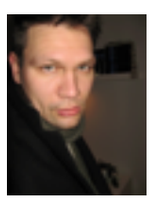
Fast zeta transforms for lattices with few irreducibles

Petteri Kaski
Aalto University & HIIT

ALGODAN Seminar — 28 October 2011

This is joint work with Andreas Björklund (Lund),
Thore Husfeldt (Copenhagen),
Mikko Koivisto (Helsinki),
Jesper Nederlof (Bergen) &
Pekka Parviainen (Helsinki)



- Algorithmic **F**oundations of data analysis
- Observation:
  At the core of many data analysis tasks, there is a computational problem with strong fundamental motivation
- **F**un interactions between
  1) fundamentals & data analysis
  2) combinatorics & algebra

Example: Cross-correlation between two time series

- Let a(t) and b(t) be time series with support in t = 0,1,2, ..., N
- Cross-correlation with delay d = 0, 1, 2, ..., N:

$$\varrho(d) = \left(\sum_{t} a(t)b(t-d) - \mu(a)\mu(b) \right) / \sigma(a)\sigma(b)$$

- *Task*: Given a and b as input,
  compute $\varrho(d)$ *for each* d

The computational bottleneck in cross-correlation

- *Task*:
  Given a(t) and b(t) with support in t = 0,1,2,...,N,
  compute $\sum_t a(t)b(t-d)$ for each d = 0,1,2,...,N

- A direct implementation:
  $O(N^2)$ time

- But can we go faster?

Rephrasing the task:
... *polynomial multiplication* in disguise!

- Given a(t) and b(t) with support in t = 0,1,2,..., N, compute $\sum_t$ a(t)b(t – d) for each d = 0,1,2,....,N

- Set b'(t) = b(N – t):
  Given a(t) and b'(t) with support in t = 0,1,2,..., 2N, compute $\sum_t$ a(t)b'(d – t) for each d = N,N+1,...,2N

- Rephrased:
  Given two polynomials of degree at most 2N, compute their product

Example: Polynomial multiplication

- Suppose p and q are polynomials of degree D
- Elementary multiplication algorithm requires $O(D^2)$ time
- Can we do better?

Mathematics of Computation 19 (1965) 297–301

An Algorithm for the Machine Calculation of Complex Fourier Series

By James W. Cooley and John W. Tukey

An efficient method for the calculation of the interactions of a 2^m factorial experiment was introduced by Yates and is widely known by his name. The generalization to 3^m was given by Box et al. [1]. Good [2] generalized these methods and gave elegant algorithms for which one class of applications is the calculation of Fourier series. In their full generality, Good's methods are applicable to certain problems in which one must multiply an N-vector by an $N \times N$ matrix which can be factored into m sparse matrices, where m is proportional to $\log N$. This results in a procedure requiring a number of operations proportional to $N \log N$ rather than N^2. These methods are applied here to the calculation of complex Fourier series. They are useful in situations where the number of data points is, or can be chosen to be, a highly composite number. The algorithm is here derived and presented in a rather different form. Attention is given to the choice of N. It is also shown how special advantage can be obtained in the use of a binary computer with $N = 2^m$ and how the entire calculation can be performed within the array of N data storage locations used for the given Fourier coefficients.

Consider the problem of calculating the complex Fourier series

$$X(j) = \sum_{k=0}^{N-1} A(k) \cdot W^{jk}, \qquad j = 0, 1, \cdots, N-1, \tag{1}$$

where the given Fourier coefficients $A(k)$ are complex [illegible] principal Nth root of unity,

$$W = e^{2\pi i/N}. \tag{2}$$

A straightforward calculation using (1) would require N^2 operations wh[illegible] tion" means, as it will throughout this note, a complex multiplication foll[illegible] complex addition.

The algorithm described here iterates on the array of given complex Fourier amplitudes and yields the result in less than $2N \log_2 N$ operations without requiring more data storage than is required for the given array A. To derive the algorithm

Via the FFT: O(D log D) time

So, cross-correlation
and polynomial
multiplication are essentially
the same thing

(this is, of course,
well known)

What about objects that are *analogous* to polynomials?

Polynomial multiplication revisited

- A polynomial such as $1x^0+4x+2x^2+3x^3$ is a "bag" of monomials
- Each monomial is associated with a scalar
- Multiplication:
  Cross-multiply monomials in both "bags" and collect

$$(1x^0+1x^1+3x^2)\cdot(1x^0+2x^1)$$
$$= 1x^0+1x^1+3x^2+2x^1+2x^2+6x^3$$
$$= 1x^0+3x^1+5x^2+6x^3$$

Multiplying two "bags" of "objects"

- Objects: Elements of a semigroup $(S, \cdot)$
- Each object is associated with a scalar
- Multiplication:
  Cross-multiply with "$\cdot$" and collect

$$(1a + 2b) \cdot (1c + 3d)$$
$$= 1a \cdot c + 2b \cdot c + 3a \cdot d + 6b \cdot d$$
$$= 1c + 2d + 3d + 6a$$
$$= 6a + 1c + 5d$$

Examples

- Monomials & sum of degrees

  $(1x^0+1x^1+3x^2) \cdot (1x^0+2x^1) = 1x^0+3x^1+5x^2+6x^3$

- Permutations & function composition

  $(1(a,b,c) + 2(a)(b,c)) \circ (1(a,c)(b) + 1(a)(b,c)) =$

  $= 2(a)(b)(c) + 1(a)(b,c) + 2(a,b,c) + 1(a,b)$

- Sets & union

  $(1\{a,b\} + 3\{c,d\}) \cup (1\{b,c\} + 2\{d\}) =$

  $= 1\{a,b,c\} + 3\{b,c,d\} + 2\{a,b,d\} + 6\{c,d\}$

In data analysis ...

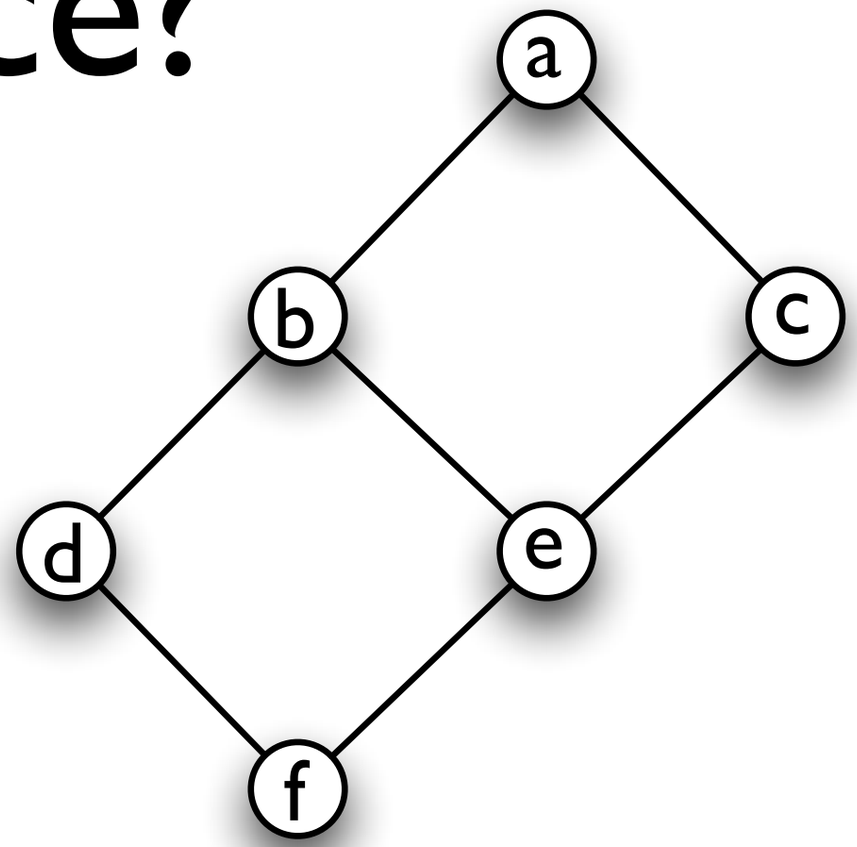
- Polynomials:
  Time series analysis, e.g. auto- and cross-correlation, spectral analysis of periodic time series via the FFT

- Permutations:
  Analysis of ranked data, e.g. ranked ballots via group-theoretic techniques [Diaconis]

- Sets:
  E.g. market-basket data, frequent itemsets, partitioning/clustering, ...
  ... in the **lattice** of subsets of an n-element set

What is a lattice?

- *Combinatorial definition:*
  A (finite) partially ordered set $(L, \leq)$ such that
  1) there is a unique minimum element; and
  2) any two elements $x,y \in L$ have a least upper bound (**join**) $x \vee y$

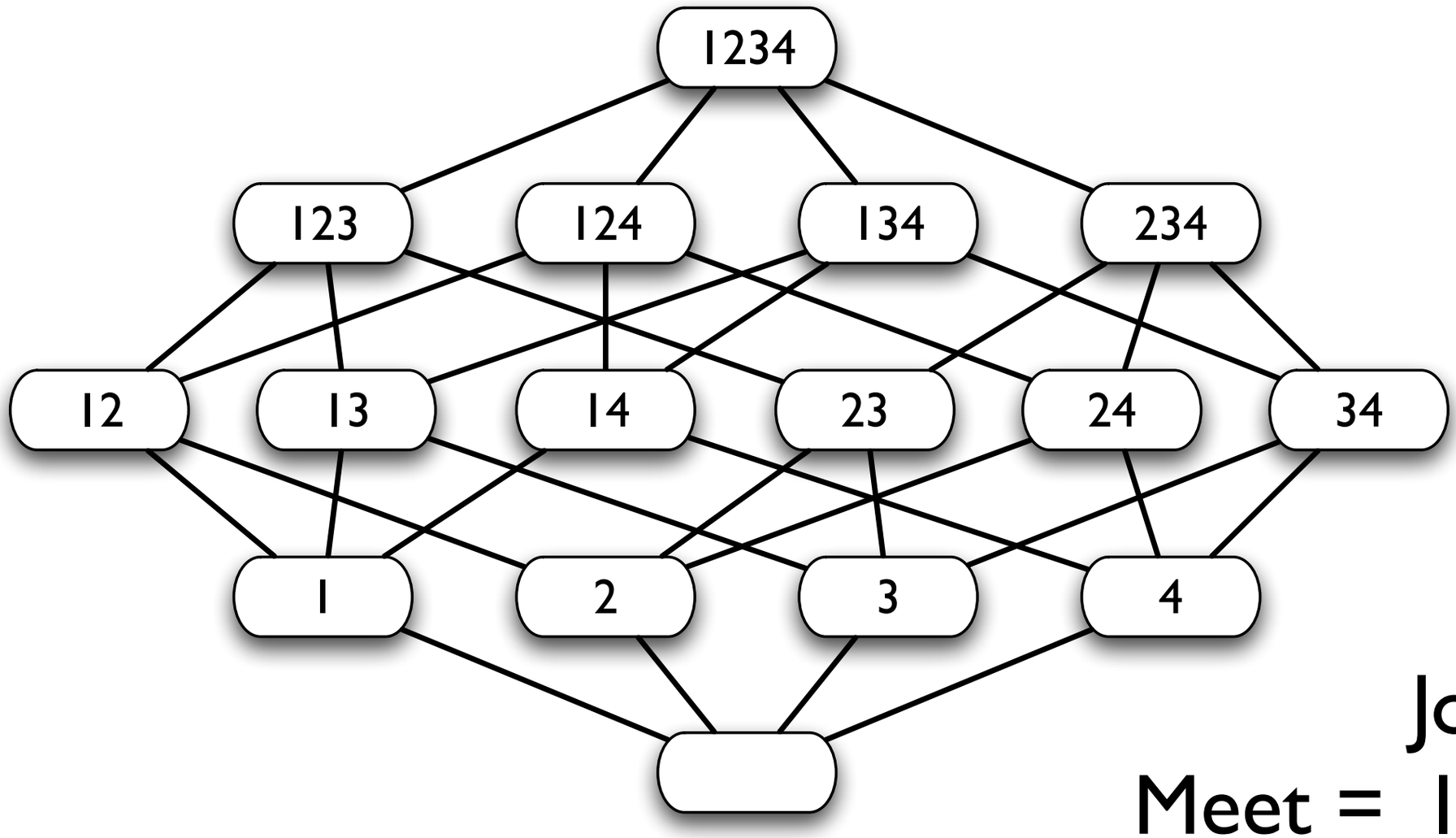


- *Algebraic definition:*
  A (finite) commutative idempotent semigroup $(L, \vee)$ with identity

$\vee$	a	b	c	d	e	f
a	a	a	a	a	a	a
b	a	b	a	b	b	b
c	a	a	c	a	c	c
d	a	b	a	d	b	d
e	a	b	c	b	e	e
f	a	b	c	d	e	f

Example: Subset lattice

- The set of all subsets of an n-element set
- Partially ordered by subset inclusion

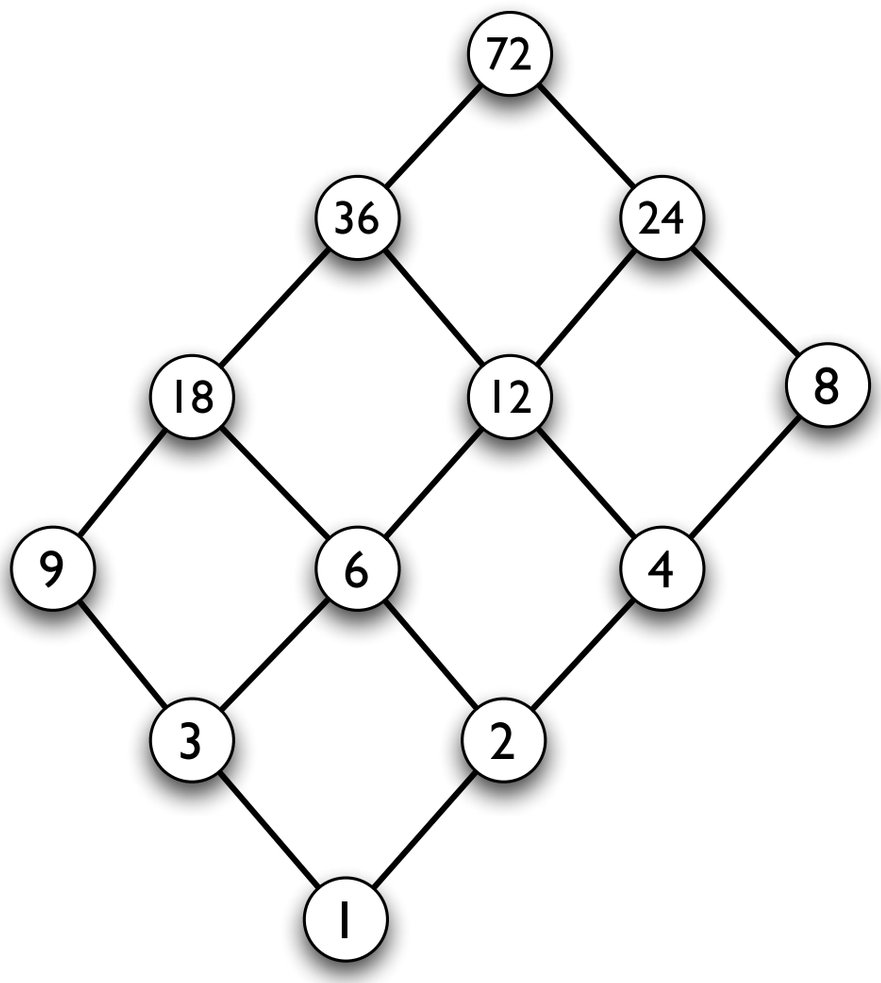


Join = Union

Meet = Intersection

Example: Divisor lattice

- The set of all positive divisors of a positive integer n
- Partially ordered by divisibility



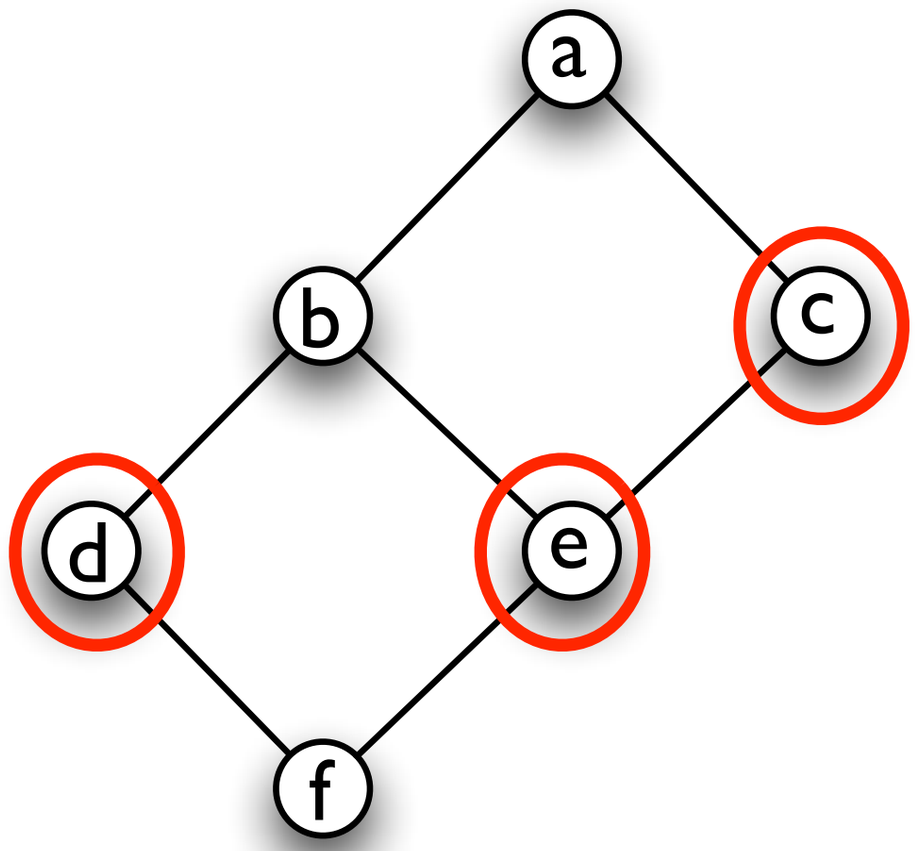
Join = lcm
Meet = gcd

Can we multiply
"polynomials"
of lattice elements *fast*,
for *arbitrary lattices* ?

Present work: Yes!
(SODA12, to appear)

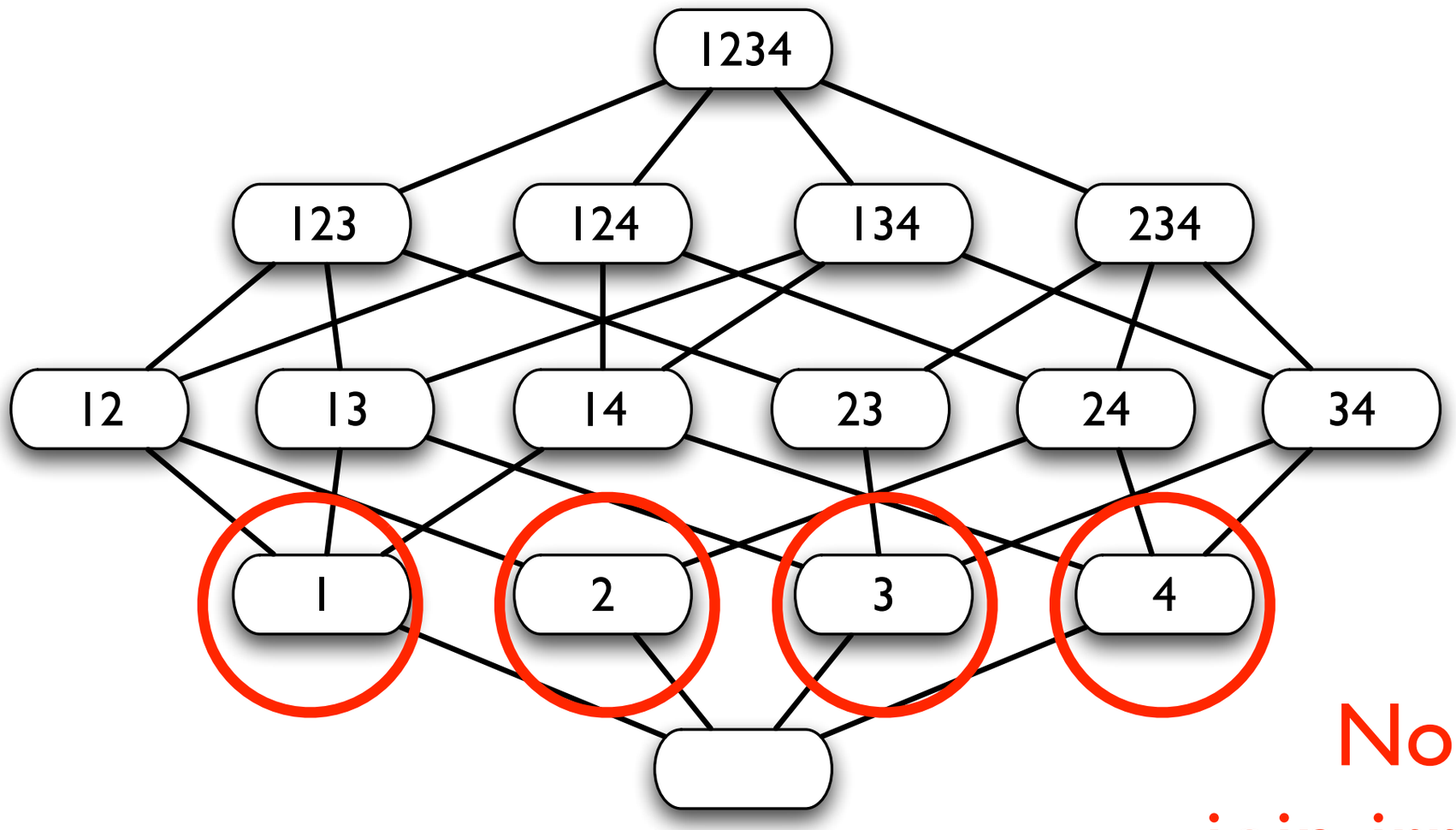
Join-irreducible elements

- Let $(L, \leq)$ be a lattice
- An element z is **join-irreducible**
  if $z = x \vee y$ implies $z = x$ or $z = y$
- The minimum ("zero") element is always join-irreducible
- *Algebraic view*:
  The set of nonzero join-irreducibles is a minimal set of generators for $(L, \vee)$



Example: Subset lattice

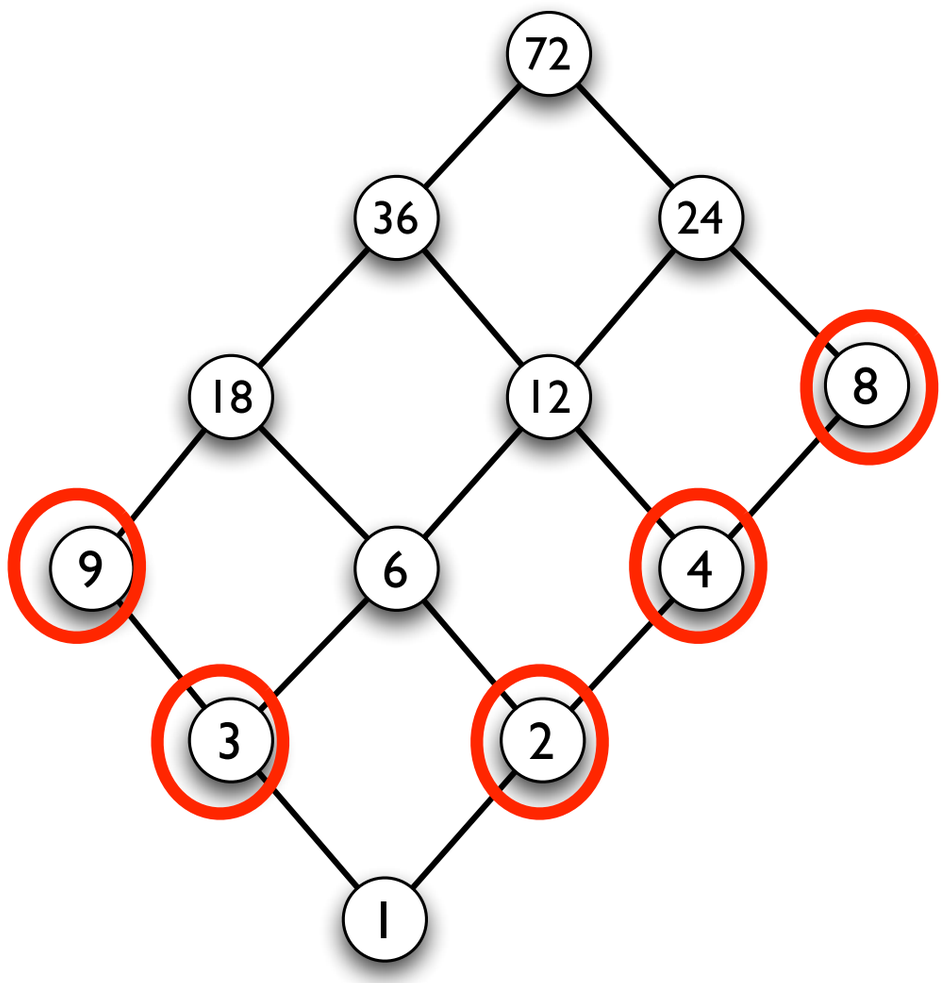
- The set of all subsets of an n-element set
- Partially ordered by subset inclusion



Nonzero
join-irreducibles

Example: Divisor lattice

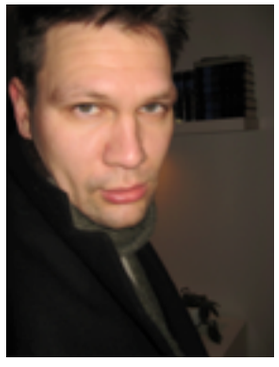
- The set of all positive divisors of a positive integer n
- Partially ordered by divisibility



Nonzero join-irreducibles

Theorem [BHKKNP]



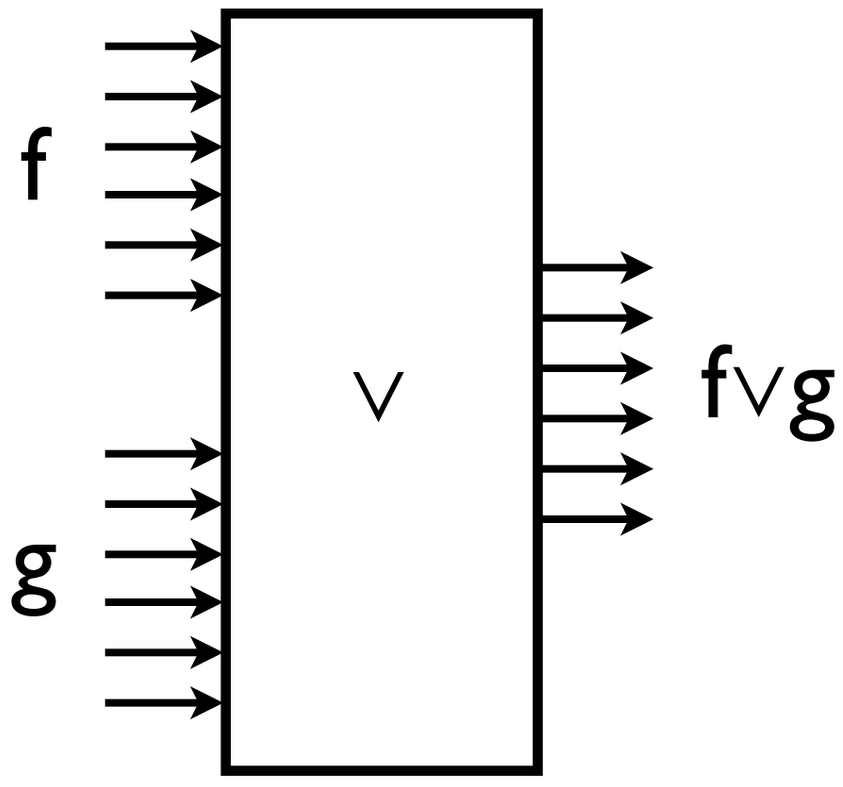



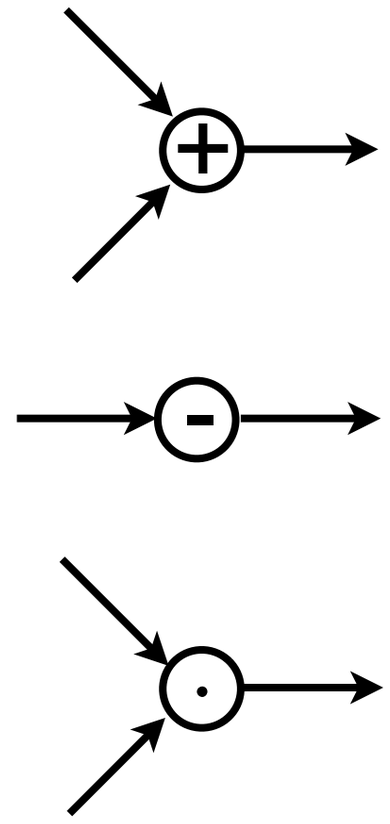
- Let $(L, \leq)$ be a lattice with v elements, n of which are nonzero and join-irreducible
- Then, there exist arithmetic circuits of size $O(vn)$ both for the zeta transform on L and for the Möbius transform on L

(Analogs of the FFT)

Corollary

- "Polynomials" of elements of L can be multiplied with an arithmetic circuit of size O(vn)
- *Cf.* cross-multiplication takes O(v^2) gates





Further work & applications

- Can we go faster?
- What about other semigroups?
- Applications in analysing lattice/semigroup-valued data—e.g. partially ranked data, partitions, ordered partitions, ...
- Analysis of mixing time of Markov chains on semigroups [Bidigare, Hanlon & Rockmore; Brown; Brown & Diaconis]
- Fast Fourier transforms on non-group semigroups, in particular the *rook monoid* [Malandro & Rockmore; Malandro]