アルゴリズム論
（第12回）

岩手県立大学
Iwate Prefectural University

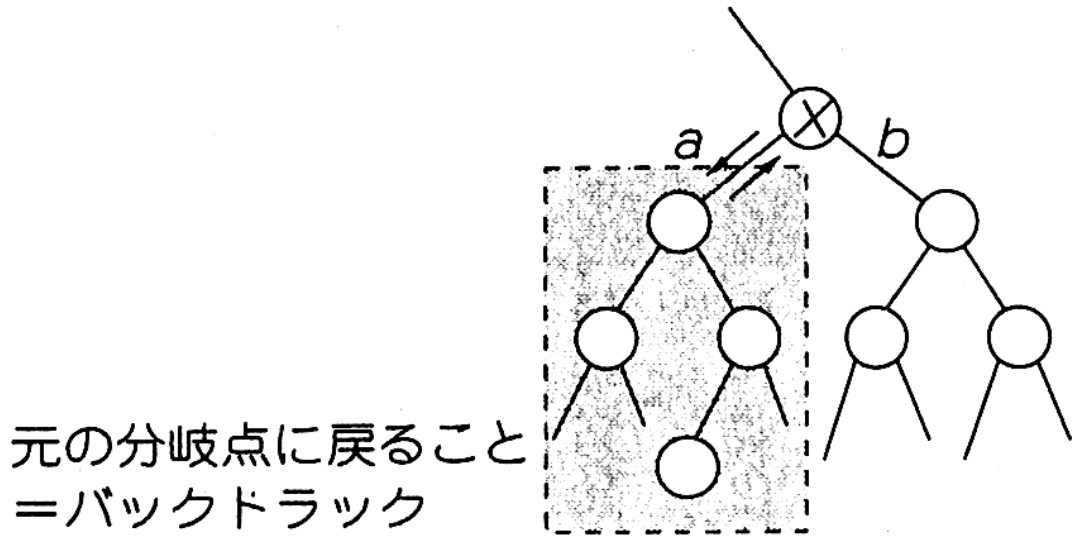
佐々木研（情報システム構築学講座）
講師　山田敬三
k-yamada@iwate-pu.ac.jp

バックトラッキング

岩手県立大学
Iwate Prefectural University

バックトラッキングとは

- 分岐点における1つの選択肢からの展開を全て調べた後に、またその分岐点に戻ること。

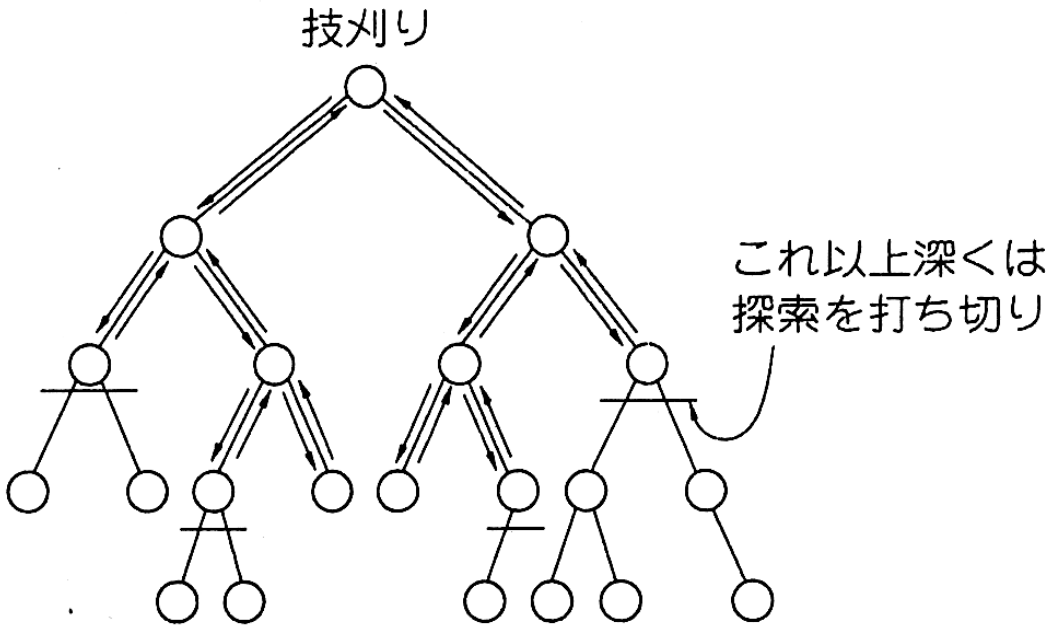


バックトラッキングとは

- 木構造のデータを全て調べていくことは無駄が多い。
- 最適解を調べたいとき、その条件を超える場合は、経験的にそれ以上調べる必要がないことはわかる。
- 探索ルートと途中で戻ることを枝刈りという。

バックトラッキングとは

- バックトラッキングと枝刈りを組み合わせて使うことで効率よく問題を解決できる。



ナップサック問題

『n個の荷があり、それらの重さは $a_1, a_2, \cdots, a_n$ である．重さWまで耐えるナップサックがあり，それにぴったりになるように荷を選びたい．それらの全ての組み合わせを求めよ．』

- nが大きくなると大変解きにくい。
  - 全体の組み合わせは 2^n
    （例） n=30 のとき　$2^{30} \fallingdotseq 10^9$

ナップサック問題

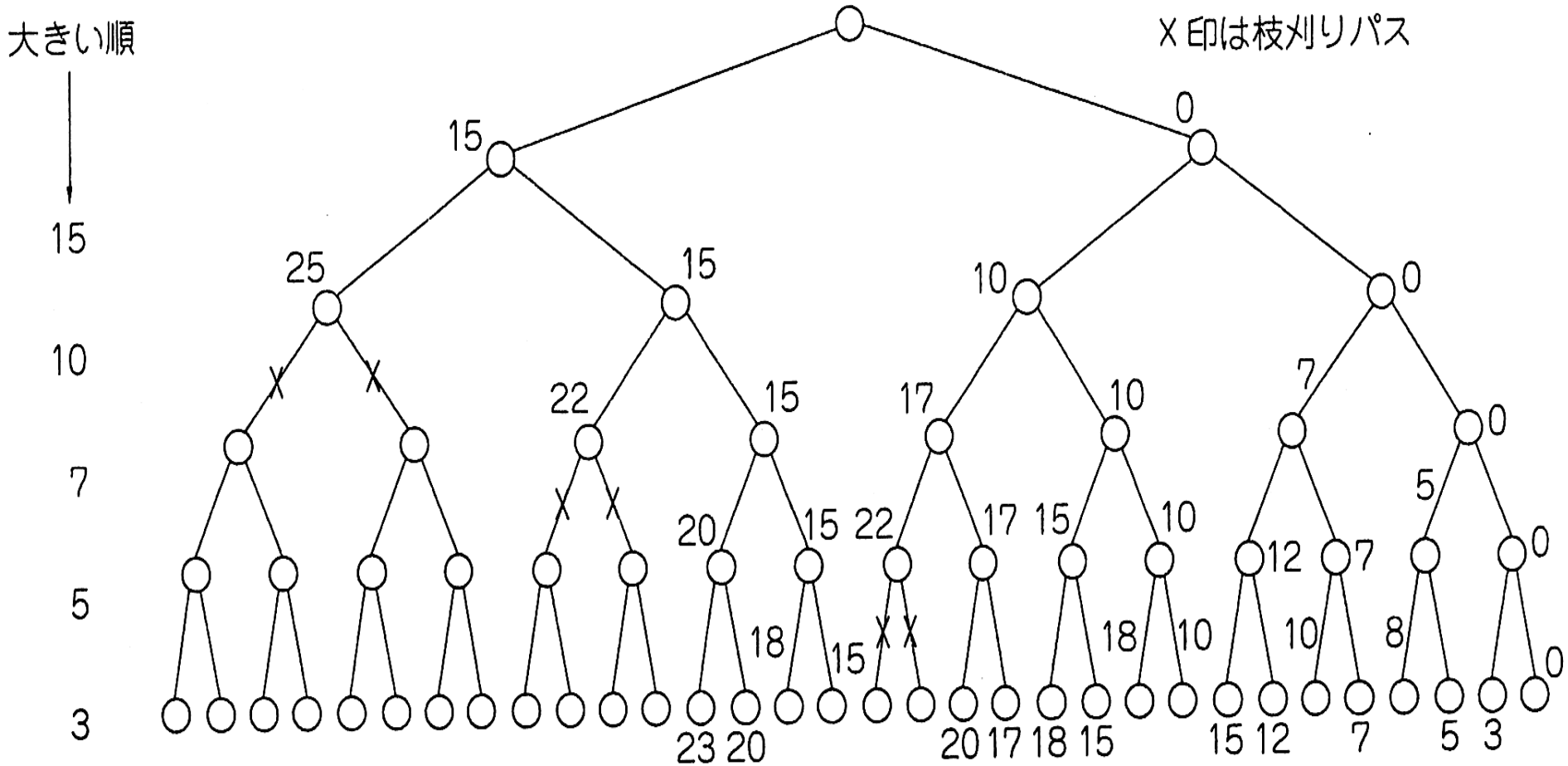
- 枝刈りの方針

『全ての組み合わせを調べる途中で，それまでの重さの計がすでにWを超えていたら，それ以後の組み合わせは考えない』

ナップサック問題

（例題）
重さ、１０，５，１５，７，３の5つの荷があり, W=22のときの全ての組み合わせを求めよ。

ナップサック問題

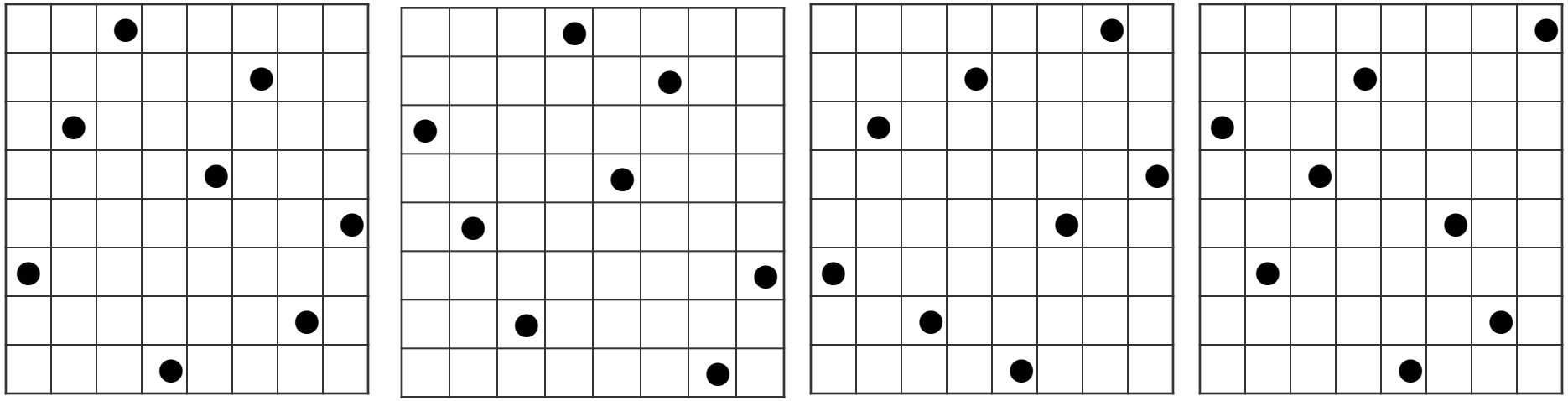

大きい順
15
10
7
5
3
X 印は枝刈りパス
15
0
25
15
10
0
22
15
17
10
7
0
20
15
22
17
15
10
12
7
5
0
18
15
18
10
10
8
0
23 20
20 17 18 15
15 12
7
5 3

枝刈りの適用例

- 8人の女王問題（エイト・クイーン）
  - 8人の女王を８×８の盤に配置する問題
  - 条件
    - 1人の女王から、その行と列およびその位置から見える対角線上に他の女王がいてはいけない
  - 置き方の総当り数は？
    - ${}_{64}C_8$ = 4,426,165,368
  - 女王は同じ行列にいることはできないので、
    8! = 8・7・6・5・4・3・2・1=40,320
    となる

配置例

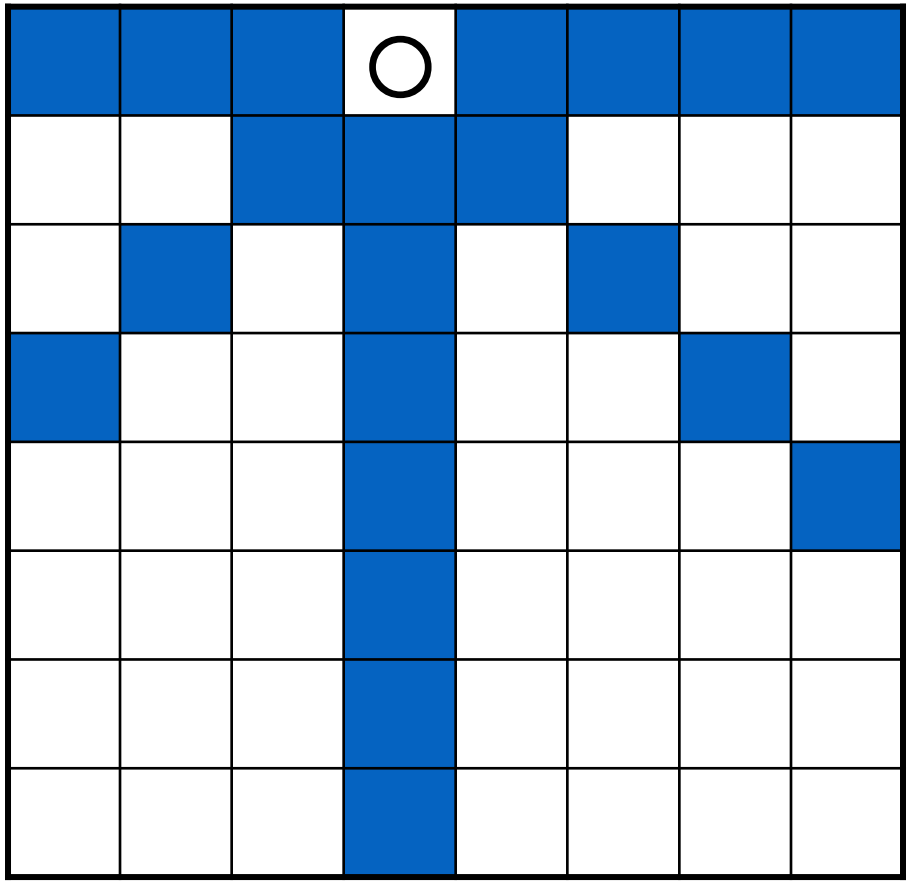
✓8人の女王を8×8の盤面に配置する.
✓ある女王を配置した行列およびその対角線上には
　他の女王を置いてはいけない.



配置例

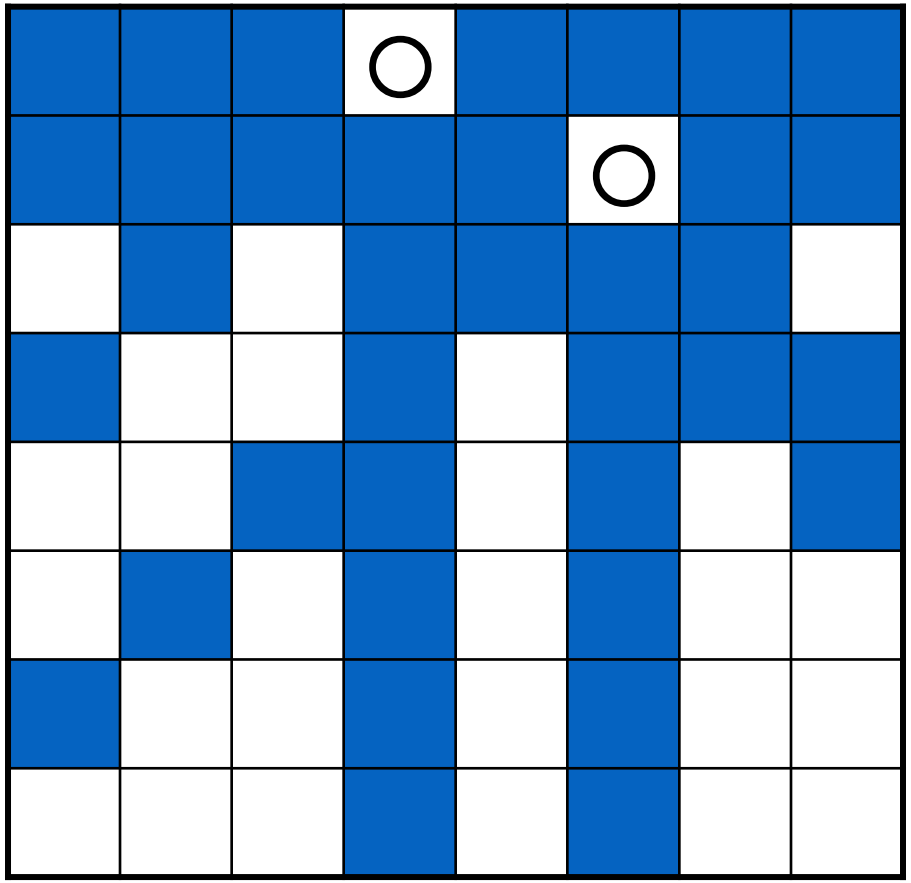
枝刈りの適用例

- 対角線上にもいないように置く
- あるパスが見込み無い場合パスの先の検索をやめる



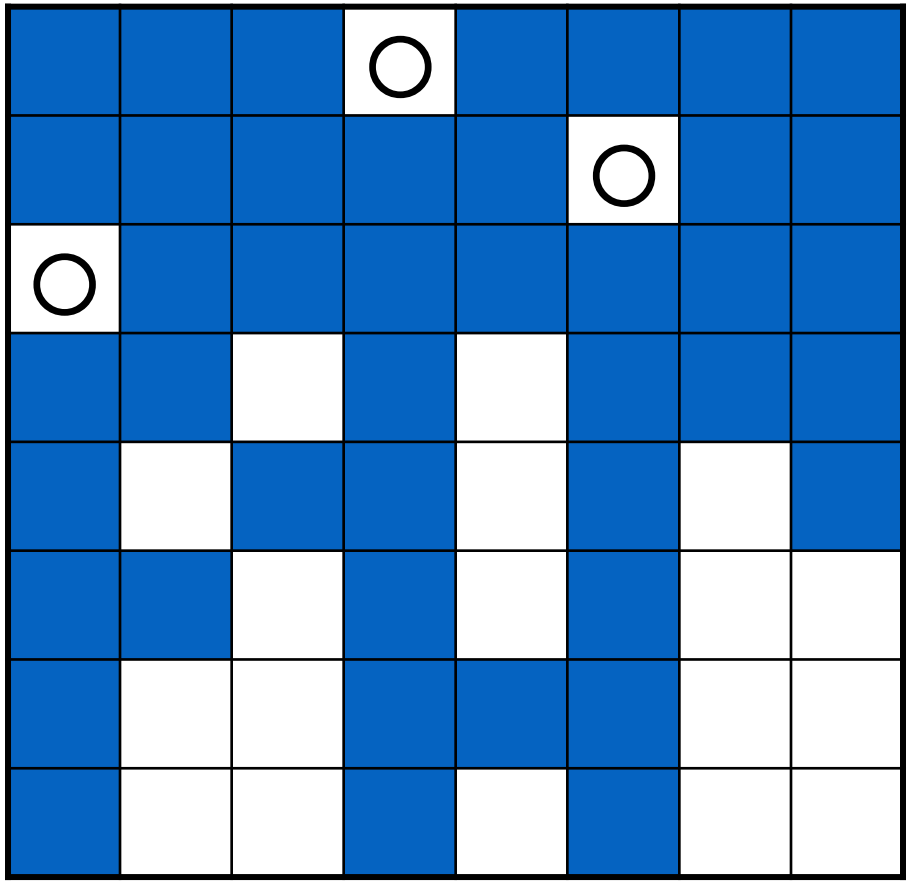
枝刈りの適用例

- 対角線上にもいないように置く
- あるパスが見込み無い場合パスの先の検索をやめる



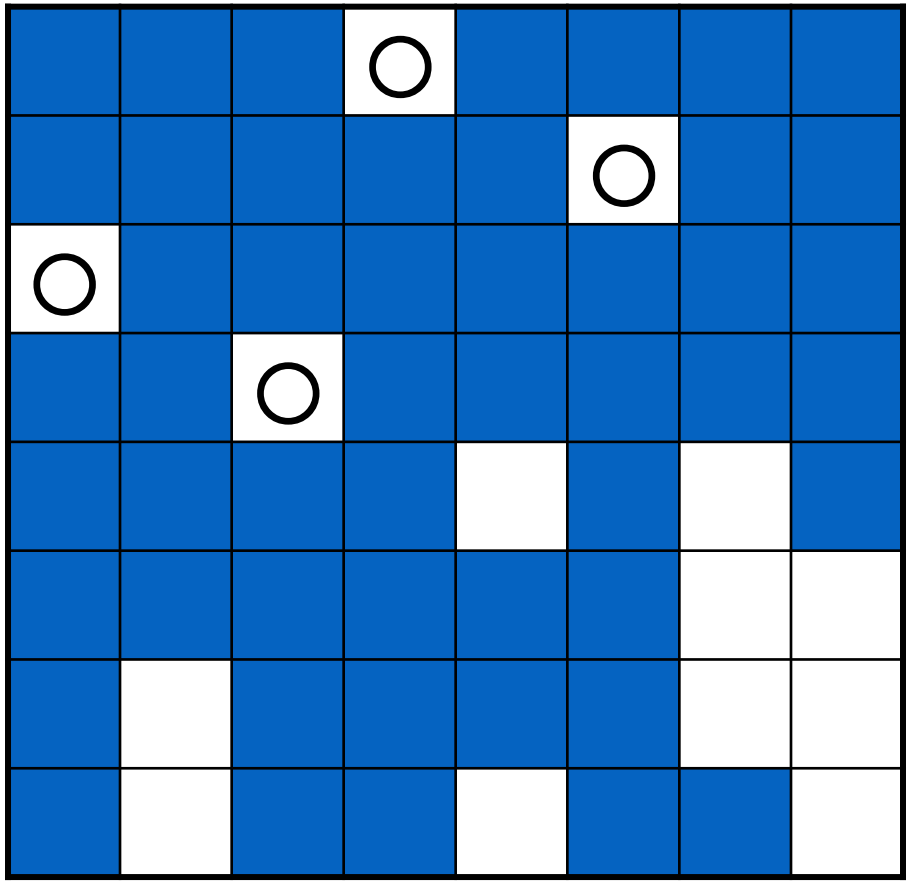
枝刈りの適用例

- 対角線上にもいないように置く
- あるパスが見込み無い場合パスの先の検索をやめる



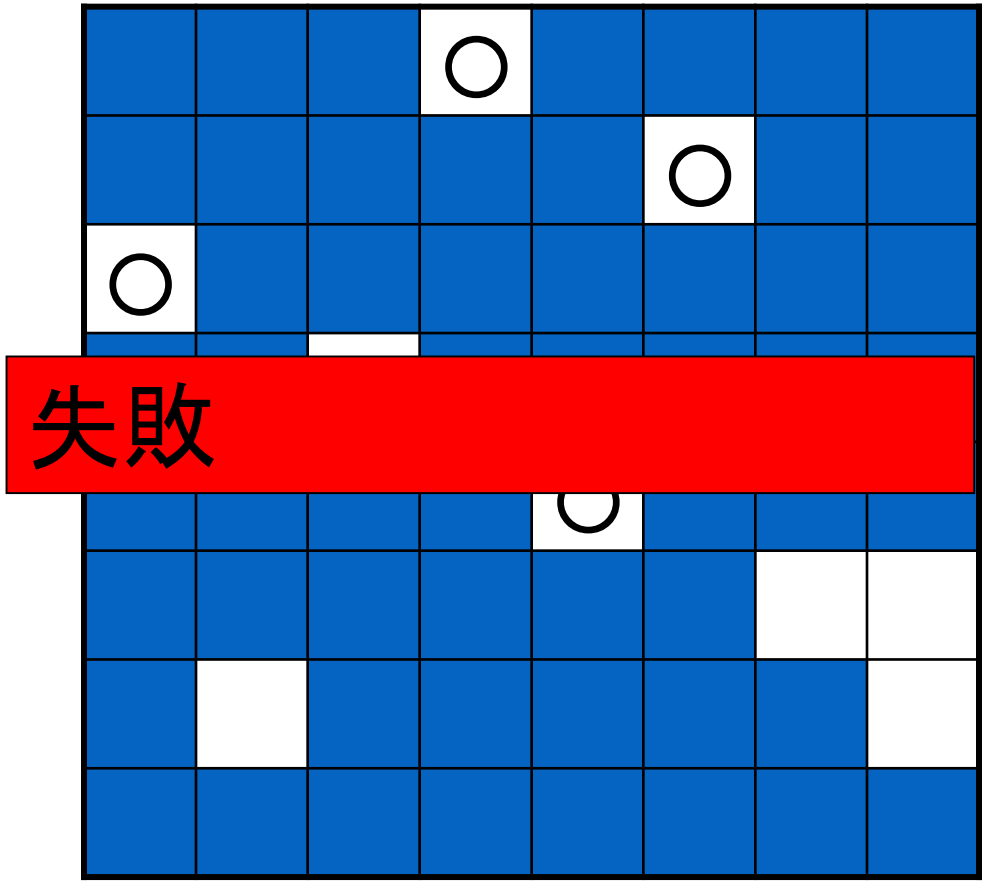
枝刈りの適用例

- 対角線上にもいないように置く
- あるパスが見込み無い場合パスの先の検索をやめる



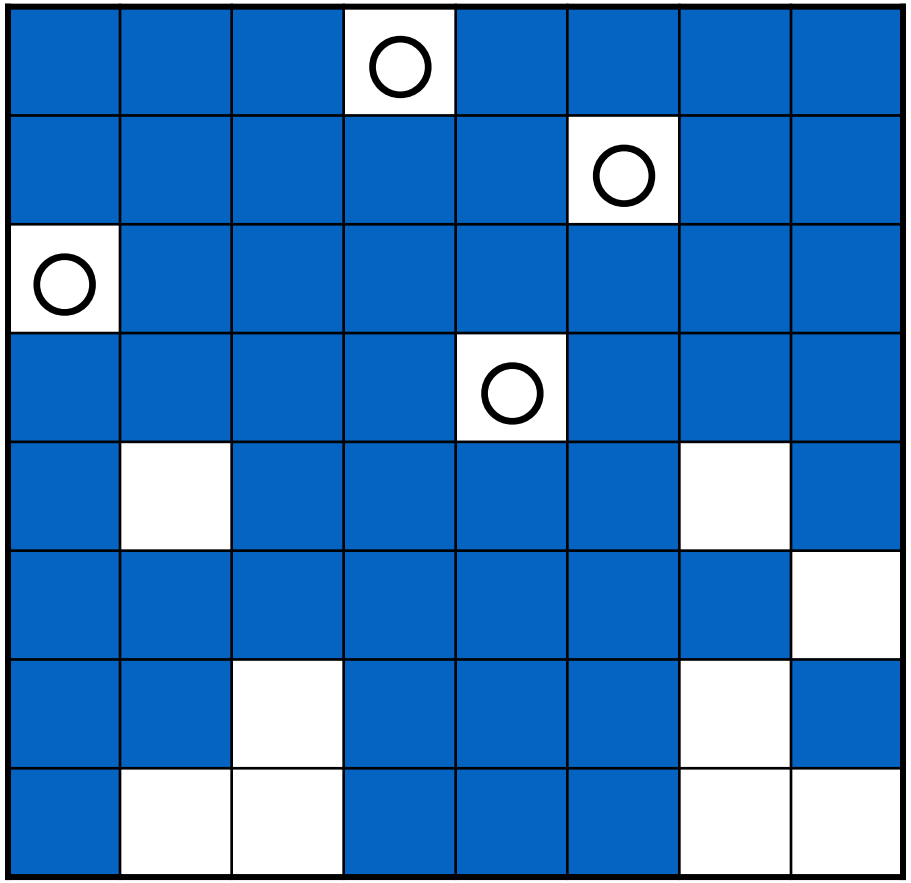
枝刈りの適用例

- 対角線上にもいないように置く
- あるパスが見込み無い場合パスの先の検索をやめる



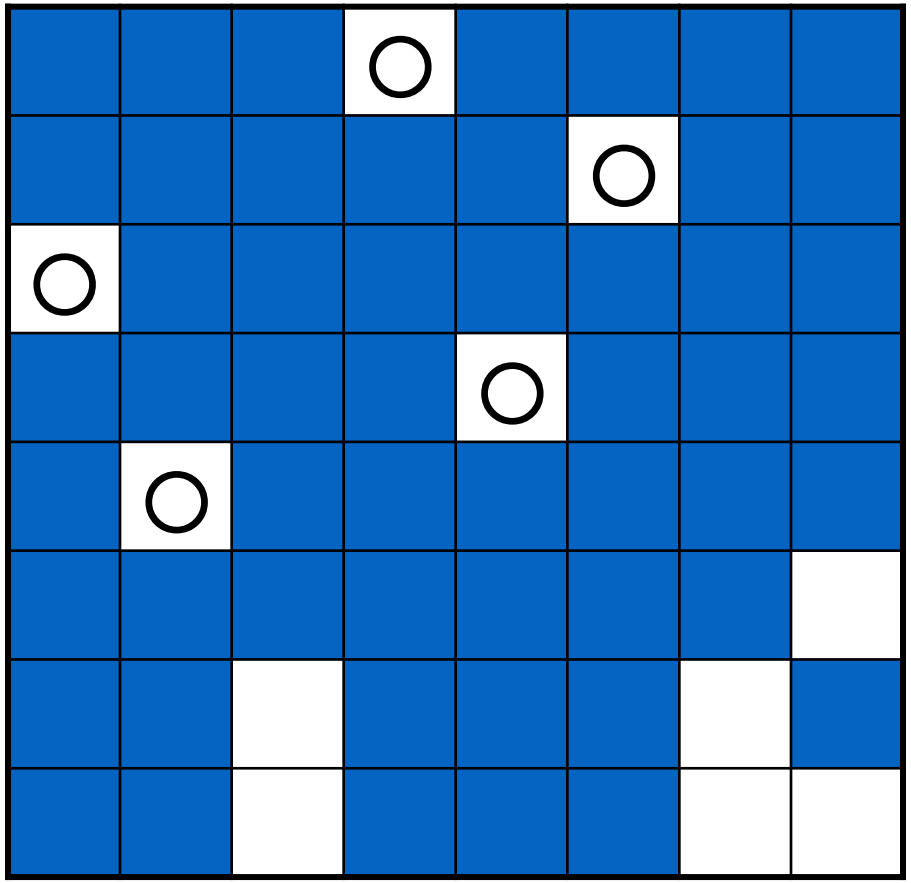
枝刈りの適用例

- 対角線上にもいないように置く
- あるパスが見込み無い場合パスの先の検索をやめる



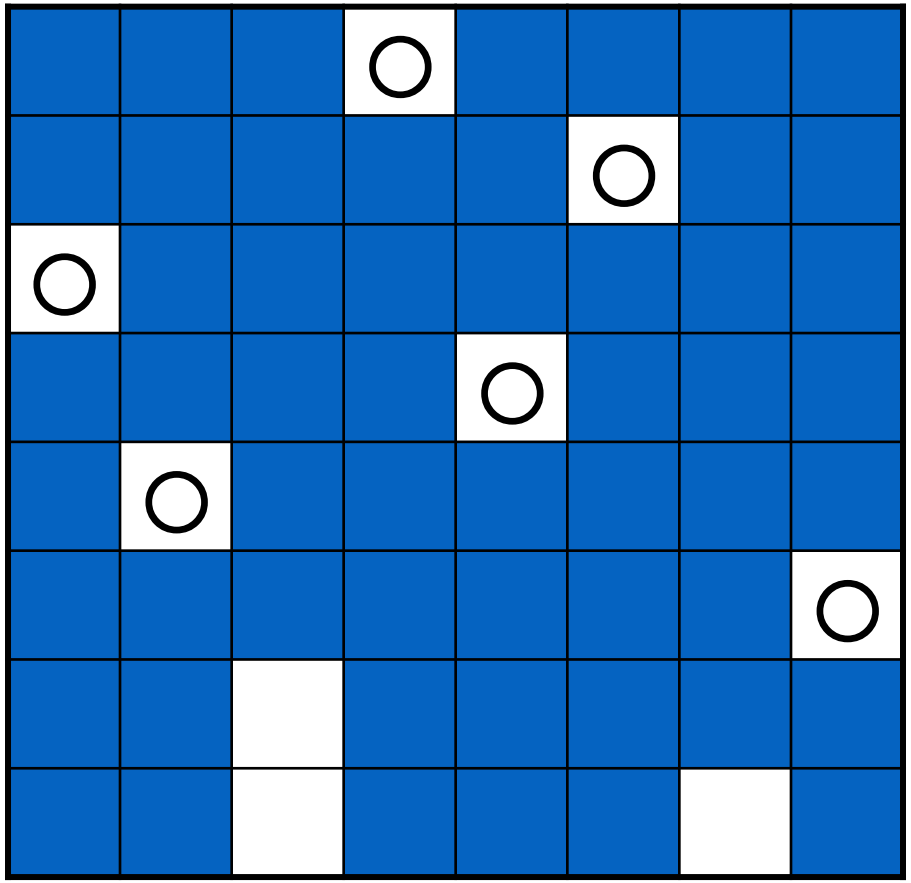
枝刈りの適用例

- 対角線上にもいないように置く
- あるパスが見込み無い場合パスの先の検索をやめる



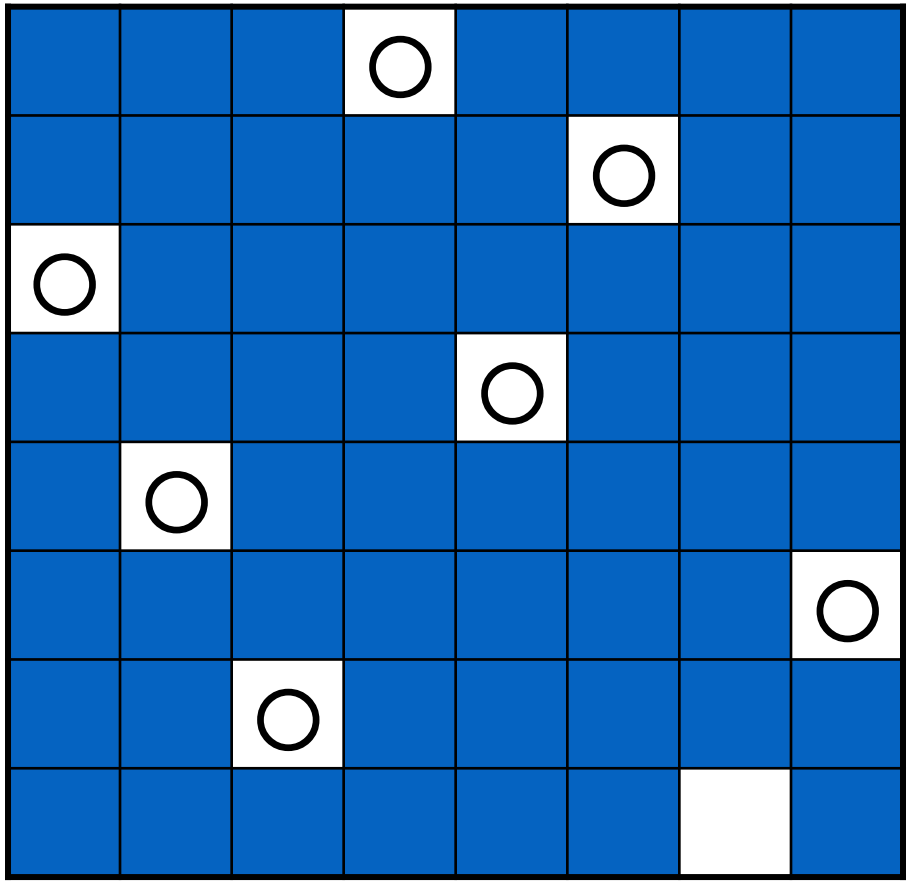
枝刈りの適用例

- 対角線上にもいないように置く
- あるパスが見込み無い場合パスの先の検索をやめる



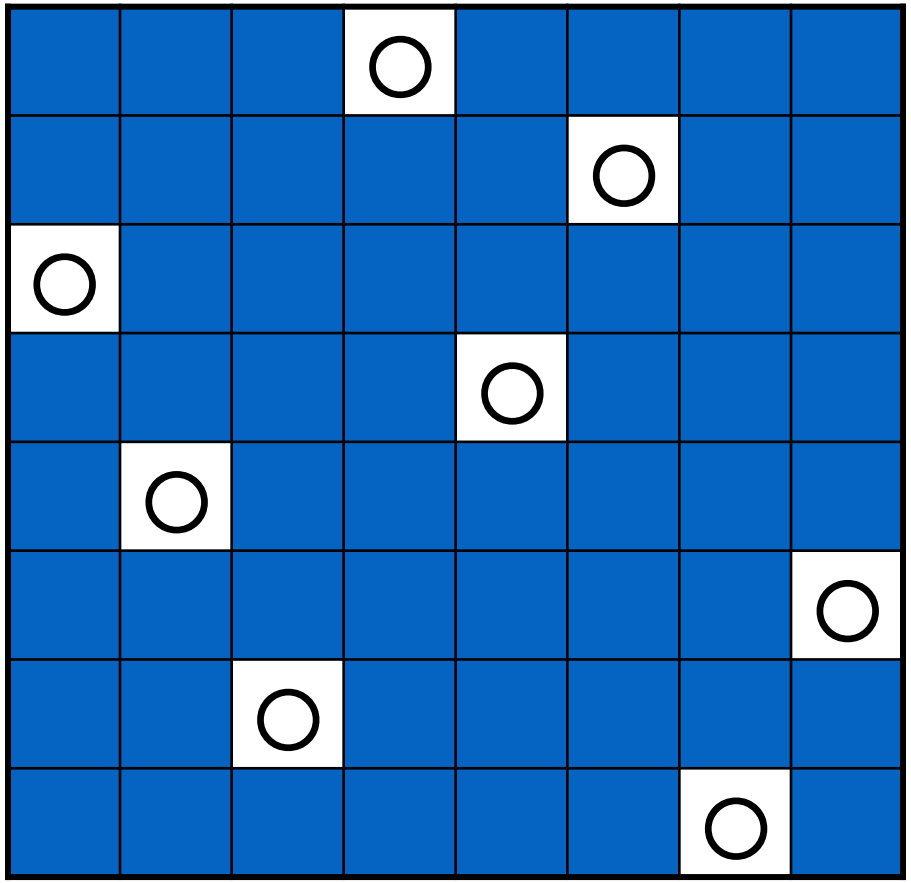
枝刈りの適用例

- 対角線上にもいないように置く
- あるパスが見込み無い場合パスの先の検索をやめる



枝刈りの適用例

- 対角線上にもいないように置く
- あるパスが見込み無い場合パスの先の検索をやめる



構文解析

岩手県立大学
Iwate Prefectural University

用語

- 帰納的に定義された文字列の無限集合を言語という（例：算術式）
- 形式的な規則によって、与えられた文字列が言語に属するか否かが決定される
  規則⇒文法(grammar) or 構文規則（syntax rule)
- 意味規則（semantic rule）：一般的な言語（算術式やプログラミング言語）に対して定められる
  - 算術式の場合、式の意味は式の値として定義される
  - プログラミング言語の場合，命令の実行順序を表す抽象的な記述（抽象的な機械語）として定義される

VSL(Very Simple Language)

- 簡単な例：VSLを定義
- VSLの実現を考える
- 実現方法(implimantation)には
  - インタプリタ（解釈系, interpreter）
  - コンパイラ（翻訳系, compiler）

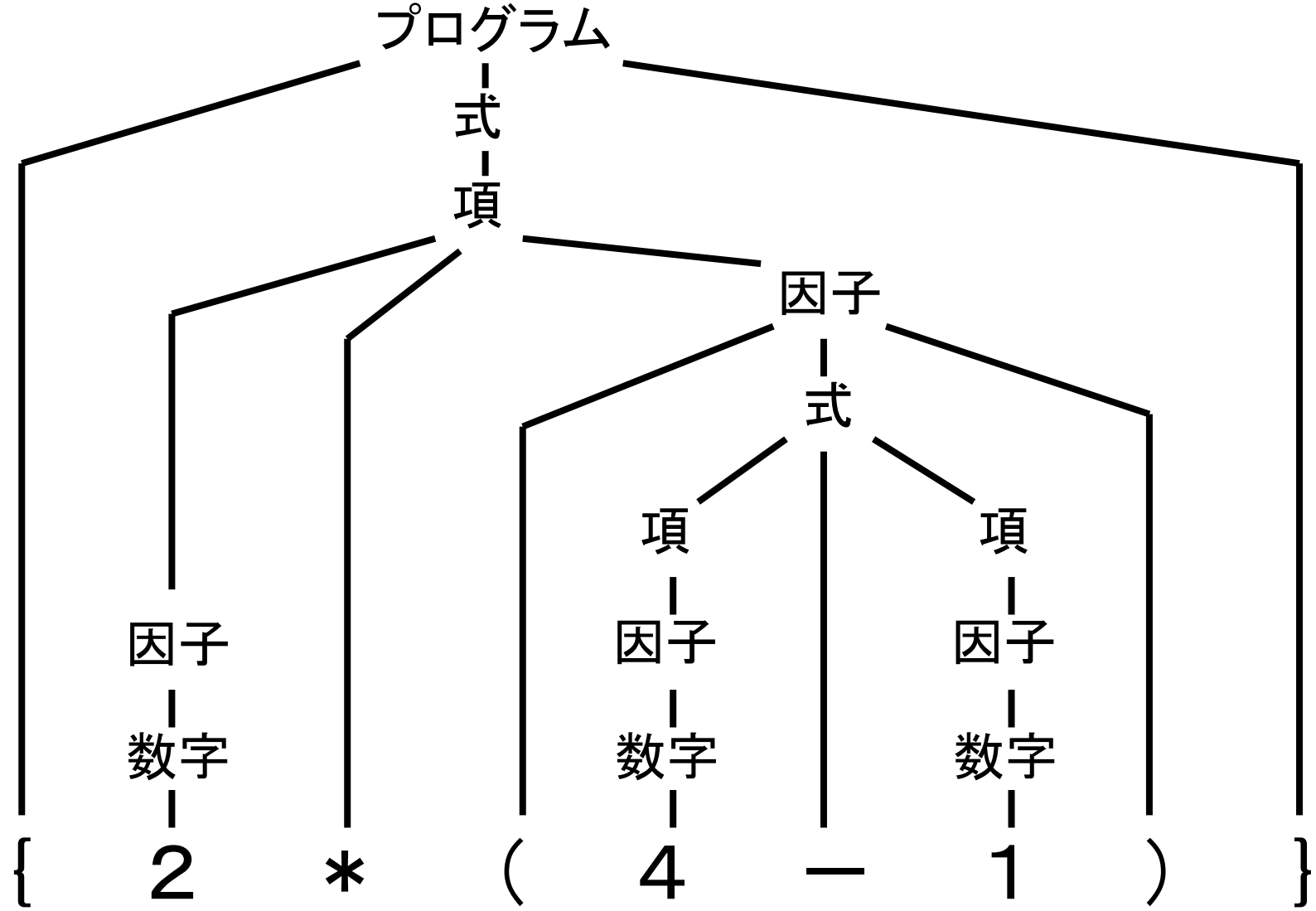
インタプリタとコンパイラ

- インタプリタ
  - 計算機を支配下に置く
  - 入力データの記述に従って処理を進める
  - 入力データは、ソースコードまたは中間コード
- コンパイラ
  - 最終的に独立したプログラムになるコードを生成する
  - 生成されたコードを目的コードとよぶ

構文グラフ

- 図９．１参照
- {，}，数字　：　終端記号またはプリミティブ
- 式　:非終端記号
- 項
- 因子

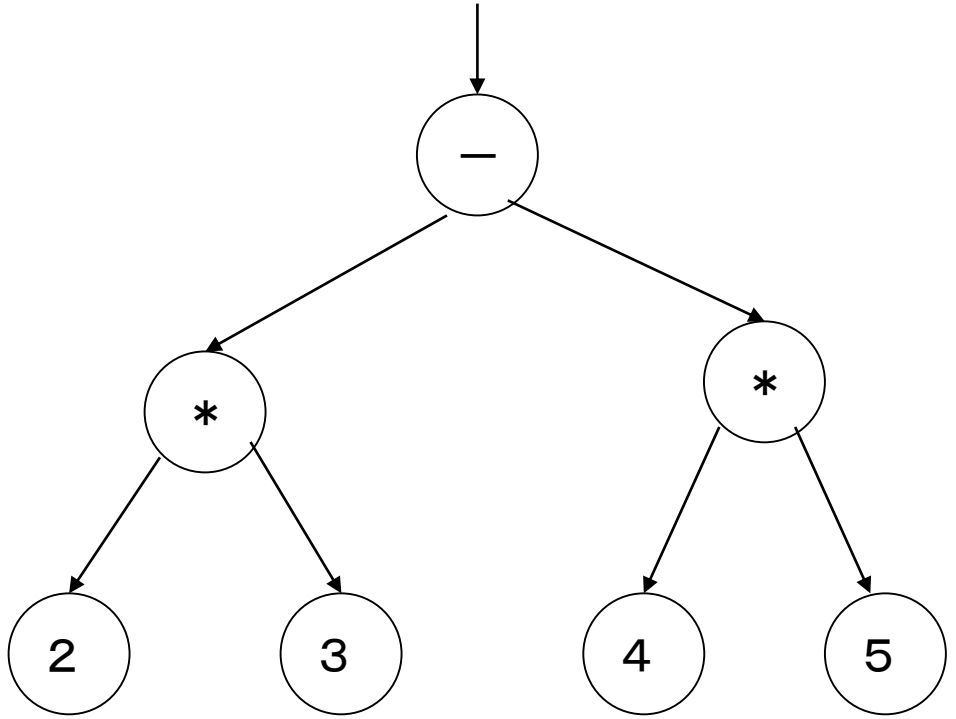
構文グラフ



式と2分木

- 式は2分木で表現できる

  {２＊３－４＊５}

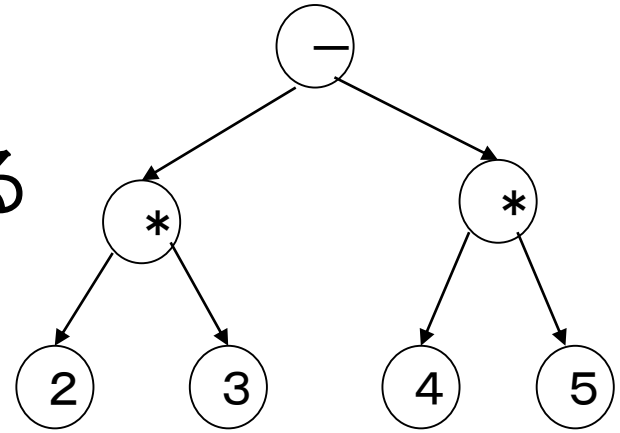


中置記法から後置記法

- 記法
  - ー　８　３　　前置
  - ８　ー　３　　中置
  - ８　３　ー　　後置(逆ポーランド記法)
- 引数を持つ関数は前置記法
  - 8-3=5　は　subtract(8, 3)
- コンパイラの作成を考える上では
  - 逆ポーランド記法が優位

中置記法から後置記法

- 変換方法
  - 中置式は2分木で表記できる
  - 2分木の通り方によって
    前置、中置、後置記法に変わる
  1. 行きがけ順→前置
  2. 通りがけ順→中置
  3. 帰りがけ順→後置



２　３　＊４　５　＊　－

- 後置記法（逆ポーランド記法）ならスタックが使える

ソーステキストインタプリタ

- プログラミング言語の設計
  - 言語構文規則
  - 意味規則
- 任意のVSLプログラムを読み込み、記述された処理を実行するCプログラム
  ⇒ソーステキストインタプリタ
- ソーステキストを解析する方法
  - 再帰降下法（recursive descent)
    再帰的な関数の階層構成に基づいた方法

目的プログラムと実行時システム

- コンパイラ
  - 解釈を必要とする中間コードの代わりに、プログラムの形式を持つ実行可能なコードを生成する
- 現在のコンパイラ
  - ソースコードから機械語を生成する
    - 途中でアセンブリコードに変換することもある
  - 中間コードとの違い
    - 中間コードは入力データである

ソーステキストの解析

- 因子などの構文的概念のそれぞれに対して1つの関数を用意する
- 大域変数 buf を準備
  - bufが0でないときだけその内容を使用
- 関数factorの機能は
  - 入力ファイルから1つの因子を読み込む
  - 因子を評価し、その値を返す

ソーステキストの解析

- 関数next
  - bufが0のときだけファイルから文字を読む
  - bufが0でないとき、bufの値を使用し、buf=0とする
  - 空白と改行は無視する
- 関数nextis
  - 与えられた文字が入力ストリームから読み込むことができるかどうかを調べる
  - 読み込めれば1を返す、そうでなければ0

ソーステキストの解析

- 関数term
  - 因子を読んで＊が続くまで読み込む
  - 各因子に対して＊の演算を行う
- 関数expression
  - 項を読んで、＋－の演算を行う

ここまでのまとめ

- INTERPR.C
  - 直接ソースコードを解釈して結果を出す
- POSTFIX.C
  - 中間コードとして後置式を出力する
- 構文解析と計算を分離する
- 中間コードを作成する方が効率がよい
  - 同じ種類の計算が何度も実行される繰り返しがあるときに、処理を1度で済ますことが出来る
- 現在のインタープリタは中間コードに変化、その後解釈する

この課題では

- 目的プログラムをC言語とする
  - VSLからC言語へのコンパイラ
  - P.300のVSLをP.301のC言語へ変換

1. OBJECT.Cをコンパイラから生成
2. OBJECT.CをCでコンパイル
3. RTSとリンクする（RTS.Cはライブラリ）
4. 実行

この課題では

- コンパイラとしてPOSTFIX.Cを変更
  P.304～P.306
- 新POSTFIX.CはVSLを後置法（逆ポーランド記法）に変換
- 変換された後置記法の表現をOBJECT.CのC言語に変換