アルゴリズムとデータ構造

第21週目

担当　情報システム部門　徳光政弘
2025年11月18日

今日の内容

- バックトラック法
- 分枝限定法（バックトラック法を補完する方法）
- 8クイーン問題（バックトラック法を勉強するときの定番問題）

バックトラック法

- 効率的に解くアルゴリズムがない場合は、解を網羅的に調べる必要がある。うまく解を列挙して、調べていく。検証も効率的である必要がある。
- 闇雲に調べてもうまく行かないので、絞り込む範囲を絞ることも必要になる。（分枝限定法）

迷路

右手法は有名な方法ではある。うまい解き方はない。

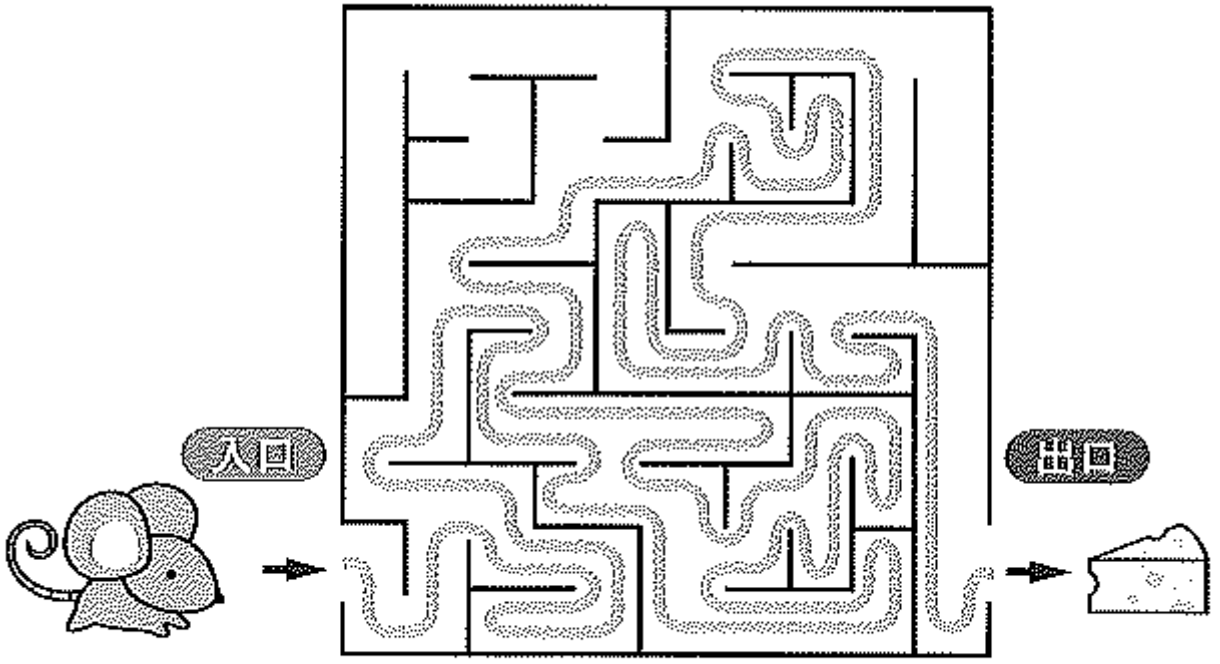


図 9.1　右手法による迷路の解法

部分和問題

【問題 9.1】 部分和問題

$\{x_1, x_2, \ldots, x_n\}$ という n 個の正の実数の集合と，s という正の実数が与えられたとする．このとき，$\{x_1, x_2, \ldots, x_n\}$ の中からその和がちょうど s になる実数の選び方を求めよ．

例えば3、14、6、9の整数を使って、12になる組み合わせをすべて調べる場合を考える。
3 + 9 = 12

例えば3、14、6、9の整数を使って、19になる組み合わせをすべて調べる場合を考える。
うまい答えはない

部分和問題

【問題 9.1】 部分和問題

$\{x_1, x_2, \ldots, x_n\}$ という n 個の正の実数の集合と，s という正の実数が与えられたとする．このとき，$\{x_1, x_2, \ldots, x_n\}$ の中からその和がちょうど s になる実数の選び方を求めよ．

例えば3、14、6、9の整数を使って、12になる組み合わせをすべて調べる場合を考える。
3 + 9 = 12

組み合わせで考えると、その数を使う・使わないで2^4=16通りある。
組み合わせに使う数が大きくなると、途端に難しくなる。

列挙木

0を使わない、1を使うとする。パターンに応じて、木を作ることができる。

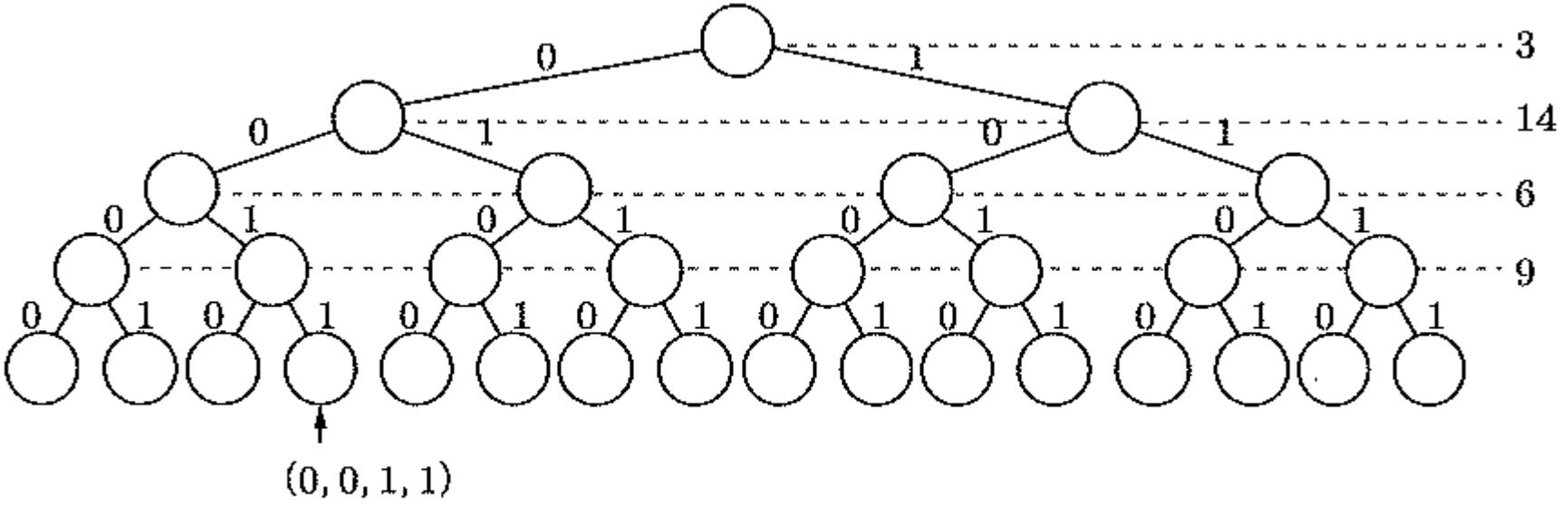
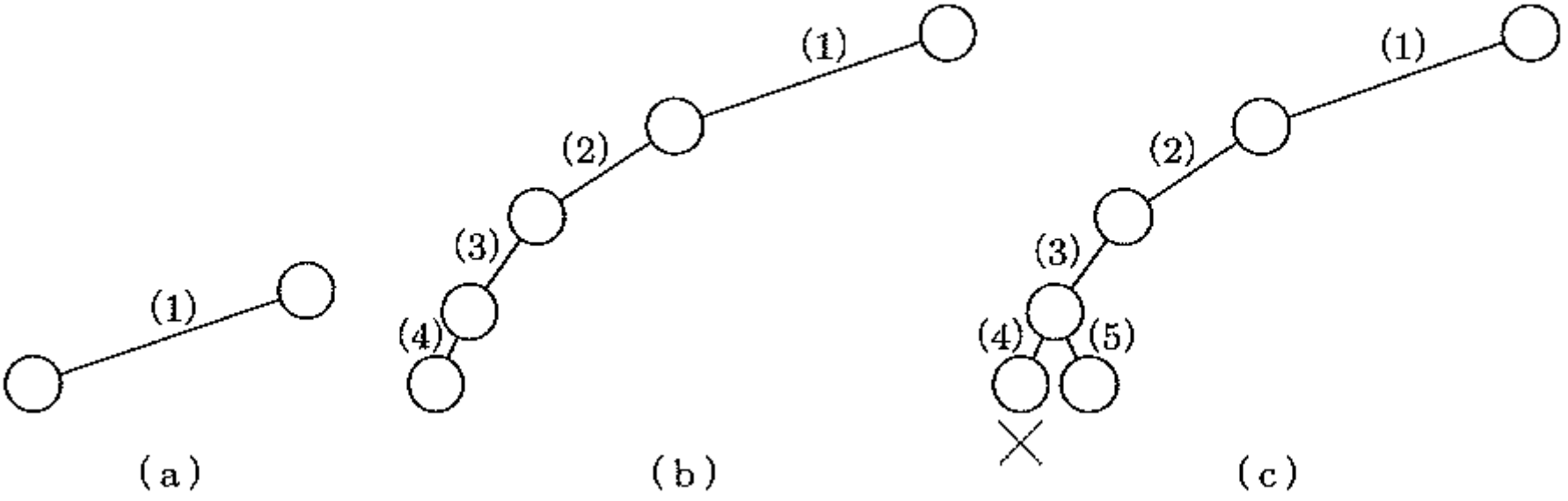


図 9.2　列挙木

バックトラック法の実行過程



バックトラック法の実行過程

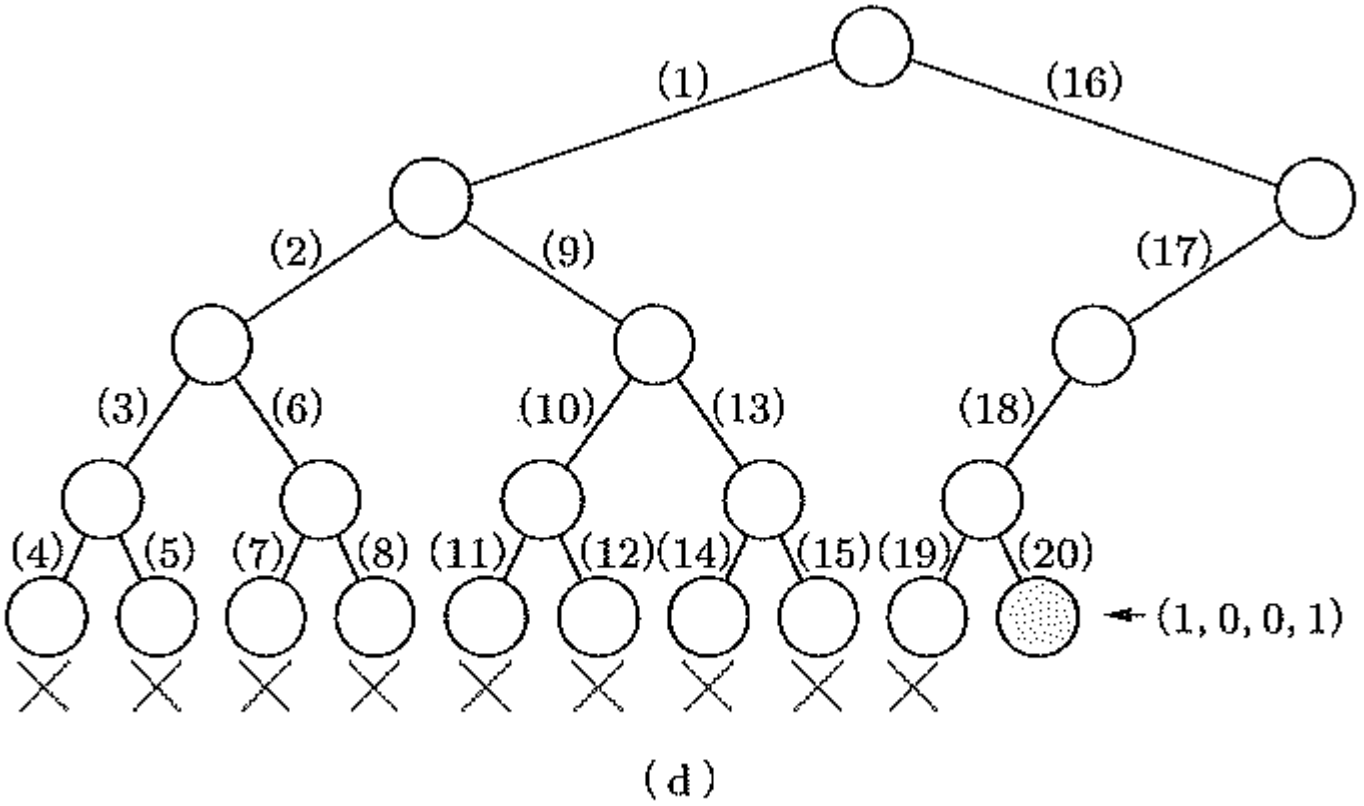


図 9.3　バックトラック法の実行(カッコ内の数字は選択の順序を表す.)

アルゴリズム 9.1

部分和問題を解くバックトラック法を用いたアルゴリズム

入力：実数の集合を表す配列 X[1], X[2], ..., X[n]，および実数 s

```
BT_subsetsum(level) {
  if (level>n) {
    sum=0;
    for (i=1; i<=n; i=i+1) { sum=sum+Y[i]*X[i]; }
    if (sum==s) { Y[1], Y[2], ..., Y[n]を出力し，アルゴリズムを終了; }
  }
  else {
    Y[level]=0; BT_subsetsum(level+1);    //X[level]を足さない場合
    Y[level]=1; BT_subsetsum(level+1);    //X[level]を足す場合
  }
  if (level==1) "s に等しい選び方が存在しない"と出力;
}
  //BT_subsetsum(1)と指定して実行することにより，部分和問題の解が求められる．
```

計算量

- 葉の数 $O(2^n)$
- 葉での計算量 $O(n)$
- 計算量の全体 $O(n2^n)$

分岐限定法

- 条件を設定して、バックトラック法の列挙木を小さくする（探索の対象を外す）

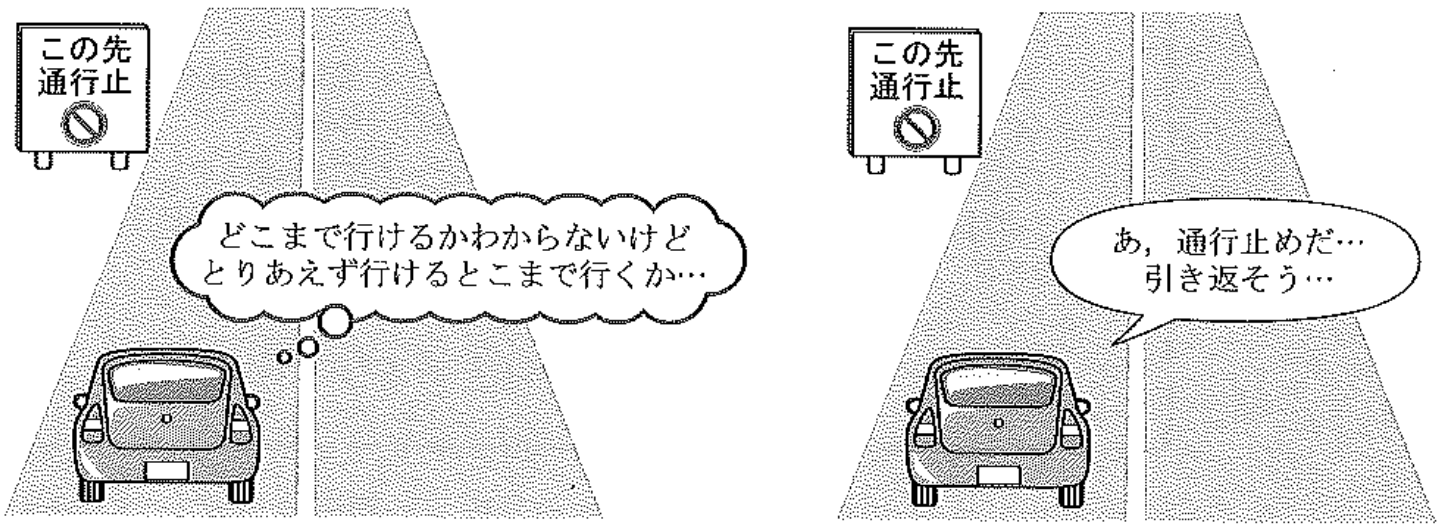


図 9.4　バックトラック法と分枝限定法のアイデアによる車の運転

分枝限定法　部分和問題の場合

【問題 9.1】　部分和問題

$\{x_1, x_2, \ldots, x_n\}$ という n 個の正の実数の集合と，s という正の実数が与えられたとする．このとき，$\{x_1, x_2, \ldots, x_n\}$ の中からその和がちょうど s になる実数の選び方を求めよ．

例えば3、14、6、9の整数を使って、12になる組み合わせをすべて調べる場合を考える。

3 + 14 +(16) + (9) = 17（16と9も足す予定だった）

和が12にならないもの（条件が外れるもの）を除外する。

分枝限定法　部分和問題の場合

アルゴリズム 9.2　部分和問題を解く分枝限定法を用いたアルゴリズム

入力：実数の集合を表す配列 X[1], X[2], ..., X[n], および実数 s

```
BB_subsetsum(level) {
  if (level>n) {
    sum=0;
    for (i=1; i<=n; i=i+1) { sum=sum+Y[i]*X[i]; }
    if (sum==s) { Y[1], Y[2], ..., Y[n]を出力し,アルゴリズムを終了; }
  }
  else {
    sum1=0; sum2=0;
    for (i=1; i<=n; i=i+1) {
      if (i<level) { sum1=sum1+Y[i]*X[i]; sum2=sum1; }
      else { sum2=sum2+X[i]; }
```

分枝限定法　部分和問題の場合

```
    sum1=0; sum2=0;
    for (i=1; i<=n; i=i+1) {
      if (i<level) { sum1=sum1+Y[i]*X[i]; sum2=sum1; }
      else { sum2=sum2+X[i]; }
    }
     //sum1は選択した整数の和，sum2は選択した整数と未選択のすべての整数の和
    if ((sum1<=s)かつ(sum2>=s)) { //枝刈りの判定
      Y[level]=0; BB_subsetsum(level+1); //X[level]を足さない場合
      Y[level]=1; BB_subsetsum(level+1); //X[level]を足す場合
    }
  }
  if (level==1) "sに等しい選び方が存在しない"と出力;
}
```

分枝限定法　部分和問題の場合

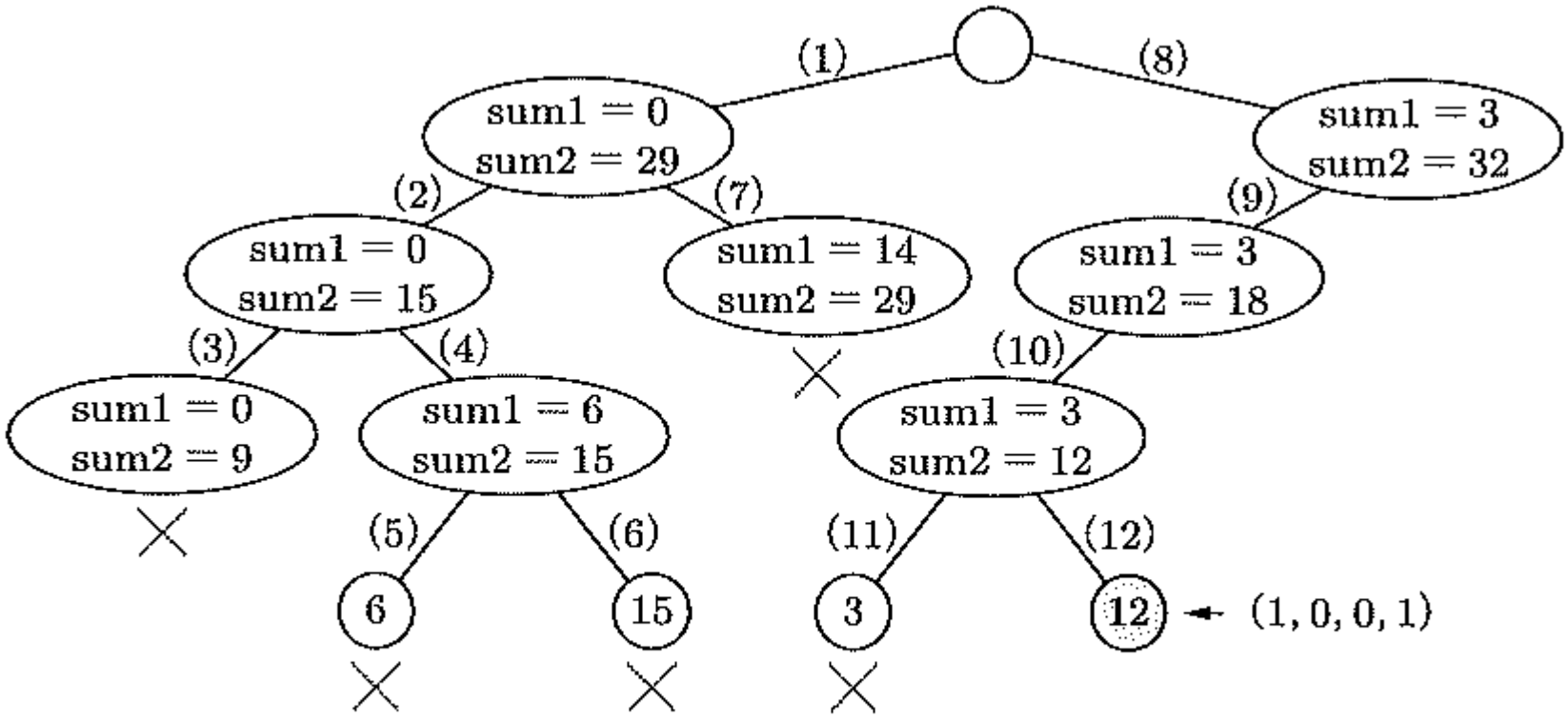


図 9.5　部分和問題に対して分枝限定法を用いた場合の列挙木($s = 12$ の場合)

教科書　0-1ナップサック問題

- 教科書は0-1ナップサック問題に例にしているが、８クイーン問題を例とする。

8クイーン問題

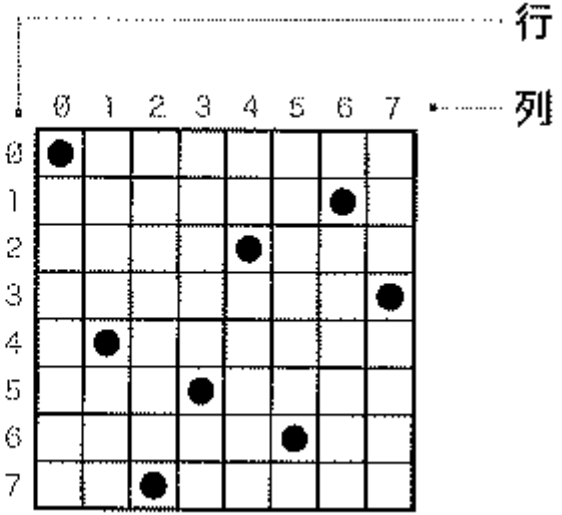
8王妃問題とは

本節で学習する8王妃問題（8–Queen problem）は、

互いに取りあえないように、8個の王妃を8×8のチェス盤に配置せよ。

チェスのクイーンは立て・横・斜めの移動ができる。

互いに取りあえないように
8個の王妃を配置する



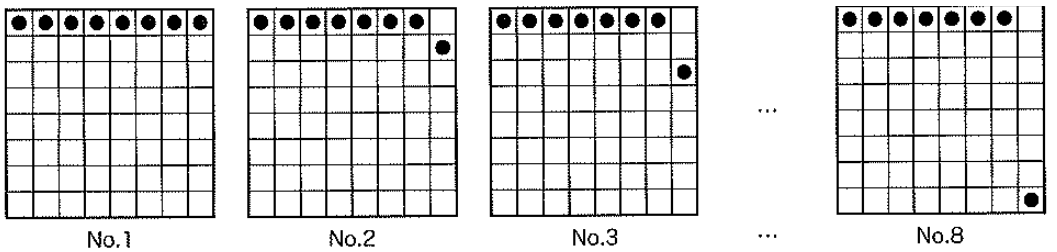
8クイーン問題の難しさ

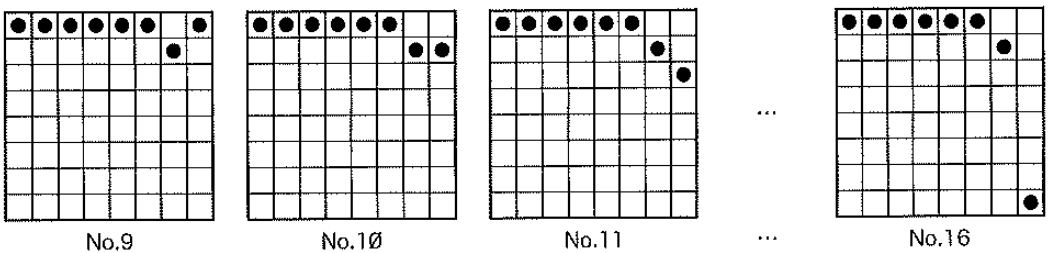
- 最初にクイーンを置ける場所は64
- その次は63、その次は62
- 組み合わせとしては、非常に大きく、簡単ではない。

$$64 \times 63 \times 62 \times 61 \times 60 \times 59 \times 58 \times 57 = 178{,}462{,}987{,}637{,}760$$

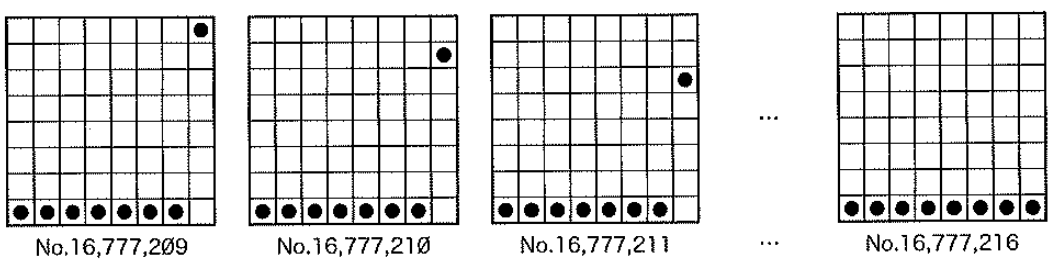
8クイーン問題の考察

【方針１】各列には王妃を１個だけ配置する。





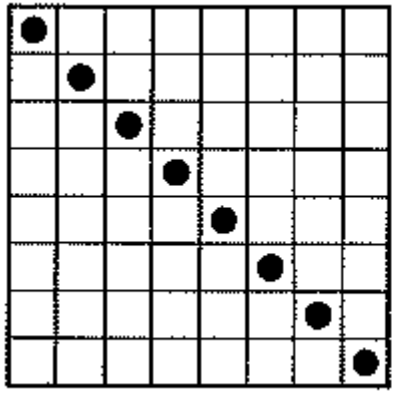
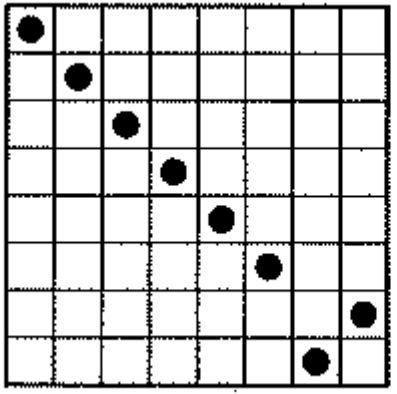
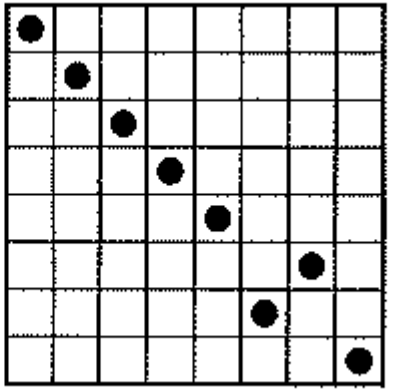
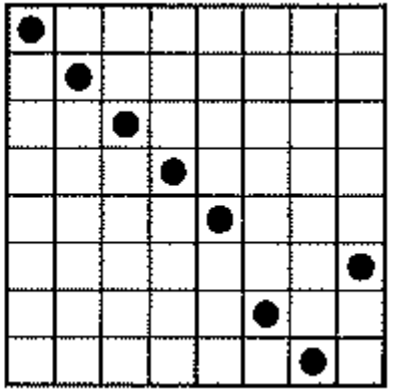
中略



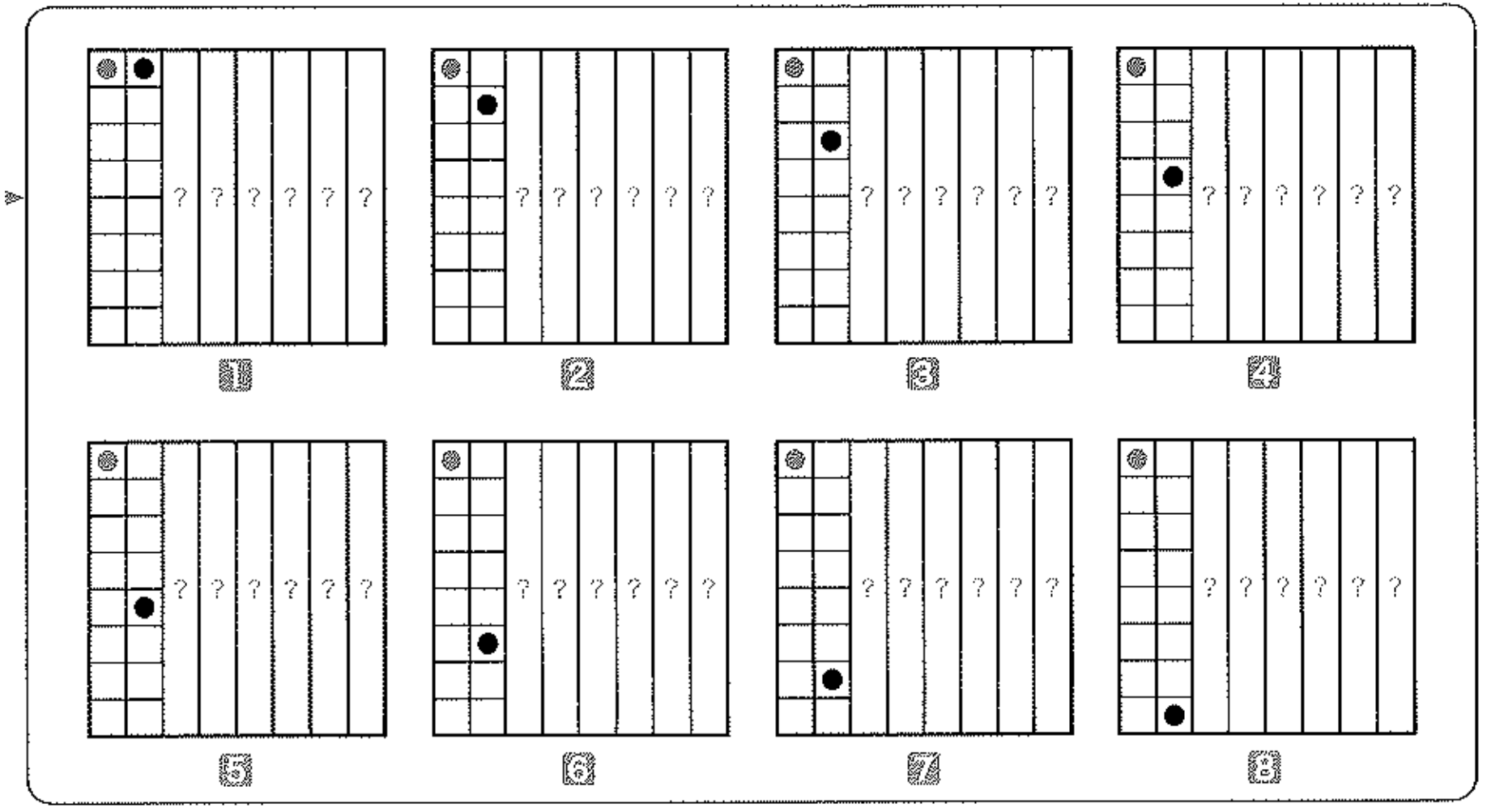
8クイーン問題の考察

初期配置を工夫して、列挙木を調べる。

【方針2】 各行には王妃を1個だけ配置する。

8クイーン問題の考察



8クイーン問題の考察　分岐操作

i列目に配置された王妃の位置がj行目であればpos[i]の値をjとする。

```
set(0);                    // 0列目に王妃を配置

set(i + 1);                // 次の列に王妃を配置
```

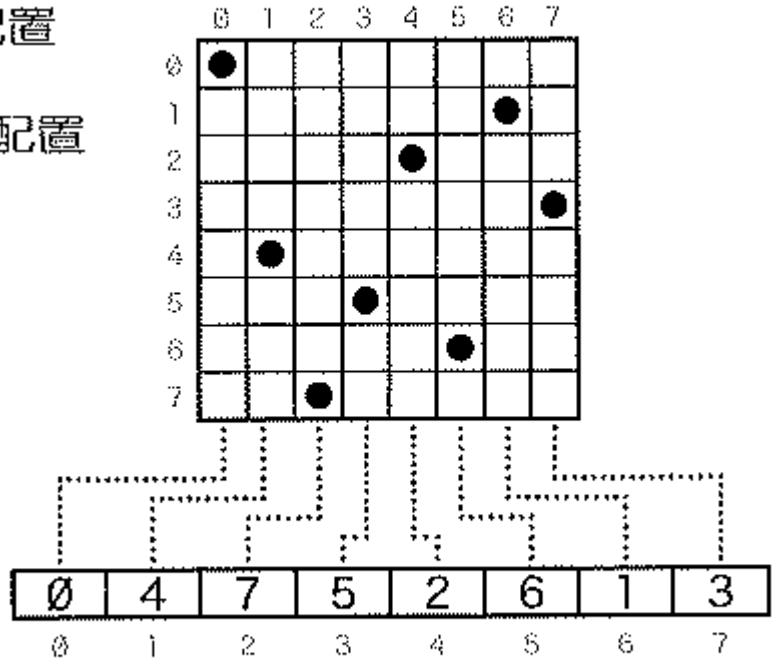
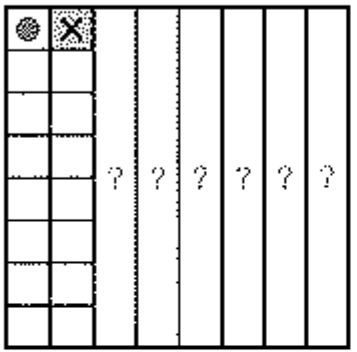


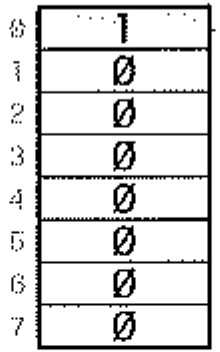
Fig.5-20　王妃配置を表現する配列

8クイーン問題の考察　分岐操作

Ø行Ø列に王妃が配置ずみである状態での、１列目への王妃の配置の検討

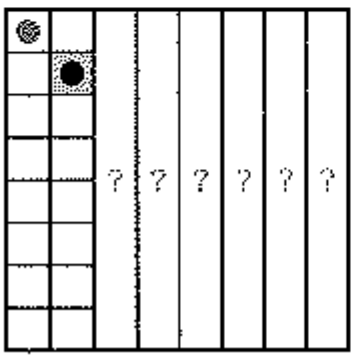
a

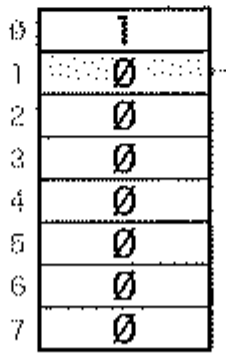




Ø行目には王妃が配置ずみなので
この組合せを考える必要はない

b





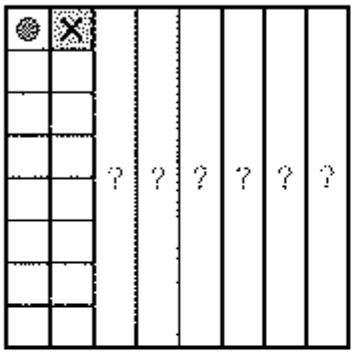
１行目には王妃が未配置なので、
そこに王妃を配置する

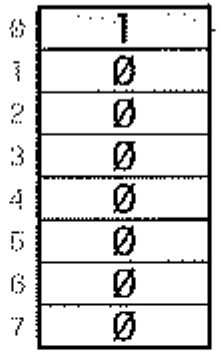
⋮

8クイーン問題の考察　分岐操作

Ø行Ø列に王妃が配置ずみである状態での、１列目への王妃の配置の検討

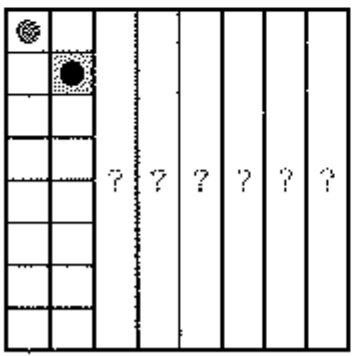
a

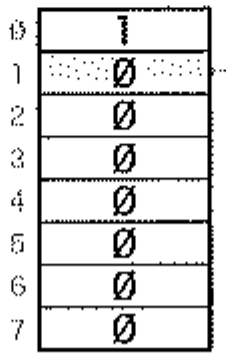




Ø行目には王妃が配置ずみなので
この組合せを考える必要はない

b



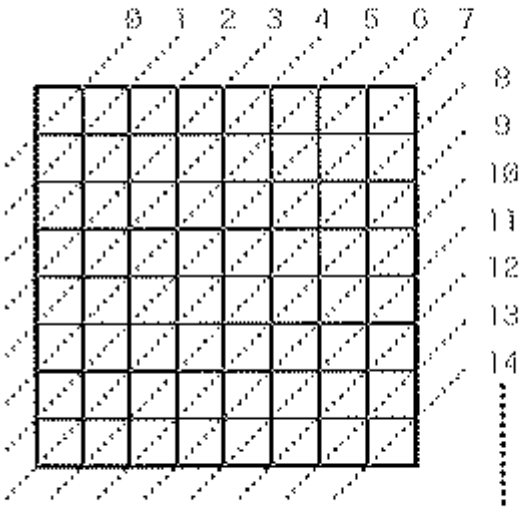


１行目には王妃が未配置なので、
そこに王妃を配置する

⋮

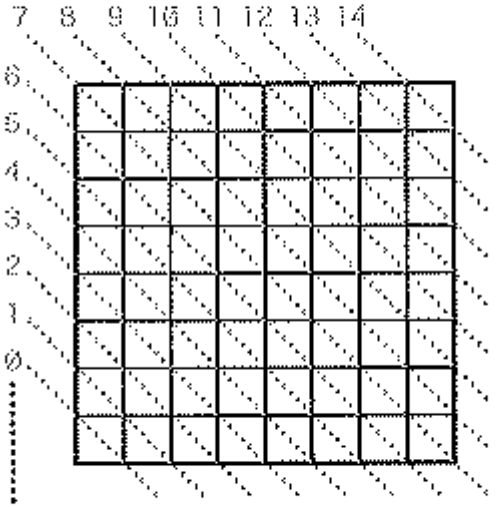
8クイーン問題の考察　斜めの配置の考慮

a 配列 flag_b に対応するライン



j 行 i 列の値は i+j によって得られる

b 配列 flag_c に対応するライン



j 行 i 列の値は i−j+7 によって得られる