

アルゴリズムとデータ構造

第18週目

担当 情報システム部門 徳光政弘
2025年10月21日

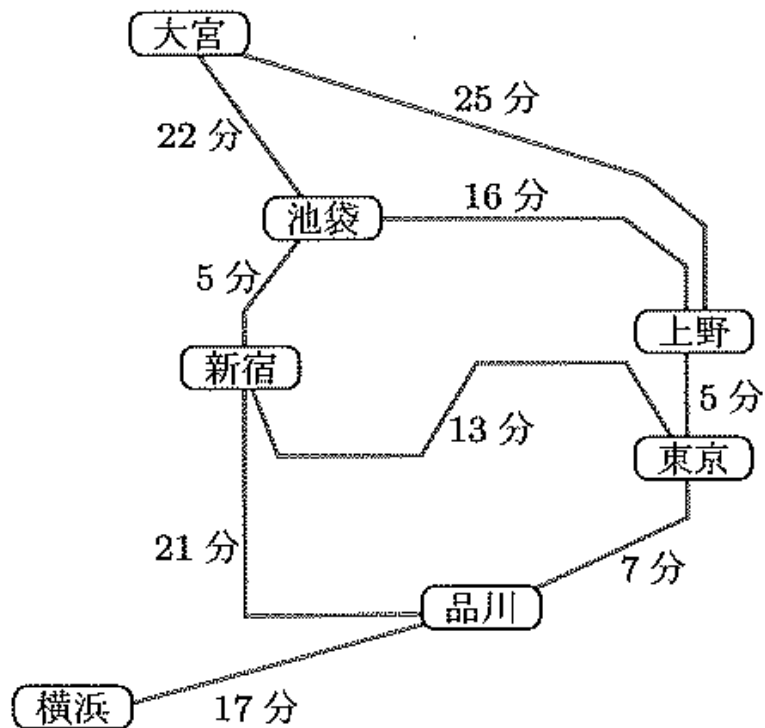
今日の内容

- グラフのデータ表現
- グラフの幅優先探索・深さ優先探索
 - 人工知能でも解探索としてよく使われる
 - おそらく5年生の人工知能でも再び登場

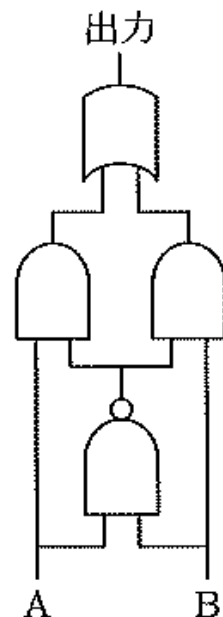
今後の主な内容

1. グラフ 2回
2. バックトラック法 1回
3. 動的計画法 1回
4. 文字列 2回
5. 多項式と行列 1回
6. 画像・動画、情報理論 3回
7. 計算複雑性 2回

グラフ



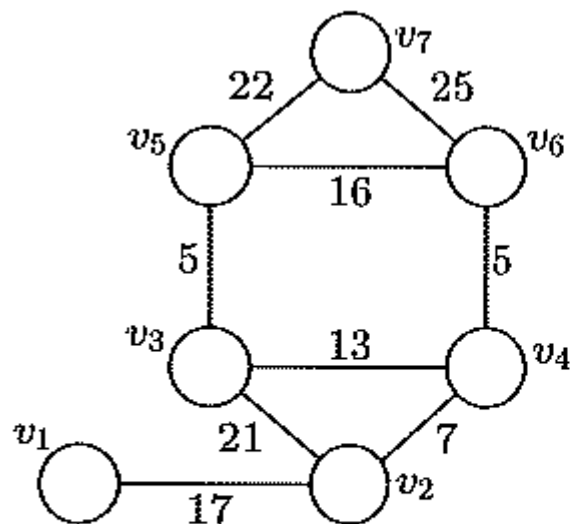
(a) 鉄道の路線図



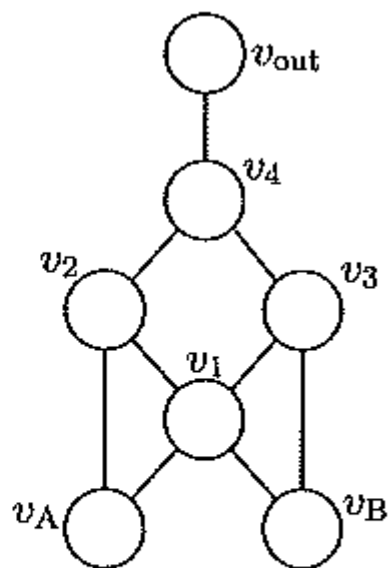
(b) 回路図

図 10.1 一般生活におけるグラフ

グラフ



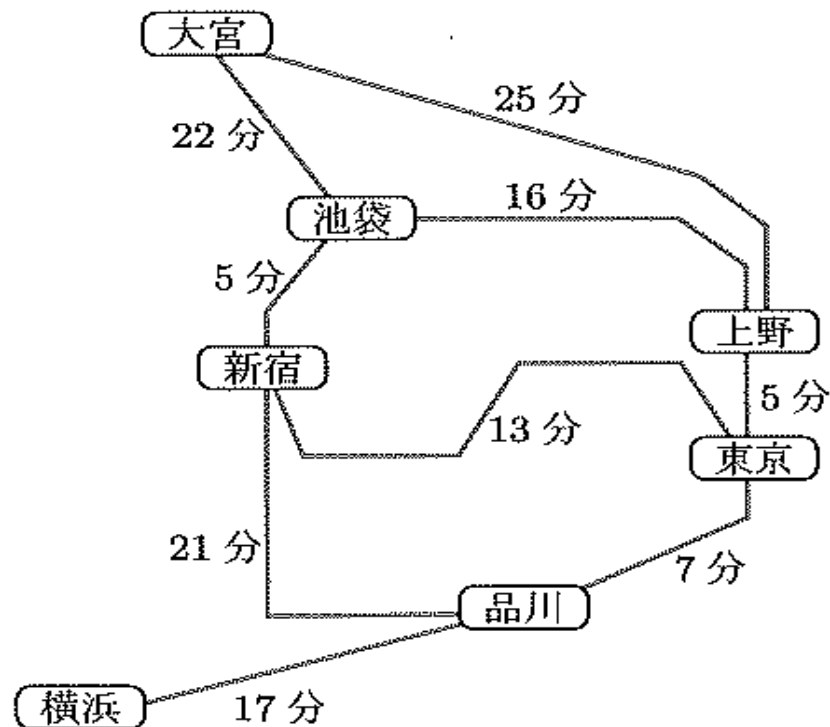
(a) 路線図を表すグラフ



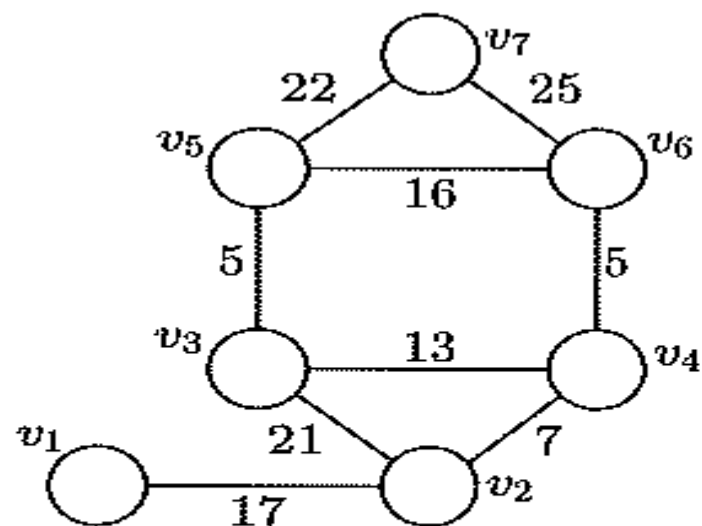
(b) 回路図を表すグラフ

図 10.2 グラフ

グラフ



(a) 鉄道の路線図



(a) 路線図を表すグラフ

グラフを表すデータ構造

$$G = (V, E)$$

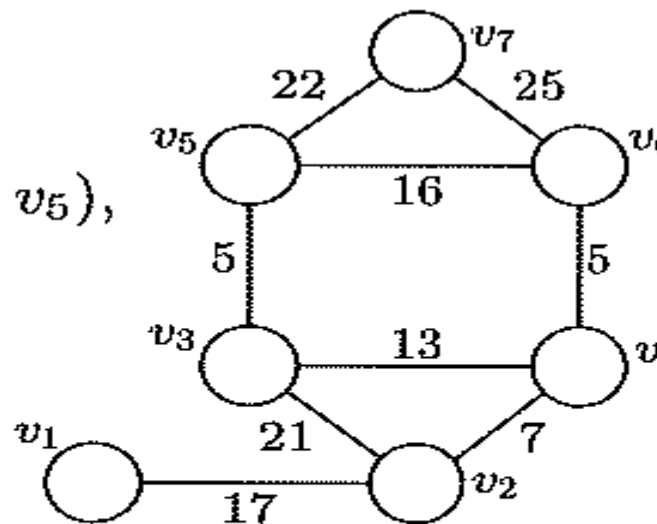
$$V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$$

$$E = \{(v_1, v_2), (v_2, v_3), (v_2, v_4), (v_3, v_4), (v_3, v_5), \\ (v_4, v_6), (v_5, v_6), (v_5, v_7), (v_6, v_7)\}$$

懐かしい形式的な定義

主な表現方法

- 隣接行列 行列で表現
- 隣接リスト リストで表現



(a) 路線図を表すグラフ

隣接行列

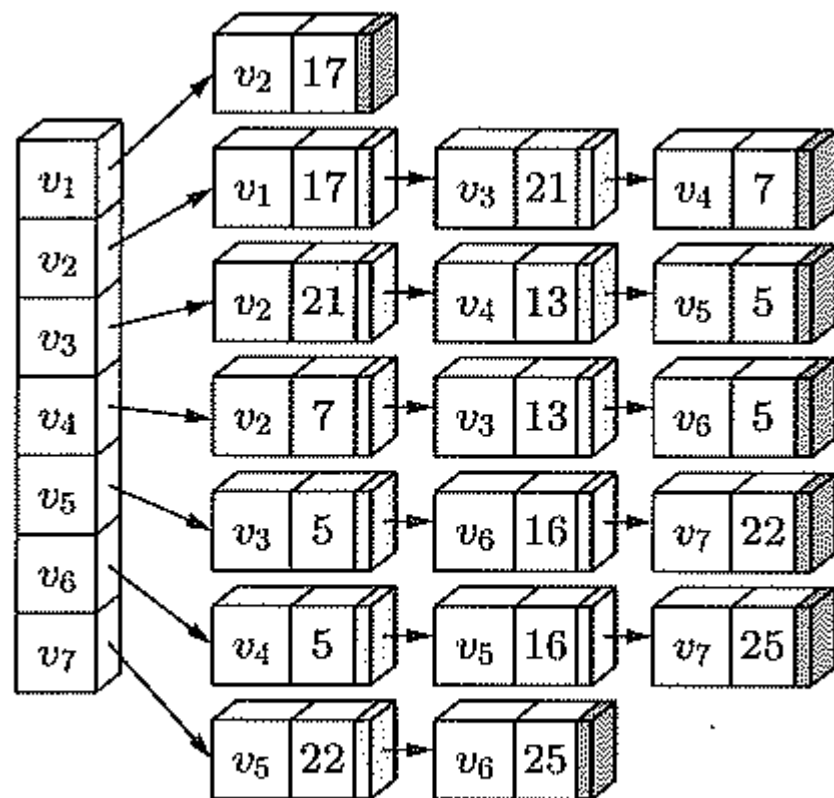
- 接続のある・なしを行列で表現する
- コストを考慮している場合は、コストを使う

—	1	2	3	4	5	6	7
1(v_1)	0	17	0	0	0	0	0
2(v_2)	17	0	21	7	0	0	0
3(v_3)	0	21	0	13	5	0	0
4(v_4)	0	7	13	0	0	5	0
5(v_5)	0	0	5	0	0	16	22
6(v_6)	0	0	0	5	16	0	25
7(v_7)	0	0	0	0	22	25	0

(a) 図 10.2(a) のグラフを表す隣接行列

隣接リスト

- 接続が「ある」をリストで表現する
- コストを考慮している場合は、コストを使う



(a) 図 10.2(a) のグラフを表す隣接リスト

グラフの探索

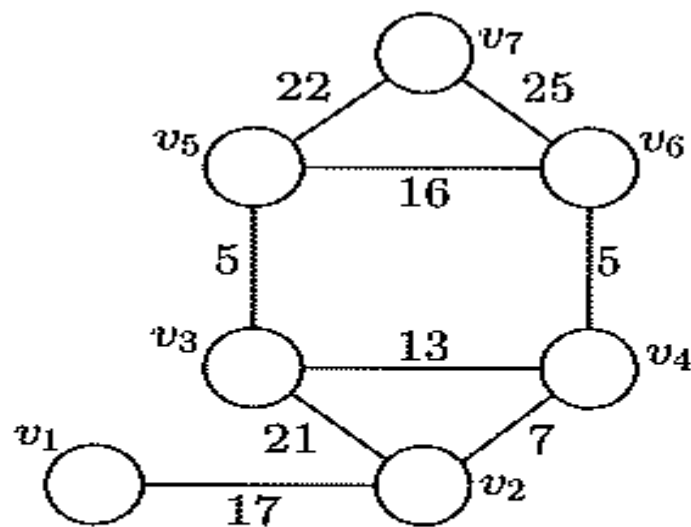
何らかの方法でグラフ全体のノードを訪問する

- 深さ優先探索
- 幅優先探索

探索の例

$v_7 \rightarrow v_5 \rightarrow v_6 \rightarrow v_3 \rightarrow v_4 \rightarrow v_2 \rightarrow v_1$

$v_7 \rightarrow v_5 \rightarrow v_3 \rightarrow v_2 \rightarrow v_1 \rightarrow v_4 \rightarrow v_6$



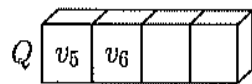
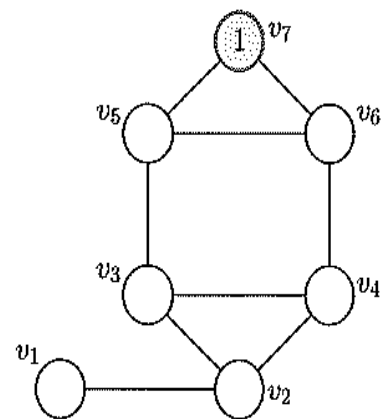
(a) 路線図を表すグラフ

図 1

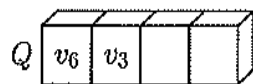
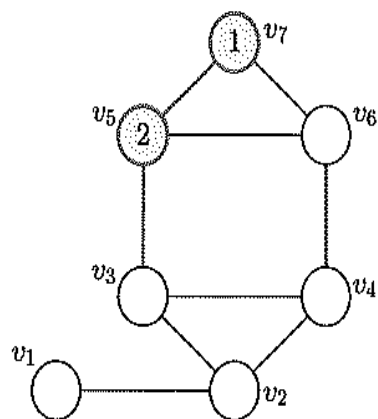
幅優先探索

- ① キュー Q を空にした後に、始点 v_s をキュー Q に加える.
- ② キュー Q が空でない間、以下の操作②-1, ②-2, ②-3 を実行する.
 - ②-1 キュー Q から頂点を取り出す(取り出した頂点を v_k とする).
 - ②-2 v_k を調査済みとする.
 - ②-3 v_k に隣接する頂点のうち、キュー Q に追加されていない、かつ、調査済みでない頂点をすべてキュー Q に追加する.

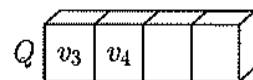
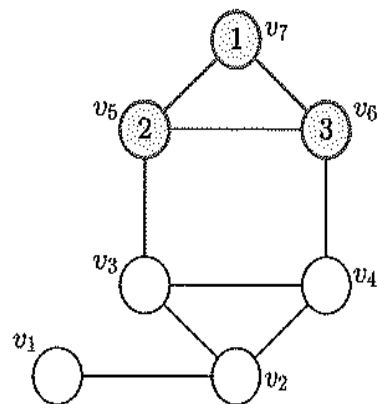
幅優先探索



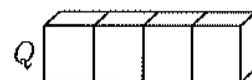
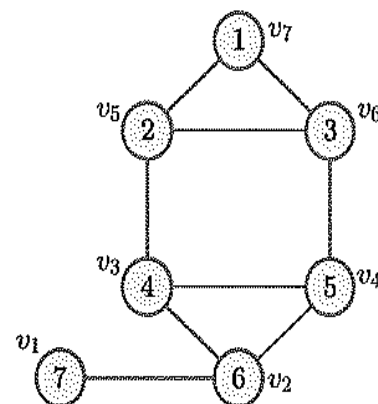
(a)



(b)



(c)



(d)

図 10.5 幅優先探索の実行例

幅優先探索

アルゴリズム 10.1 幅優先探索

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 $v[i]$ と表す)

```
for (i=1; i<=n; i=i+1) { InQ[i]=0; C[i]=0; }
```

キューQを空にする;

```
enqueue(Q, v[s]);
```

```
while (キューQが空でない) {
```

```
    v[k]=dequeue(Q);    //キューQから取り出した頂点をv[k]とする
```

```
    C[k]=1;              //頂点v[k]を調査済みにする
```

v[k]のすべての隣接頂点v[h]について以下を実行する;

```
    if ((InQ[h]==0) かつ (C[h]==0)) { enqueue(Q, v[h]); InQ[h]=1; }
```

```
}
```

```
}
```

幅優先探索の計算量

enqueue、dequeue: $O(1)$

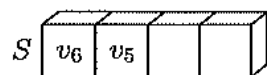
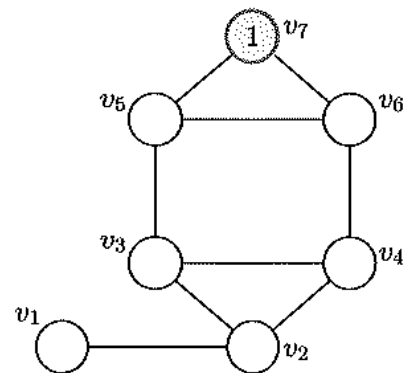
while文: 場合分けが必要

- 隣接行列: n 個の頂点に対して、最大で n 個の頂点を調べる
- 隣接リスト: 最大で m 個の辺

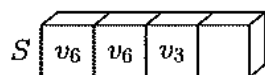
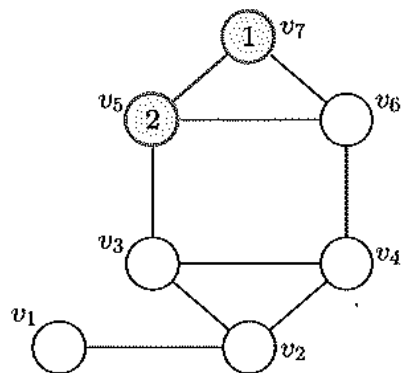
全体

- 隣接行列 $O(n + n^2) = n^2$
- 隣接リスト $O(n + m)$

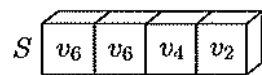
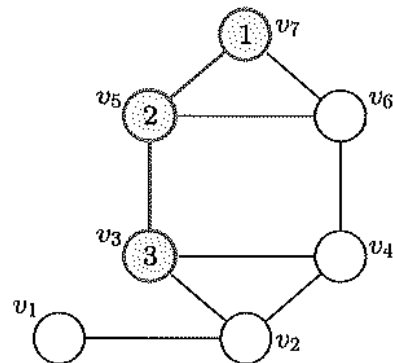
深さ優先探索



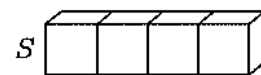
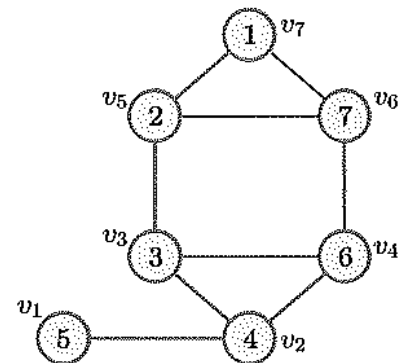
(a)



(b)



(c)



(d)

深さ優先探索

- ① スタック S を空にした後に, 始点 v_s をスタック S に加える.
- ② スタック S が空でない間, 以下の操作②-1, ②-2, ②-3 を実行する.
 - ②-1 スタック S から頂点を取り出す(取り出した頂点を v_k とする).
 - ②-2 v_k が調査済みでなければ, 以下の②-2-1 と②-2-2 を実行する.
 - ②-2-1 v_k を調査済みとする.
 - ②-2-2 v_k に隣接する頂点のうち, 調査済みでない頂点をすべてスタック S に追加する.

深さ優先探索

アルゴリズム 10.2 深さ優先探索

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 $v[i]$ と表す)

```
for (i=1; i<=n; i=i+1) { C[i]=0; }
```

スタック S を空にする;

```
push(S, v[s]);
```

```
while (スタック  $S$  が空でない) {
```

```
    v[k]=pop(S);    //スタック  $S$  から取り出した頂点を  $v[k]$  とする
```

```
    if (C[k]==0) {
```

```
        C[k]=1;        //頂点  $v[k]$  を調査済みにする
```

```
         $v[k]$  のすべての隣接頂点  $v[h]$  について以下を実行する;
```

```
            if (C[h]==0) { push(S, v[h]); }
```

```
    }
```

```
}
```

深さ優先探索の計算量

push、pop: $O(1)$

while文: 場合分けが必要

- 隣接行列: n 個の頂点に対して、最大で n 個の頂点を調べる
- 隣接リスト: 最大で m 個の辺

全体

- 隣接行列 $O(n + n^2) = n^2$
- 隣接リスト $O(n + m)$