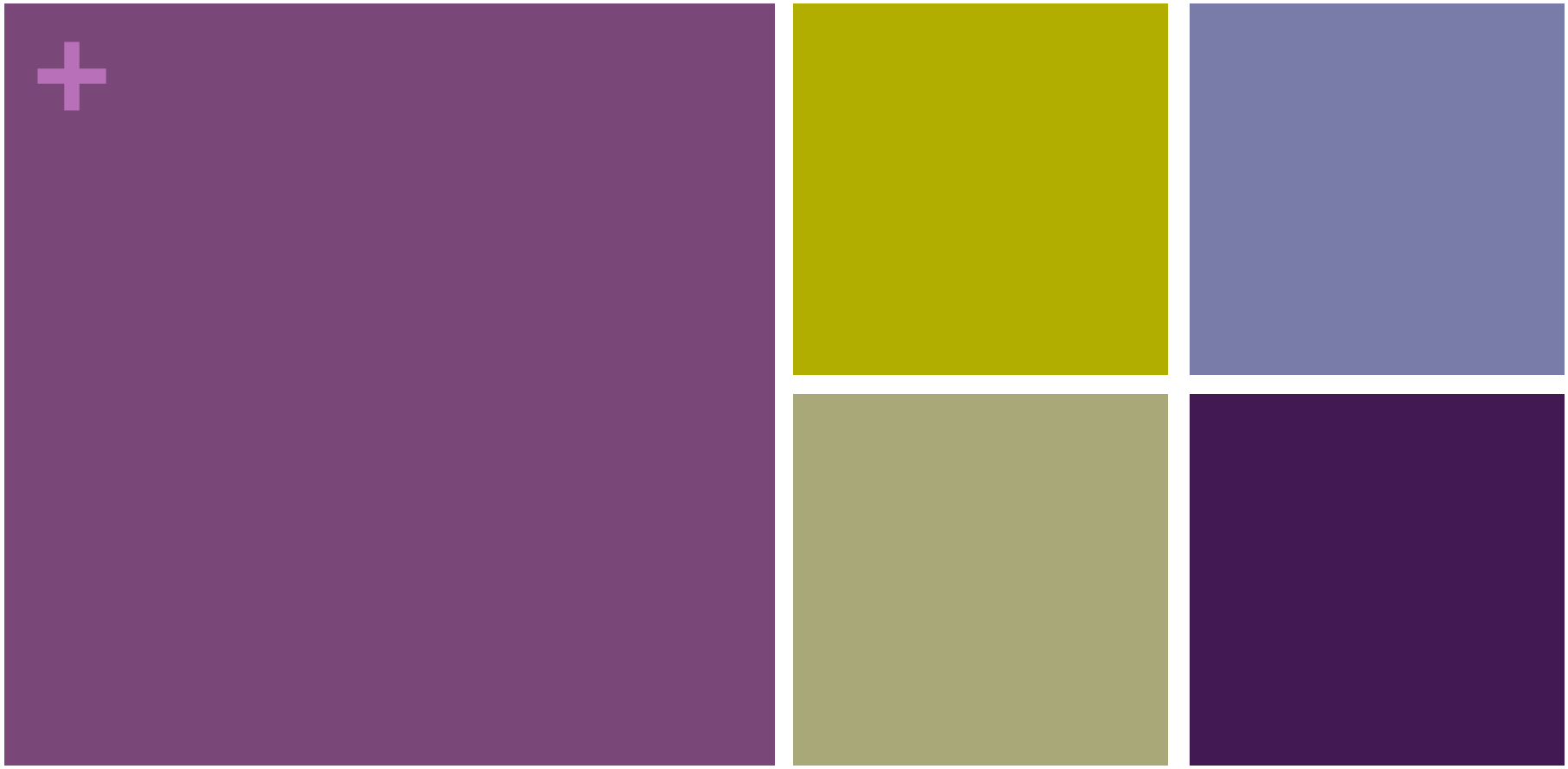


E: 小野小町の編集合戦

原案: 村主
解答: 橋本
問題文・解答・解説: 山口

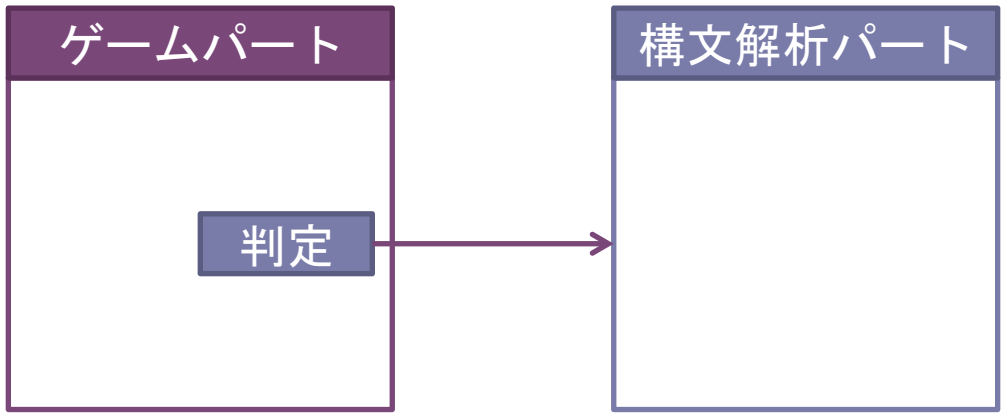
概要

- 2人で交互に数式を編集しあうゲーム
  - 可能なこと
    - 1文字削除
    - 1文字挿入
    - 無効な数式になる場合は不可
  - 例: (1+2)*3
    - 1文字挿入⇒ (91+2)*3
    - 1文字削除⇒ (12)*3
- 先手は値の最大化・後手は値の最小化を狙う
- 両者が最善を尽くした時の結果は?

方針

1. 数式の値を求める
   ⇒ 構文解析: 再帰下降
2. 2人ゲームで最善を尽くしたときの結果
   ⇒ ゲーム木探索: minimax 法

- プログラムの構成



モジュール化しやすい
⇒役割分担しよう

構文解析パート

再帰下降構文解析

構文解析

1. まず BNF / EBNF を書きましょう
   - ルールは優先順位の降順にする
2. BNFの各行に対応する関数を書きましょう
   - 再帰下降構文解析が簡単 (LL系)

```
exp      = or;
or       = or, [ "|", xor ];
xor      = xor, [ "^", and ];
and      = and, [ "&", plus ];
plus     = plus, [ ("+" | "-"), term ];
term     = term, [ "*", factor];
factor   = num | "(", or, ")";
num      = nonzero, { digit };
nonzero  = "1" | ... | "9";
digit    = "0" | ... | "9";
```

EBNF

```
private long or() throws ParsingException {
   long ans = xor();
   while (tokenizer.hasNext() && match('|')) {
      tokenizer.poll(); ans |= xor();
   return ans;
}
private long xor() throws ParsingException {
   long ans = and();
   while (tokenizer.hasNext() && match('|'))
      tokenizer.poll(); ans |= and();
   return ans;
}
```

Parser

注意点

- 左再帰に注意!
  - よく見るとEBNFが左側で再帰している
    - 一般にこういうものを素直に実装すると Stack Overflow になる
  - 解決法
    - 左再帰を含まないEBNFに直す
      - 一般には面倒なことが多い
    - ループに書きなおす
      - オススメ
      - 例えば前頁の図

```
exp     = or;
or      = or, [ "|", xor ];
xor     = xor, [ "^", and ];
and     = and, [ "&", plus ];
plus    = plus, [ ("+" | "-"), term ];
term    = term, [ "*", factor];
factor  = num | "(", or, ")";
num     = nonzero, { digit };
nonzero = "1" | ... | "9";
digit   = "0" | ... | "9";
```

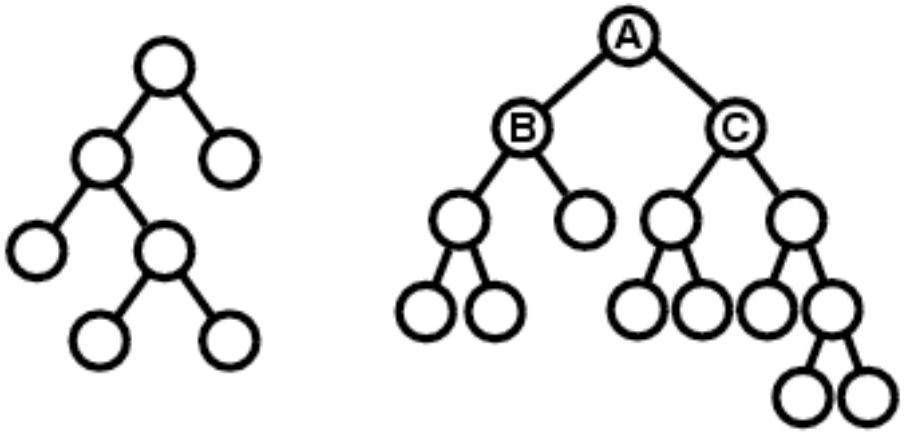
EBNF

関連する問題

- 2012 F: 一般化ポーカー
  - BNF はあるけれどあまり構文解析らしくないかも

```
hand_pattern = card_pattern_1 ' ' card_pattern_2 ' ' ... ' ' card_pattern_L
card_pattern = '*' | var_plus
var_plus = variable | var_plus '+'
variable = 'a' | 'b' | 'c'
```

- 2007 F:部陪博士，あるいは，われわれはいかにして左右非対称になったか
  - 構文解析部分がエッセンスのみ ⇒ 初めての人におすすめ



構文解析なんて不要?

- 競技ルールを見てみよう

> 6. プログラム自動生成ツールの使用禁止
> yacc や lex といったプログラムの自動生成が可能なツールの使用は禁止します。

字句解析器ジェネレータ

構文解析器ジェネレータ

- 構文解析問題に便利な自動化ツールの使用が禁止されている！
  - 使わないなら禁止しない？
  - 使うかも？

ゲームパート

minimax 法

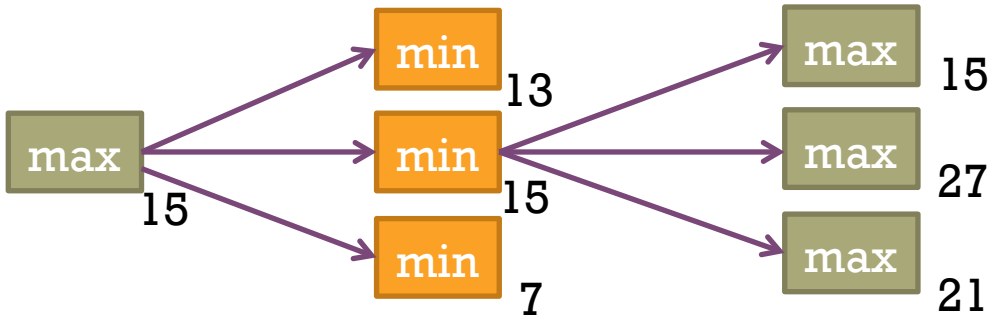
minimax 法

- 2人ゲームで最善を尽くすとは:
  1. 先読みをする
  2. その上で最良の結果に辿り着ける選択をする

- 具体的には:
  - 各選択肢について最良の結果が既知のとき
  - それらの結果の中で最良の選択肢を選ぶはず

  ⇒ 各盤面で最良の結果は一意

  ⇒ 最後の手から逆順に帰納的に最良が求まる



計算量は大丈夫？

- すごく大雑把に見積もって
  - 挿入: 16種の記号 × 最大 11+6+1 箇所
  - 削除: 最大 11+6 箇所
  - 11ターン
  - $(16\times18+17)^{11}\fallingdotseq2\times10^{27}$通り?
- 実は9埋めの結果が最大
  - メモ化して$11\times2\times10^{18}$通り?
  - 実際にはもっと少ないはず
  - とは言っても時間的にも空間的にも無理そう…

こういうときは？

- 机上で唸っていても仕方がない！

■実際に遊ぼう！

⇒ 何か戦略が見つかるかも

キャンセルの利用

- このゲームでは相手の操作をキャンセルする方法がある
  - 任意の1文字を削除できる
  - 以前の状態なので有効であることは保証されている
- 実はキャンセルって強いのでは？
  - 少なくとも現状が維持できる
    - 不利なことをされたら元に戻してしまえばいい

結局どうなるの？

- 先手:
  1. 有利な手はどうせキャンセルされるので、
     不利にならない程度に適当にやる
  2. 最後のターンだけ全力を出す
- 後手:
  1. 有利な手はどうせキャンセルされるので、
     とりあえずキャンセルして相手の手を潰し続ける
  2. 総ターン数が偶数のときは、最後のターンで好きな手が打てる！
     - もう相手の番は来ない
       - キャンセルされることを気にしなくてよい！
       - したくなかったらキャンセルしなくてもよい！

結論

- 総ターン数が奇数のとき:
  - 1ターンだけのときと同じ
    - 残りはキャンセルしつづける
- 総ターン数が偶数のとき:
  - 2ターンだけのときと同じ
    - 最後のターンを除きキャンセルし続ける

0ターンではない!

2チームがここでミス!

結果発表

結果

- 解いたチーム: 3 チーム
  - MadokaMAgica [非現役]: 55分
  - watashi [外国]: 162分
  - ~shiokawa: 166分
- その他2チームが挑戦したものの - 前述のように誤答でした
  - 仮説を立てたら必ず原理を検証しよう
  - 浅知恵で特攻するのはやめましょう
- E 問題にしては少々難しかったようです
  - 普通は10〜20チーム解きます
  - 3チーム以上進出する大学の方は本番では確実に解きましょう