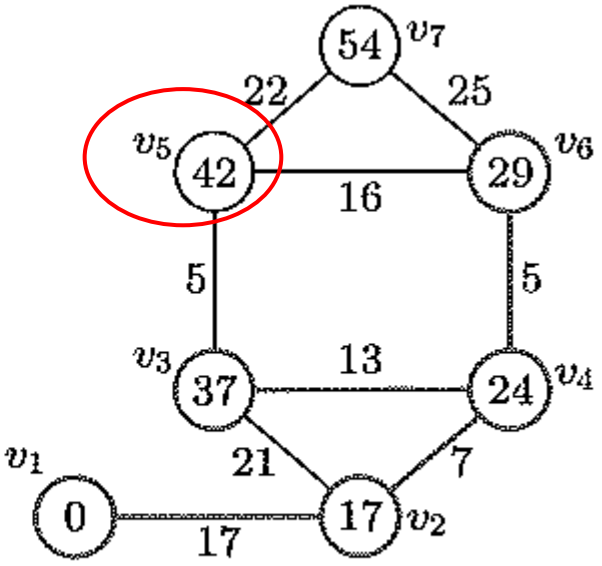
ダイクストラ法


∞
v_7
22
25
v_5
∞
29
v_6
16
5
5
v_3
37
13
24
v_4
21
7
v_1
0
17
17
v_2

(e)


54
v_7
22
25
v_5
45
29
v_6
16
5
5
v_3
37
13
24
v_4
21
7
v_1
0
17
17
v_2

(f)


54
v_7
22
25
v_5
42
29
v_6
16
5
5
v_3
37
13
24
v_4
21
7
v_1
0
17
17
v_2

(g)

ダイクストラ法

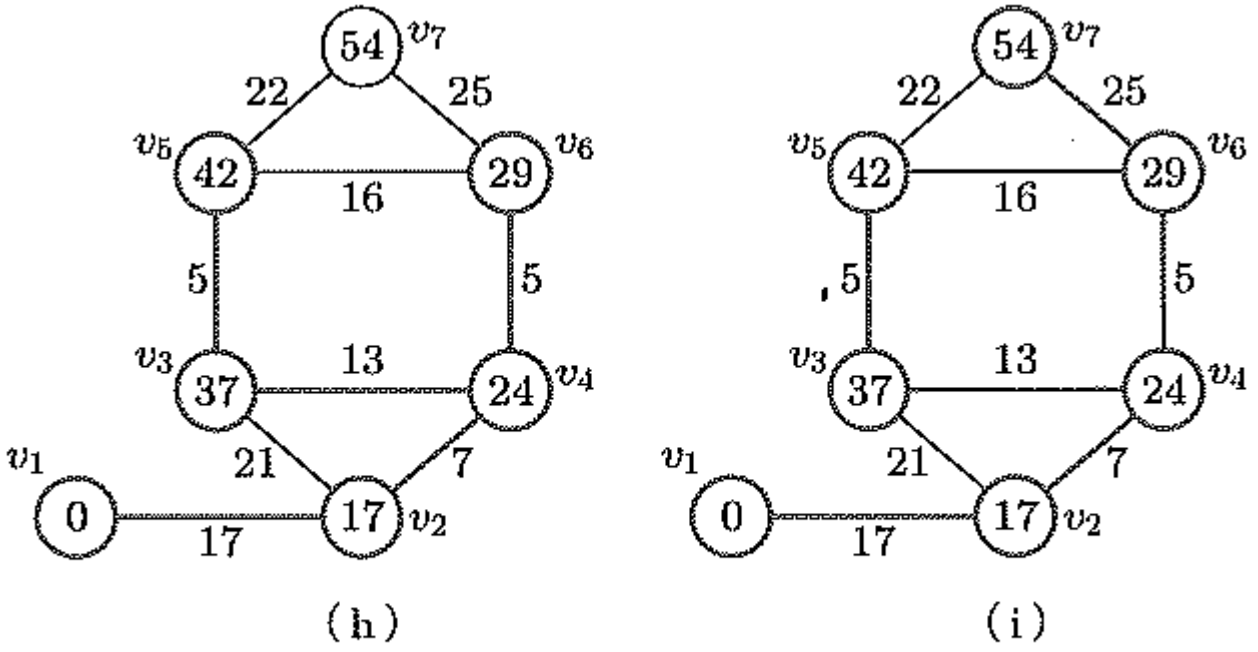


図 10.9　ダイクストラ法の実行例

ダイクストラ法

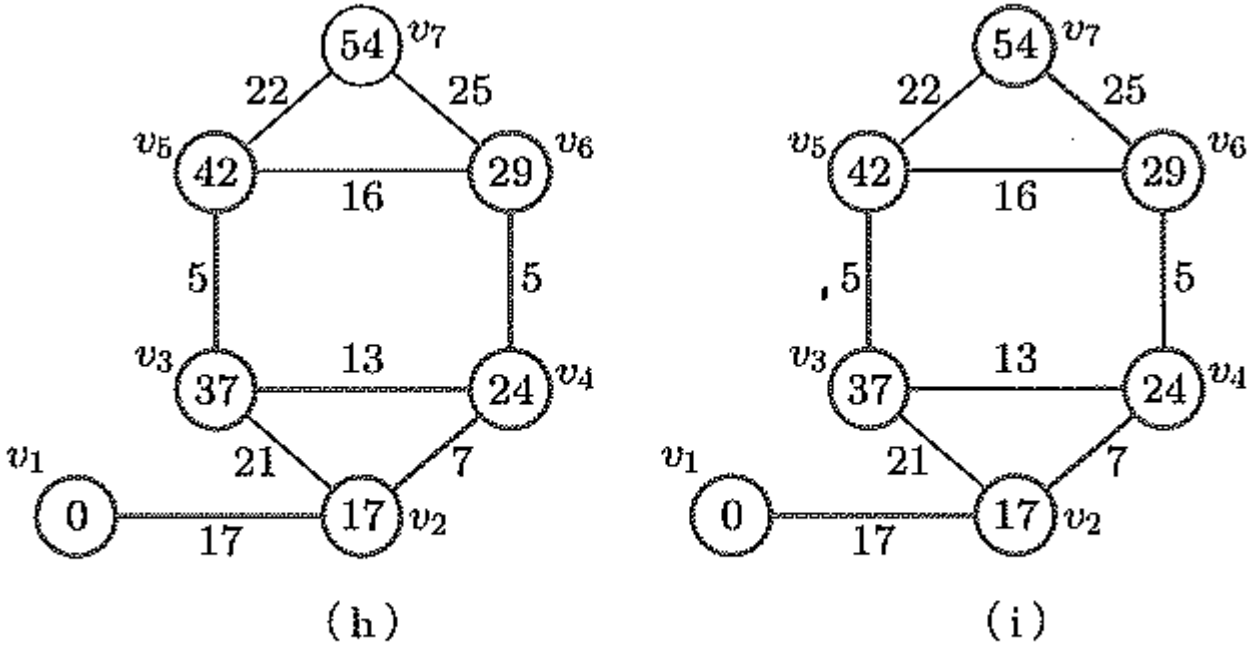


図 10.9　ダイクストラ法の実行例

頂点集合Sに対する計算のみを実行している

ダイクストラ法

アルゴリズム 10.3　ダイクストラ法

入力：頂点の集合 V と辺の集合 E，および V に含まれる始点 v_s．
（各頂点 v_i は v[i] と表し，辺 (v_i, v_j) の重みは e[i][j] で表す．）

```
for (i=1; i<=n; i=i+1) { D[i]=\infa; }
S=φ; s=1; D[s]=0;
for (i=1; i<=n; i=i+1) {
  S に含まれない頂点の中から，配列 D の値が最小の頂点 v[k] を求める;
  頂点v[k]をS に追加する;
  for (j=1; j<=n; j=j+1) {
    if ((v[j]がv[k]に隣接する) かつ (v[j]がS に含まれない))
      { D[j]=min(D[j], D[k]+e[k][j]); }
  }
}
```

実装では、訪問のチェックに配列、隣接行列を使うのがやりやすい。

ダイクストラ法

- for文が2個ある　少なくともO(n^2)
- 頂点に対するコストの計算（Sと配列）　O(n)
- 合計すると、多項式の次数が優先されてO(n^2)