Problem Solving, Presenting, and Programming

Tom Verhoeff
Dept. Math. & Computer Science

To be presented at IOI 2016

TU/e Technische Universiteit Eindhoven University of Technology

Where innovation starts

Some Background

- **Involved in IOI from 1994 until 2007**
  - **IS: website with historic data**
  - **ISC: co-founder, chair**
  - **IOI Syllabus, co-author, editor**

- **Teach math enrichment classes in primary school**
  - **Weekly**
  - **Focus on problem solving**
  - **Classical problems + exotic/newer problems**
    - **For references, see the article**

Nim

- **Classical two-player game**

- **State: Several piles of items**
- **Move: Take *one or more* items from *one* pile**
- **End: Whoever takes last item loses**

Mathematical Analysis

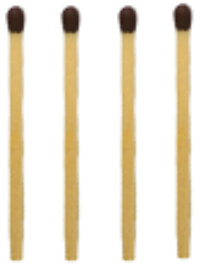
- **Known as *impartial* game**
  - **Both players can make the same kind of moves**

- **Algorithm for perfect play is known**
  - **Involves *binary notation* and *nim sum* (xor)**
  - **Harder to teach in primary school**
  - **So, look for simplification**

Nim Sum

- ≤ 1 pile with > 1 item: “easy” to win

- 3 = 011
- 4 = 100
- 5 = 101



- $\oplus$ 010 (nim sum)
- Lost if nim sum = 0
- Else: take from pile k with k $\oplus$ nim sum < k
  - and leave k $\oplus$ nim items

Simplified Algorithm, Part 1

- **For each pile**
  1. **Break it down into groups of size 1**
  2. **Repeatedly merge two groups of equal size**
  3. **This terminates when all group sizes differ**

- **19 = *//// //// //// //// + // + /*** **5 = *//// + /*** **3 = *// + /***

- **Find *largest* group size G with *odd* occurrence count**

- **If all even: lost**

Simplified Algorithm, Part 2

- **Take 1 from any max-size-G group, say in pile P**
- **Split remainder of that group**
- **18 = *//// //// + //// + // + / + // + /* 5 = *//// + /* 3 = *// + /***
- **Continue taking from pile P such that *all* group sizes have *even* occurrence count**
- **6 = *//// + //*** 5 = *//// + /* 3 = *// + /***

Programming

- Not so convenient in imperative language
- Easier in functional language with patterns
- Wolfram Language
  - Mathematica
  - Free on Raspberry Pi, Intel Edison
  - Free in the Wolfram Cloud

- www.wolfram.com/language

Programming in Wolfram Language

- singletons[n_Integer] := Table[{1}, n]

- combine[{x___, a_, y___, a_, z___}] := {x, Join[a, a], y, z}
- combine[list_List] := list
- combineStar[list_List] := FixedPoint[combine, list]

- split[n_Integer] := combineStar[singletons[n]]
- split[position_List] := Map[split, position]

- split[{3, 4, 5}] -> {{{1, 1}, {1}}, {{1, 1, 1, 1}}, {{1, 1, 1, 1}, {1}}}

Conclusion

- **Presenting a solution can be a problem in itself**
- **Simple and insightful algorithm for Nim**
- **Exercise:**
  - **Find algorithm when taking from *one or two* piles**

Questions?