Efficient Generation of the Ideals of a Poset

Lalla Mouatadid
Advisor: Dr. Gara Pruesse
Vancouver Island University

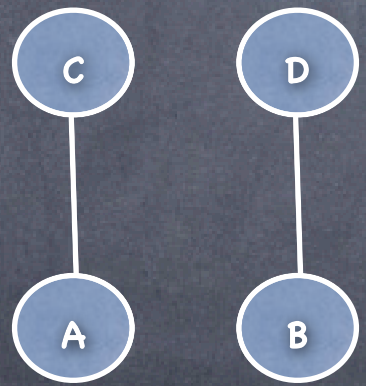
MathFest 2009

Overview

- Definition & Background Information
- Problem & Algorithm
- Open Problems
- Questions

Definitions

- **A partial order set** (poset) denoted P(E, ≤) is the ground set E of elements where ≤ is the order relation between elements of E.

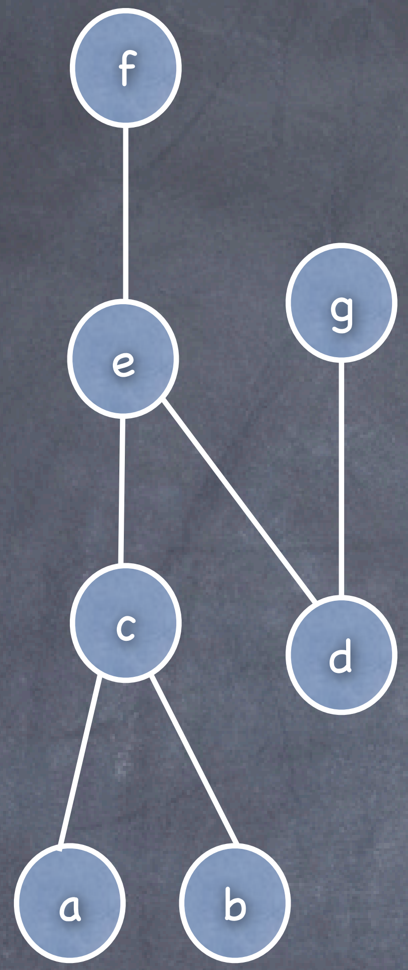


E = {A, B, C, D}

- A ≤ C
- B ≤ D
- A and B are incomparable

Definitions

- **An Ideal** of a poset is a subset $I \subset E$, such that if $x \in I$ and $y \leq x$ in P, then $y \in I$.



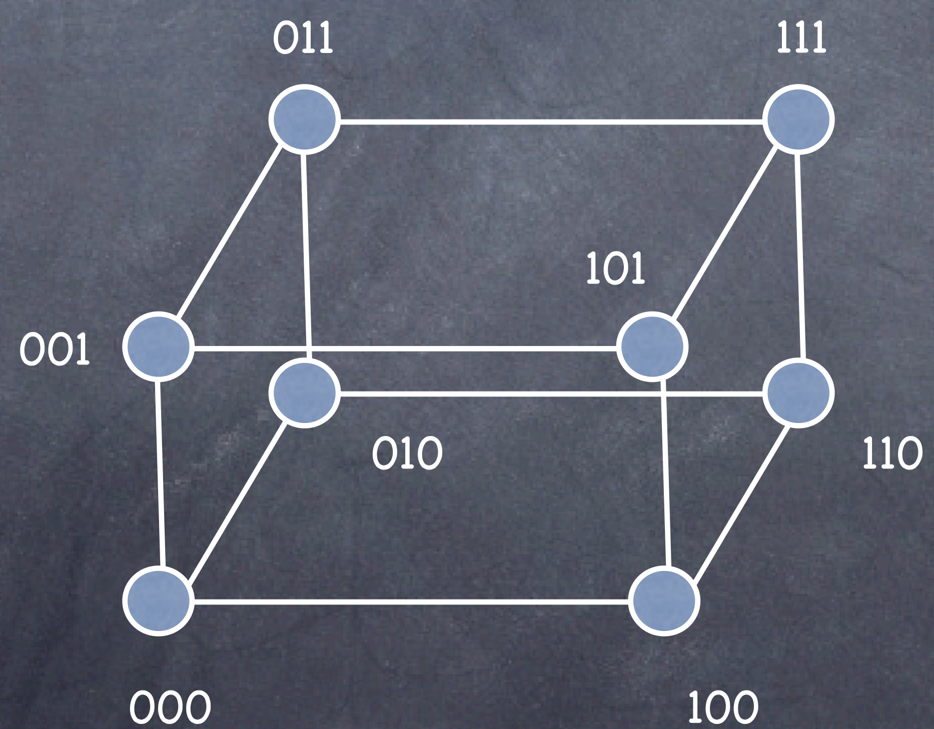
- $P(E, \leq)$
- $E = \{a, b, c, \ldots, g\}$
- $I = \{a, b, c\}$
- $(c \in I)$ and $(a \leq c)$ $\Rightarrow a \in I$
- $I' = \{a, c\}$ is not an ideal because $b \leq c$, but $b \notin I'$

Ideals of $P(E, \leq)$:

$\{\varnothing\}$, $\{a\}$, $\{b\}$, $\{d\}$, $\{a, b\}$, $\{a, d\}$, $\{b, d\}$, $\{a, b, d\}$, $\{a, b, c\}$, $\{a, b, c, d\}$, $\{a, b, d, g\}$, $\{a, b, c, d, e\}$ $\{a, b, c, d, e, g\}$, $\{a, b, c, d, e, f\}$, $\{a, b, c, d, e, f, g\}$

Definitions

- **A Gray Code** is a listing of all instances of combinatorial objects such that successive instances differ in only one bit
- Example: For a binary system, the listing of all bit strings in a REFLECTED gray code manner, where n=3 is the number of bits, is: 000, 001, 011, 010, 110, 111, 101, 100



0 00
0 01
0 11
0 10
------
1 10
1 11
1 01
1 00

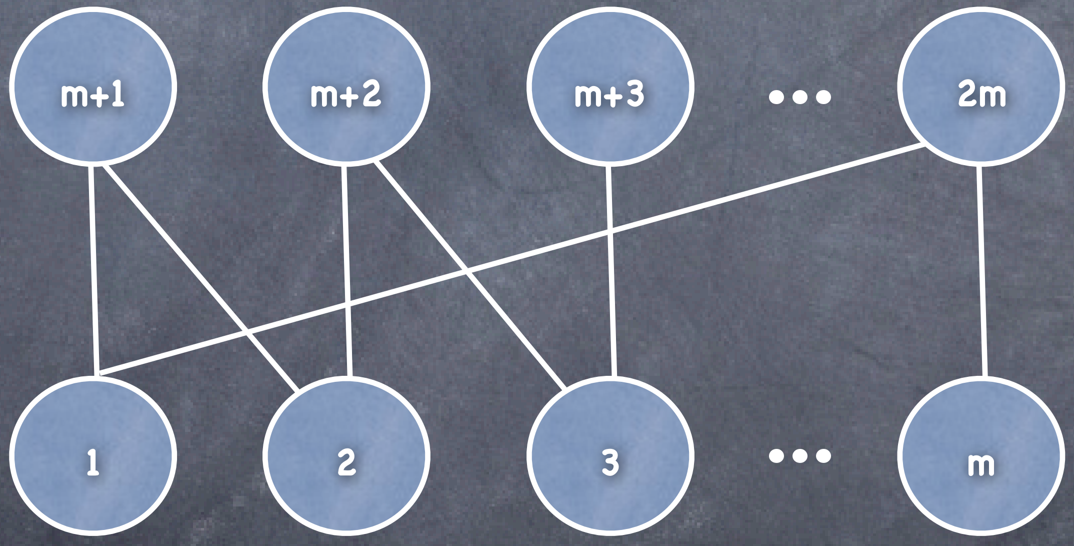
Posets & Ideals

- Ultimate Goal: Generate ideals efficiently and in a gray code manner
  - Most efficient algorithm runs in O(logn) but is not a gray code. [Squire 1995]
  - Pruesse and Ruskey's algorithm (1993) is a gray code but takes O(n) in the worst case
- Our Focus: Crown posets (a class of N posets)
- Our goal is to list all the ideals of a poset in a gray code manner in constant amortized time.

Crown Posets

- A crown poset is a poset with elements $\{1, 2, \ldots, 2m\}$ where $m \geq 2$ and in which

  $i < (m+i)$, $(i+1)<(m+i)$ for each $i = 1, 2, \ldots, (m-1)$, and $1 < 2m$, and $m < 2m$.



Crown Posets

- The Lucas numbers L_n are defined as follow:
  $L(n) = L(n-1) + L(n-2)$ for $n > 1$, where $L(0) = 2$ and $L(1) = 1$

  2, 1, 3, 4, 7, 11, 18, 29, 47, 76, 123 …

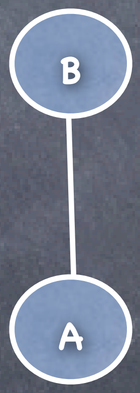
- The number of ideals of a crown poset $P(E, \leq)$ are counted by the even Lucas numbers L_{2n} where $|E| = 2n$

Example:

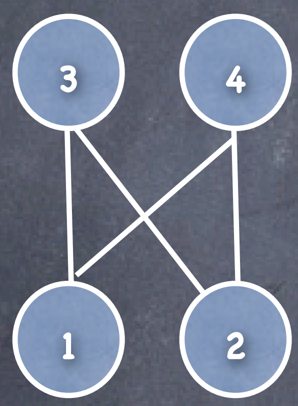
- For $n = 1$

$I = \{\varnothing\}, \{A\}, \{A, B\}$

$L(2n) = L(2) = 3$



Lattice Graph



E = {1, 2, 3, 4}

Ideals:

{∅}

{1}

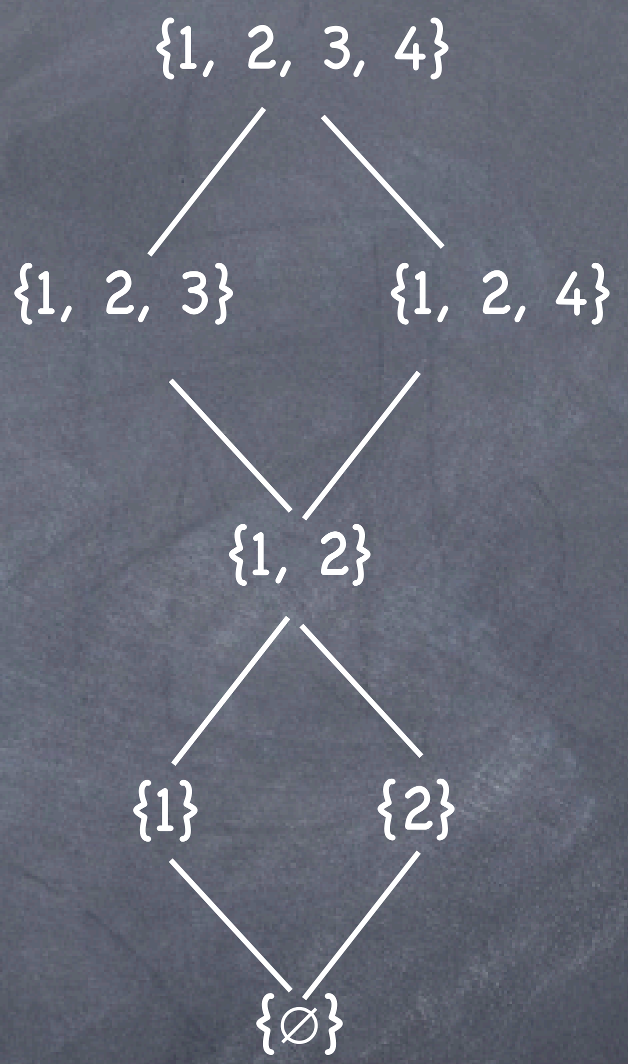
{2}

{1, 2}

{1, 2, 3}

{1, 2, 4}

{1, 2, 3, 4}

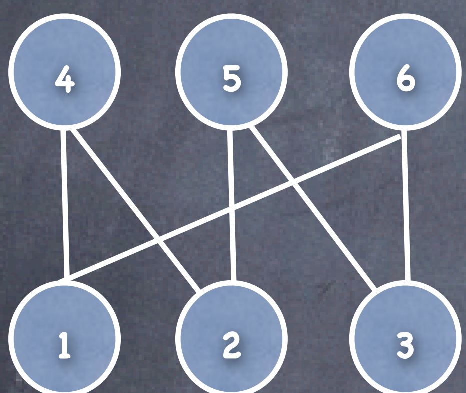


Algorithm

- Let P(E, ≤) be a crown poset where
|E| = 2n
- array_ideals[L(2n)]
- m = # of possible bit strings of the minima = 2^n
- array_bitstr[n]

```
loop(m times){
    0. Total parity = 0
    1. Set prefix parity
    2. loop(n times){
        2.0 Update prefix parity
        2.1 Find bit to flip (i.e 1st bit with same parity as the prefix)
        2.2 Update array_bitstr
        2.3 Check bit's neighbors
        2.4 Update array_ideals
        2.5 List all subsets of maxima
        2.6 Update total parity
        2.7 Flip last bit changed in ideals[] and restore array_bitstr
    }
}
```

Example



minima = {1, 2, 3}

maxima = {4, 5, 6}

Bit strings => Feasible set

$00\underline{0} \Rightarrow \{\varnothing\}$

$0\underline{0}1 \Rightarrow \{3\}$

$01\underline{1} \Rightarrow \{2, 3\}$

$010 \Rightarrow \{2\}$

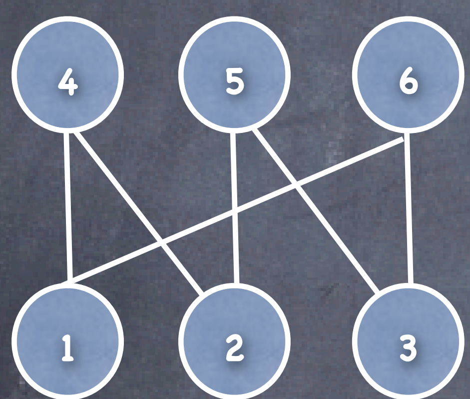
$11\underline{0} \Rightarrow \{1, 2\}$

$1\underline{1}1 \Rightarrow \{1, 2, 3\}$

$10\underline{1} \Rightarrow \{1, 3\}$

$100 \Rightarrow \{1\}$

Example



minima = {1, 2, 3}
maxima = {4, 5, 6}
Bit strings:

000 ⇒ {∅}

001 ⇒ {3}

011 ⇒ {2, 3}

⇒ {2, 3, 5}

010 ⇒ {2}

110 ⇒ {1, 2}

⇒ {1, 2 , 4}

111 ⇒ {1, 2, 3}

⇒ {1, 2, 3, 4}

⇒ {1, 2, 3, 4, 5}

⇒ {1, 2, 3, 4, 5, 6}

⇒ {1, 2, 3, 4, 6}

⇒ {1, 2, 3, 6}

⇒ {1, 2, 3, 5, 6}

⇒ {1, 2, 3, 5}

101 ⇒ {1, 3}

⇒ {1, 3, 6}

100 ⇒ {1}

Analysis

- The minima bit strings are generated in a binary reflected gray code n-bit strings

- We only add or remove one element at each step to get the next bit string (i.e O(1))

- The prefix parity to updated at each step

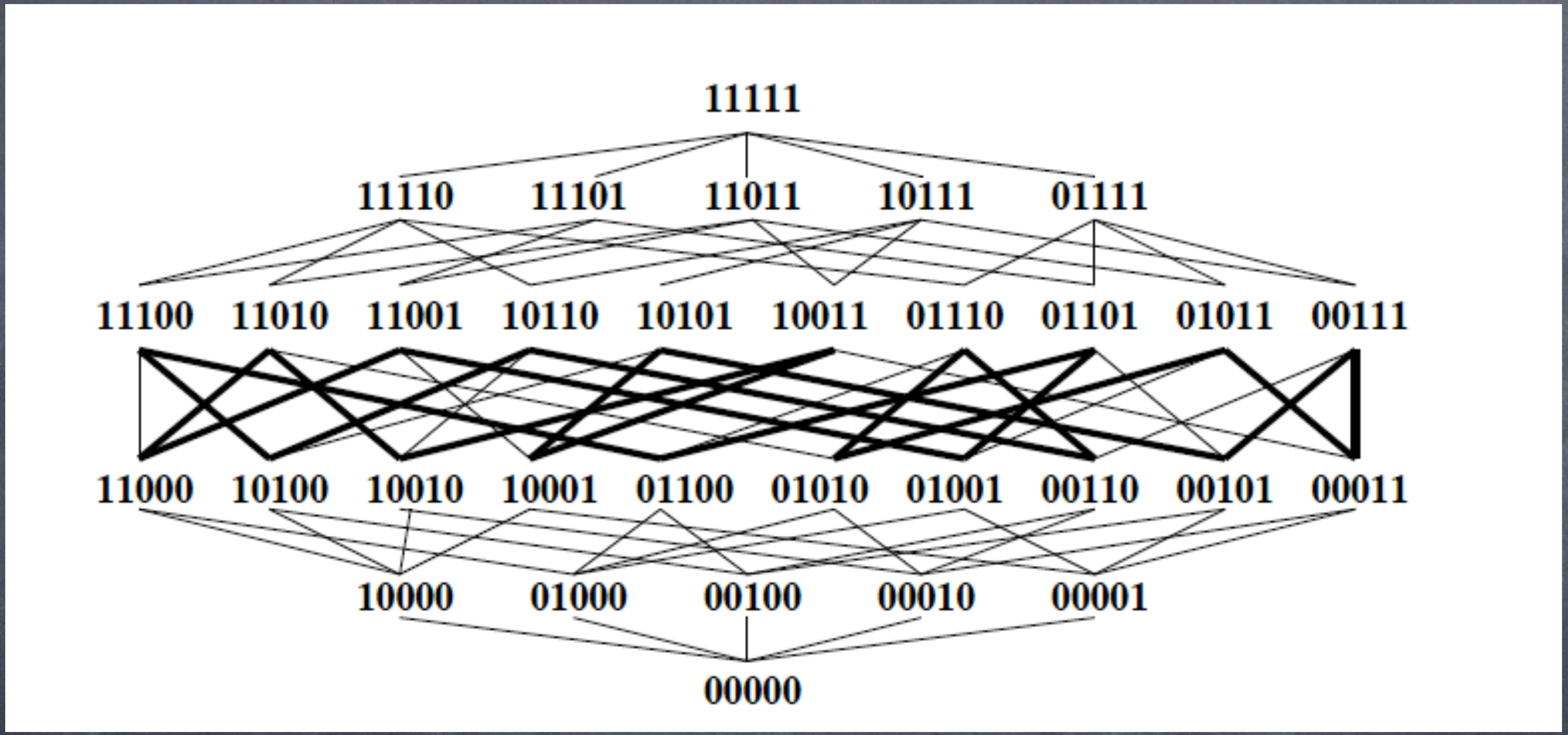
- The feasible maxima bit strings are generated in the same gray code as the minima's

- At each step, one of the following happens:
  1. Add or delete a maximal element
  If removing last maximal:
  2. Add or remove minimal element

$\Rightarrow$ 2 changes happen at the most.

Open problems

- Main problem: Generate ideals of ANY poset in constant amortized time.
- The Middle Levels problem:
  - For the hypercube of order n (where n = 2k+1), determine if there is a Hamilton cycle in the middle levels k and k+1 of the lattice graph



Hypercube of order n = 5

Special thank you to

- Dr. Gara Pruesse
- Vancouver Island University
- MAA
- Alycia Kolat

Any questions?

Thank you