Easy Counting and Ranking for Simple Loops

Alain KETTERLIN

Inria / CAMUS   CNRS / ICUBE   Université de Strasbourg / ICPS

IMPACT 2024: January, 17, 2024

Problem Statement

Integer Polynomials

Algorithms

Conclusion

Quantitative aspects of loops

```
for i = 0 to n
    S1
    for j = 0 to i
        S2
    S3
    for j = i+1 to n
        S4
        for k = 0 to i
            S5
        S6
```

(cholesky from PolyBench v3)

- **Counting**: how many instruction executions overall? $\left(\frac{5n+6n^2+n^3}{6}\right)$
- **Ranking**: how many instructions before S5(i,j,k)? $\left(\frac{6+(9n-7)i+(3n-9)i^2-2i^3+6(i+2)j+6k}{6}\right)$
- **Unranking**: what is the instruction with rank r?

Existing solutions in the polyhedral framework

- Clauss/Ehrhart: polynomials with periodic coefficients
- Verdoolaege/Barvinok: step-polynomials (in integer parts)

Pros

- extremely general/powerful solutions
- work at the polyhedral level

Cons

- unconventional form of the result
- work at the polyhedral level

Loops to sums to polynomials

```
for i = 0 to n
    S1
    for j = 0 to i
        S2
    S3
    for j = i+1 to n
        S4
        for k = 0 to i
            S5
        S6
```

$$\Rightarrow \sum_{i=0}^{n-1}\Big(1 + \sum_{j=0}^{i-1}(1) + 1 + \sum_{j=i+1}^{n-1}\big(1 + \sum_{k=0}^{i-1}(1) + 1\big)\Big) = \left(\frac{5n+6n^2+n^3}{6}\right)$$

$$\sum_{j=i+1}^{n-1}\big(1 + \sum_{k=0}^{i-1}(1) + 1\big) = 2n-2+(n-3)i-i^2$$

Mechanical translation + algebra:

→ when is this possible/correct?

→ how to implement this?

Loop syntax

- parameters, with affine inequalities
- arbitrary nesting of loops with affine bounds
- no `min`/`max`, no `mod` or integer parts

Loop semantics

$$\texttt{for } i\texttt{=}l \texttt{ to } u \texttt{ . . .} \qquad \Rightarrow \qquad \sum_{i=l}^{u-1} \cdots$$

This only makes sense for *simple* loops, with:

1. *unit step*: loop counter incremented by 1
2. *bounds coherence*: $l \leq u$ for every instance of the loop

The latter needs explicit verification

Representing integer polynomials

$$\sum_{i=0}^{n} a_i x^i$$ a ring for the coefficients + a monomial basis

Correctness: all polynomials are integer-valued

Completeness: all integer sequences can be represented

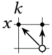
The problem with integer polynomials

	Correct	Complete	In practice
$a_i \in \mathbb{Z}$	✓	$\frac{x(x-1)}{2}$	essentially unusable
$a_i \in \mathbb{Q}$	$\frac{x(x-2)}{2}$	✓	correctness proofs

$$x^{\underline{k}|} \triangleq \binom{x}{k} = \frac{x \cdot (x-1)\cdots(x-k+1)}{k!} = \frac{x!}{k!(x-k)!} = \frac{x^{\underline{k}}}{k!}$$

- $x^{\underline{k}|}$ appears in Pascal's triangle, with

$$(x+1)^{\underline{k+1}|} = x^{\underline{k}|} + x^{\underline{k+1}|}$$



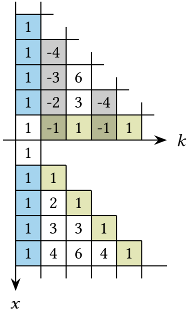
- defined for all $x \in \mathbb{Z}$, $k \in \mathbb{Z}_{\geq 0}$

$$(-x)^{\underline{k}|} = (-1)^k (x+k-1)^{\underline{k}|}$$

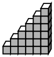
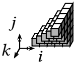
- relation between the various powers:

$$x^{\underline{n}|} = \frac{1}{n!}\sum_{k=0}^{n}(-1)^{n-k}\begin{bmatrix} n \\ k \end{bmatrix} x^k \qquad x^n = \sum_{k=0}^{n} k! \begin{Bmatrix} n \\ k \end{Bmatrix} x^{\underline{k}|}$$

with $\left[{n \atop k}\right]$ and $\left\{{n \atop k}\right\}$ the unsigned Stirling numbers



- triangles are to $x^{\underline{k|}}$ what squares are to x^k

$x^{\underline{0|}} =$  $x^{\underline{1|}} =$  $x^{\underline{2|}} =$  $x^{\underline{3|}} =$ 

$\dots$

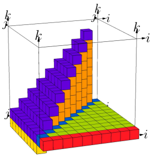
- triangular sums and loops

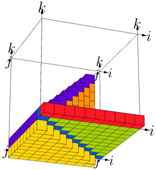
$$\sum_{i_1=0}^{n-1} \sum_{i_2=0}^{i_1-1} \cdots \sum_{i_d=0}^{i_{d-1}-1} 1 = n^{\underline{d|}}$$

for i=0 to n	$n^{\underline{3|}}$
for j=0 to i	$i^{\underline{2|}}$
for k=0 to j	$j^{\underline{1|}}$
S	1

- the Cholesky iteration domain

```
for i = 0 to n
    S1
    for j = 0 to i
        S2
    S3
    for j = i + 1 to n
        S4
        for k = 0 to i
            S5
        S6
```





Polynomials = integer coefficients and binomial powers

$$\sum_{i=0}^{n} a_i x^{\underline{i}} \qquad \text{with } a_i \in \mathbb{Z}$$

this representation is correct and complete

Completeness via interpolation

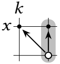
For *any* sequence of integers $v_0, \ldots, v_n$,
there is a unique *interpolating* polynomial $p(x) = \sum_{i=0}^{n} a_i \cdot x^{\underline{i}}$

$$\begin{aligned} v_0 &= p(0) = a_0 + a_1 \cdot 0^{\underline{1}} + a_2 \cdot 0^{\underline{2}} + a_3 \cdot 0^{\underline{3}} \cdots \\ v_1 &= p(1) = a_0 + a_1 \cdot 1^{\underline{1}} + a_2 \cdot 1^{\underline{2}} + a_3 \cdot 1^{\underline{3}} \cdots \\ v_2 &= p(2) = a_0 + a_1 \cdot 2^{\underline{1}} + a_2 \cdot 2^{\underline{2}} + a_3 \cdot 2^{\underline{3}} \cdots \\ &\cdots \end{aligned} \quad \Rightarrow \quad a_i = \sum_{j=0}^{i} (-1)^{\underline{i-j}} \cdot i^{\underline{j}} \cdot v_j$$

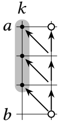
Implementation note: no need for rational numbers

Variation and Summation

$$\Delta x^{\underline{k+1}} = (x+1)^{\underline{k+1}} - x^{\underline{k+1}} = x^{\underline{k}}$$



$$\sum_{x=a}^{b-1} x^{\underline{k}} = x^{\underline{k+1}}\Big|_a^b \qquad (a \le b)$$
$$= b^{\underline{k+1}} - a^{\underline{k+1}}$$



Discrete calculus terminology

finite difference $\quad \Delta f(x) = f(x+1) - f(x)$

anti-difference $\quad \Delta^{-1} f(x) = \sum f(x)$

$$\Delta \begin{matrix} x^{\underline{k+1}} \\ x^{\underline{k}} \end{matrix} \Delta^{-1}$$

discrete analogs to: derivative, anti-derivative (or indefinite integral)

$$\sum_{x=a}^{b-1} p(x) = \Delta^{-1} p(x)\Big|_a^b = \Delta^{-1} p(b) - \Delta^{-1} p(a)$$

- given a loop with known per-iteration count: e.g.,

for i=5 to N

. . . *(executes* $3i^{\underline{1}} + 7i^{\underline{2}}$ *instructions)*

- the total count of the loop is:

$$\sum_{i=5}^{N-1} \quad 3 \cdot i^{\underline{1}} + 7 \cdot i^{\underline{2}}$$

$$\Big\downarrow \Delta^{-1}$$

$$= \quad 3 \cdot i^{\underline{2}} + 7 \cdot i^{\underline{3}} \Big|_5^N = (3 \cdot N^{\underline{2}} + 7 \cdot N^{\underline{3}}) - \underbrace{(3 \cdot 5^{\underline{2}} + 7 \cdot 5^{\underline{3}})}_{=100}$$

$$\sum_{x=a}^{b-1} p(x) = \Delta^{-1} p(x)\Big|_a^b = \Delta^{-1} p(b) - \Delta^{-1} p(a)$$

- given a loop with known per-iteration count: e.g.,

  `for` i`=5 to` N

  . . . *(executes* $3i^{\underline{1}|} + 7i^{\underline{2}|}$ *instructions)*

- the count *before* (the start of) an iteration i (the rank of…)

$$\begin{aligned} &\sum_{i=5}^{i-1} \quad p(i) \qquad \text{(hmm…)} \\ = \quad & \Delta^{-1} p(i)\Big|_5^i = \underbrace{(3 \cdot i^{\underline{2}|} + 7 \cdot i^{\underline{3}|})}_{\Delta_i^{-1} p} - \underbrace{(3 \cdot 5^{\underline{2}|} + 7 \cdot 5^{\underline{3}|})}_{\Delta^{-1} p(5)} \end{aligned}$$

Abstract Syntax Trees

- strict alternation between loops and sequences of statements

```
Loop  :=  for id = expr to expr Seq
Seq   :=  do (Loop|name)+ done
```

- every statement *and* sequence is decorated with polynomials

- bottom-up traversal, post-order addition/summation
- on a basic instruction:

```
S                      → 1
```

- on a sequence:

```
do                     → c₀ + c₁ + ···
    s0   (with count c0)
    s1   (with count c1)
    . . .
done
```

$\texttt{do} \longrightarrow c_0 + c_1 + \cdots$

- on a loop:

```
for i=l to u           → Δ^{-1}c(u) − Δ^{-1}c(l)
    do   (with count c)
         . . .
    done
```

$\texttt{for i=}l\texttt{ to }u \longrightarrow \Delta^{-1}c(u) - \Delta^{-1}c(l)$

```
with n when n >= 0                                          (count)
do
    for i = 0 to n
        do
            S1
            for j = 0 to i
                do S2 done
            S3
            for j = 1+i to n
                do
                    S4
                    for k = 0 to i
                        do S5 done
                    S6
                done
        done
done
```

```
with n when n >= 0                                               (count)
do
    for i = 0 to n
        do
            S1                                                  1
            for j = 0 to i
                do S2 done
            S3
            for j = 1+i to n
                do
                    S4
                    for k = 0 to i
                        do S5 done
                    S6
                done
        done
done
```

```
with n when n >= 0                                          (count)
do
    for i = 0 to n
        do
            S1                                          1
            for j = 0 to i
                do S2 done                              1
            S3
            for j = 1+i to n
                do
                    S4
                    for k = 0 to i
                        do S5 done
                    S6
                done
        done
done
```

```
with n when n >= 0                                        (count)
do
    for i = 0 to n
        do
            S1                                          1
            for j = 0 to i                              i    = Σ_{j=0}^{i-1} 1
                do S2 done                         1
            S3
            for j = 1+i to n
                do
                    S4
                    for k = 0 to i
                        do S5 done
                    S6
                done
        done
done
```

```
with n when n >= 0                                          (count)
do
    for i = 0 to n
        do
            S1                                          1
            for j = 0 to i                              i
                do S2 done                          1
            S3                                          1
            for j = 1+i to n
                do
                    S4
                    for k = 0 to i
                        do S5 done
                    S6
                done
        done
done
```

```
with n when n >= 0                                          (count)
do
    for i = 0 to n
        do
            S1                                          1
            for j = 0 to i                              i
                do S2 done                          1
            S3                                          1
            for j = 1+i to n
                do
                    S4                              1
                    for k = 0 to i
                        do S5 done
                    S6
                done
        done
done
```

```
with n when n >= 0                                        (count)
do
    for i = 0 to n
        do
            S1                                        1
            for j = 0 to i                            i
                do S2 done                        1
            S3                                        1
            for j = 1+i to n
                do
                    S4                            1
                    for k = 0 to i
                        do S5 done            1
                    S6
                done
        done
done
```

```
with n when n >= 0                                        (count)
do
    for i = 0 to n
        do
            S1                                        1
            for j = 0 to i                            i
                do S2 done                        1
            S3                                        1
            for j = 1+i to n
                do
                    S4                            1
                    for k = 0 to i                i
                        do S5 done            1
                    S6
                done
        done
done
```

$$= \sum_{k=0}^{i-1} 1$$

```
with n when n >= 0                                      (count)
do
    for i = 0 to n
        do
            S1                                          1
            for j = 0 to i                              i
                do S2 done                           1
            S3                                          1
            for j = 1+i to n
                do
                    S4                                  1
                    for k = 0 to i                      i
                        do S5 done                   1
                    S6                                  1
                done
        done
done
```

```
with n when n >= 0                                          (count)
do
    for i = 0 to n
        do
            S1                                          1
            for j = 0 to i                              i
                do S2 done                          1
            S3                                          1
            for j = 1+i to n
                do                                  2 + i     = 1 + i + 1
                    S4                          1
                    for k = 0 to i              i
                        do S5 done          1
                    S6                          1
                done
        done
done
```

```
with n when n >= 0                                   (count)
do
    for i = 0 to n
    do
        S1                                         1
        for j = 0 to i                             i
            do S2 done                        1
        S3                                         1
        for j = 1+i to n        -2 + 2n - 4i + in - 2i^(2)
            do                              2 + i
                S4                     1
                for k = 0 to i         i
                    do S5 done    1
                S6                     1
            done
    done
done
```

$$-2 + 2n - 4i + in - 2i^{\underline{2}} = \sum_{j=1+i}^{n-1} (2+i)$$

$$= (2+i) \cdot j^{\underline{1}} \Big|_{1+i}^{n}$$

```
with n when n >= 0                                        (count)
do
    for i = 0 to n
        do                                    2n − 3i + in − 2i^2   = 1 + i + 1 + ···
            S1                                                1
            for j = 0 to i                                    i
                do S2 done                                    1
            S3                                                1
            for j = 1+i to n              −2 + 2n − 4i + in − 2i^2
                do                                        2 + i
                    S4                                        1
                    for k = 0 to i                            i
                        do S5 done                            1
                    S6                                        1
                done
        done
done
```

`with n when n >= 0` (count)

`do`

`for i = 0 to n` $2n + 3n^{\boxed{2}} + n^{\boxed{3}} = \sum_{i=0}^{n-1} \cdots$

`do` $2n - 3i + in - 2i^{\boxed{2}}$

`S1` 1

`for j = 0 to i` i

`do S2 done` 1

`S3` 1

`for j = 1+i to n` $-2 + 2n - 4i + in - 2i^{\boxed{2}}$

`do` $2 + i$

`S4` 1

`for k = 0 to i` i

`do S5 done` 1

`S6` 1

`done`

`done`

`done`

	(count)
`with n when n >= 0`	(count)
`do`	$2n + 3n^{2} + n^{3} = \cdots$
`  for i = 0 to n`	$2n + 3n^{2} + n^{3}$
`    do`	$2n - 3i + in - 2i^{2}$
`      S1`	1
`      for j = 0 to i`	i
`        do S2 done`	1
`      S3`	1
`      for j = 1+i to n`	$-2 + 2n - 4i + in - 2i^{2}$
`        do`	$2 + i$
`          S4`	1
`          for k = 0 to i`	i
`            do S5 done`	1
`          S6`	1
`        done`	
`    done`	
`done`	

```
with n when n >= 0
do
    for i = 0 to n
        do
            S1
            for j = 0 to i
                do S2 done
            S3
            for j = 1+i to n
                do
                    S4
                    for k = 0 to i
                        do S5 done
                    S6
                done
        done
done
```

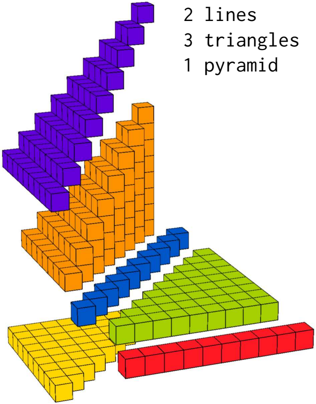
(count)

$\boxed{2n + 3n^{\underline{2}} + n^{\underline{3}}}$

2 lines

3 triangles

1 pyramid



- top-down traversal, propagating ranks and using counts

$\longrightarrow$ when processing a node, assign ranks to all its children

- on a sequence:

```
do          (with rank r)
     s0     (with count c0)  → r
     s1     (with count c1)  → r + c0
     s2     (with count c2)  → r + c0 + c1
     . . .
done
```

- on a loop

```
for i=l . . . (with rank r)
     do       (with count c)   → r + Δ_i^{-1}c − Δ_i^{-1}c(l)
          . . .
     done
```

- on a basic instruction: its rank is already set

	(rank)	(count)
`with n when n >= 0`		
`do`	0	
`  for i = 0 to n`		
`    do`		$2n - 3i + in - 2i^{2}$
`      S1`		1
`      for j = 0 to i`		i
`        do S2 done`		1
`      S3`		1
`      for j = 1+i to n`		
`        do`		$2 + i$
`          S4`		1
`          for k = 0 to i`		i
`            do S5 done`		1
`          S6`		
`        done`		
`    done`		
`done`		

the root of the AST is primed with rank 0

code	(rank)	(count)
`with n when n >= 0`		
`⟹ do`	0	
`for i = 0 to n`	0	
`do`		$2n - 3i + in - 2i^2$
`S1`		1
`for j = 0 to i`		i
`do S2 done`		1
`S3`		1
`for j = 1+i to n`		
`do`		$2 + i$
`S4`		1
`for k = 0 to i`		i
`do S5 done`		1
`S6`		
`done`		
`done`		
`done`		

the 1st statement in a sequence inherits the rank of the sequence

	(rank)	(count)
`with n when n >= 0`		
`do`	0	
$\Longrightarrow$ `for i = 0 to n`	0	
`  do`	$2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}} \xleftarrow{\Delta^{-1}}$	$2n - 3i + in - 2i^{\underline{2}}$
`    S1`	$=$ rank$_{\text{for}}$ (=0 here)	1
`    for j = 0 to i`	$+\,\Delta_i^{-1}$count$_{\text{do}}$ (wrt i)	i
`      do S2 done`	$-\,\Delta^{-1}$count$_{\text{do}}(0)$ (=0 here)	1
`    S3`		1
`    for j = 1+i to n`		
`      do`		$2+i$
`        S4`		1
`        for k = 0 to i`		i
`          do S5 done`		1
`        S6`		
`      done`		
`  done`		
`done`		

	(rank)	(count)
`with n when n >= 0`		
`do`	0	
`  for i = 0 to n`	0	
$\Longrightarrow$ `do`	$2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	$2n - 3i + in - 2i^{\underline{2}}$
`      S1`	$2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	1
`      for j = 0 to i`	$1 + 2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	i
`        do S2 done`		1
`      S3`	$1 + i + 2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	1
`      for j = 1+i to n`	$2 + i + 2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	
`        do`		$2 + i$
`          S4`		1
`          for k = 0 to i`		i
`            do S5 done`		1
`          S6`		
`        done`		
`    done`		
`done`		

S1 inherits the rank of the sequence
others accumulate counts

	(rank)	(count)
`with n when n >= 0`		
`do`	0	
`  for i = 0 to n`	0	
`    do`	$2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	$2n - 3i + in - 2i^{\underline{2}}$
`      S1`	$2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	1
$\Longrightarrow$ `for j = 0 to i`	$1 + 2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	i
`        do S2 done`	$1 + 2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}} + j$	1
`      S3`	$1 + i + 2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	1
`      for j = 1+i to n`	$2 + i + 2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	
`        do`		$2 + i$
`          S4`		1
`          for k = 0 to i`		i
`            do S5 done`		1
`          S6`		
`        done`		
`    done`		
`done`		

$+ \quad \Delta^{-1}$

$= \mathsf{rank}_{\mathsf{for}}$

$+ \Delta_j^{-1}\mathsf{count}_{\mathsf{do}}$ (wrt j, gives j)

$- \Delta^{-1}\mathsf{count}_{\mathsf{do}}(0)$ (=0 here)

	(rank)	(count)
`with n when n >= 0`		
`do`	0	
`  for i = 0 to n`	0	
`    do`	$2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	$2n - 3i + in - 2i^{\underline{2}}$
`      S1`	$2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	1
`      for j = 0 to i`	$1 + 2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	i
`        do S2 done`	$1 + 2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}} + j$	1
`      S3`	$1 + i + 2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	1
⟹ `for j = 1+i to n`	$2 + i + 2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	
`        do`	$-3i + 2in - 5i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}} + 2j + ji \xleftarrow{+}\ \xleftarrow{\Delta^{-1}}$	$2 + i$
`          S4`		1
`          for k = 0 to i`		i
`            do S5 done`		1
`          S6`		
`        done`		
`    done`		
`done`		

$= \mathsf{rank}_{\mathsf{for}}$

$+\ \Delta_j^{-1}\mathsf{count}_{\mathsf{do}}$ (wrt j, gives $2j + ji$)

$-\ \Delta^{-1}\mathsf{count}_{\mathsf{do}}(1 + i)$ $(= (2(1 + i) + (1 + i)i)$

	(rank)	(count)
`with n when n >= 0`		
`do`	0	
`  for i = 0 to n`	0	
`    do`	$2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	$2n - 3i + in - 2i^{\underline{2}}$
`      S1`	$2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	1
`      for j = 0 to i`	$1 + 2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	i
`        do S2 done`	$1 + 2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}} + j$	1
`      S3`	$1 + i + 2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	1
`      for j = 1+i to n`	$2 + i + 2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	
⟹ `do`	$-3i + 2in - 5i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}} + 2j + ji$	$2 + i$
`          S4`	$-3i + 2in - 5i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}} + 2j + ji$	1
`          for k = 0 to i`	$1 - 3i + 2in - 5i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}} + 2j + ji$	i
`            do S5 done`		1
`          S6`	$1 - 2i + 2in - 5i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}} + 2j + ji$	
`        done`		
`    done`		
`done`		

inheritance, plus a simple accumulation of counts

	(rank)	(count)
`with n when n >= 0`		
`do`	0	
`  for i = 0 to n`	0	
`    do`	$2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	$2n - 3i + in - 2i^{\underline{2}}$
`      S1`	$2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	1
`      for j = 0 to i`	$1 + 2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	i
`        do S2 done`	$1 + 2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}} + j$	1
`      S3`	$1 + i + 2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	1
`      for j = 1+i to n`	$2 + i + 2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	
`        do`	$-3i + 2in - 5i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}} + 2j + ji$	$2 + i$
`          S4`	$-3i + 2in - 5i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}} + 2j + ji$	1
⟹ `for k = 0 to i`	$1 - 3i + 2in - 5i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}} + 2j + ji$	i
`            do S5 done`	$1 - 3i + 2in - 5i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}} + 2j + ji + k$	$+$ Δ^{-1} 1
`          S6`	$1 - 2i + 2in - 5i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}} + 2j + ji$	
`        done`		
`    done`		
`done`		

$= \mathsf{rank}_{\mathsf{for}}$

$+\ \Delta_k^{-1}\mathsf{count}_{\mathsf{do}}$ (wrt k, gives k)

$-\ \Delta^{-1}\mathsf{count}_{\mathsf{do}}(0)$ (=0 here)

	(rank)	(count)
`with n when n >= 0`		
`do`	0	
`  for i = 0 to n`	0	
`    do`	$2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	$2n - 3i + in - 2i^{\underline{2}}$
`      S1`	$2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	1
`      for j = 0 to i`	$1 + 2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	i
`        do S2 done`	$1 + 2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}} + j \Leftarrow$	1
`      S3`	$1 + i + 2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	1
`      for j = 1+i to n`	$2 + i + 2in - 3i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}}$	
`        do`	$-3i + 2in - 5i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}} + 2j + ji \Leftarrow$	$2 + i$
`          S4`	$-3i + 2in - 5i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}} + 2j + ji \Leftarrow$	1
`          for k = 0 to i`	$1 - 3i + 2in - 5i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}} + 2j + ji \Leftarrow$	i
`            do S5 done`	$1 - 3i + 2in - 5i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}} + 2j + ji + k \Leftarrow$	1
`          S6`	$1 - 2i + 2in - 5i^{\underline{2}} + i^{\underline{2}}n - 2i^{\underline{3}} + 2j + ji \Leftarrow$	
`        done`		
`    done`		
`done`		

Note: many loop-body ranks are linear in the counter of the loop

Polyhedral arrays of structure of... (in extended Pascal syntax):
arrays as loops, records as sequences

```
data :
array [a:=0..n-1] of        position in
                            this array
    record
        s1: real;
        j1: array [b:=0 .. a-1] of
            real;
        s3: real;
        j2: array [b:=a+1 .. n-1] of
            record
                s4: real;
                k1: array [c:=0 .. a-1] of
                    real;
                s6: real;
            end;
    end;
```

Polyhedral arrays of structure of... (in extended Pascal syntax):
arrays as loops, records as sequences, count as size

```
data :
array [a:=0..n-1] of
    record
        s1: real;
        j1: array [b:=0 .. a-1] of
            real;
        s3: real;
        j2: array [b:=a+1 .. n-1] of
            record
                s4: real;
                k1: array [c:=0 .. a-1] of
                    real;
                s6: real;
            end;
    end;
```

position in this array

size of data[a].j2 $= -2 + 2n - 4a + an - 2a^{2}$

Polyhedral arrays of structure of... (in extended Pascal syntax): arrays as loops, records as sequences, count as size, rank as offset

```
data :
array [a:=0..n-1] of
    record
        s1: real;
        j1: array [b:=0 .. a-1] of
            real;
        s3: real;
        j2: array [b:=a+1 .. n-1] of
            record
                s4: real;
                k1: array [c:=0 .. a-1] of
                    real;
                s6: real;
            end;
    end;
```

position in this array

size of data[a].j2 $= -2 + 2n - 4a + an - 2a^{\underline{2}}$

offset of data[a].j2[b].k1[c] $= 1 - 3a + 2an - 5a^{\underline{2}} + a^{\underline{2}}n - 2a^{\underline{3}} + 2b + ba + c$

Polyhedral arrays of structure of... (in extended Pascal syntax):
arrays as loops, records as sequences, count as size, rank as offset

```
data :
array [a:=0..n-1] of
    record
        s1: real;
        j1: array [b:=0 .. a-1] of
            real;
        s3: real;
        j2: array [b:=a+1 .. n-1] of
            record
                s4: real;
                k1: array [c:=0 .. a-1] of
                    real;
                s6: real;
            end;
    end;
```

position in this array

size of data[a].j2 $= -2 + 2n - 4a + an - 2a^{\underline{2}}$

offset of data[a].j2[b].k1[c] $= 1 - 3a + 2an - 5a^{\underline{2}} + a^{\underline{2}}n - 2a^{\underline{3}} + 2b + ba + c$

This array has the shape of the Cholesky kernel iteration domain...

Polyhedral arrays of structure of... (in extended Pascal syntax): arrays as loops, records as sequences, count as size, rank as offset

```
data :
array [a:=0..n-1] of
    record
        s1: real;
        j1: array [b:=0 .. a-1] of
            real;
        s3: real;
        j2: array [b:=a+1 .. n-1] of
            record
                s4: real;
                k1: array [c:=0 .. a-1] of
                    real;
                s6: real;
            end;
    end;
```

position in this array

size of data[a].j2 = $-2 + 2n - 4a + an - 2a^{\underline{2}}$

offset of data[a].j2[b].k1[c] = $1 - 3a + 2an - 5a^{\underline{2}} + a^{\underline{2}}n - 2a^{\underline{3}} + 2b + ba + c$

This array has the shape of the Cholesky kernel iteration domain...

(my humble tribute to Niklaus Wirth)

Given a *valid* rank R (a number)
Find a path down the AST to determine:

- the location of the instruction with rank R
- the values of the enclosing loop counters

1. On a sequence of statements:

```
do
    s0   (with rank r0)
    s1   (with rank r1)
    . . .
done
```

$\rightarrow \max\{p \mid r_p(\vec{v}) \leq R\}$

generate conditional expressions?

2. On a loop

```
for i=l to u
    do  (with rank r)
        . . .
    done
```

$\rightarrow \max\{i \mid r(\vec{v}, i) \leq R\}$

$\equiv$ a root-finding problem...

Given a *valid* rank R (a number) and the values of the parameters
Find a path down the AST to determine:

- the location of the instruction with rank R
- the values of the enclosing loop counters

1. On a sequence of statements:

```
do
    s0   (with rank r0)
    s1   (with rank r1)
    . . .
done
```

$\rightarrow \max\{p \mid r_p(\vec{v}) \leq R\}$

all variables in scope have known values (in $\vec{v}$)
$\rightarrow$ simple scan

2. On a loop

```
for i=l to u
    do  (with rank r)
        . . .
    done
```

$\rightarrow \max\{i \mid r(\vec{v}, i) \leq R\}$

a root-finding problem requires numerical resolution ($r(\vec{v}, i)$ is univariate in i)

- generate code computing the result, to be used at runtime
- use a solver: `unisolve` (p, l, u, R)
  returns $\max\{x \mid l \le x < u \land p(x) \le R\}$

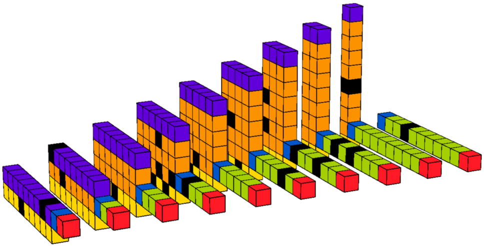
```
def dyn_unrank (n, RANK):
⇒ i = unisolve ([0, 2n, −3 + n, −2], 0, n, RANK)
   if RANK < 1 + 2in − 3i^{\underline{2}} + i^{\underline{2}}n − 2i^{\underline{3}}:
       return ([i], [0, 0])
   elif RANK < 1 + i + 2in − 3i^{\underline{2}} + i^{\underline{2}}n − 2i^{\underline{3}}:
     ⇒ j = unisolve ([1 + 2in − 3i^{\underline{2}} + i^{\underline{2}}n − 2i^{\underline{3}}, 1], 0, i, RANK)
       return ([i, j], [0, 1, 0])
   elif RANK < 2 + i + 2in − 3i^{\underline{2}} + i^{\underline{2}}n − 2i^{\underline{3}}:
       return ([i], [0, 2])
   else:
     ⇒ j = unisolve ([−3i + 2in − 5i^{\underline{2}} + i^{\underline{2}}n − 2i^{\underline{3}}, 2 + i], 1 + i, n, RANK)
       if RANK < 1 − 3i + 2in − 5i^{\underline{2}} + i^{\underline{2}}n − 2i^{\underline{3}} + 2j + ji:
           return ([i, j], [0, 3, 0])
       elif RANK < 1 − 2i + 2in − 5i^{\underline{2}} + i^{\underline{2}}n − 2i^{\underline{3}} + 2j + ji:
         ⇒ k = unisolve ([1 − 3i + 2in − 5i^{\underline{2}} + i^{\underline{2}}n − 2i^{\underline{3}} + 2j + ji, 1], 0, i, RANK)
           return ([i, j, k], [0, 3, 1, 0])
       else:
           return ([i, j], [0, 3, 2])
```

- generate code computing the result, to be used at runtime
- use a solver: `unisolve` $(p,\ l,\ u,\ R)$
  returns $\max\{x \mid l \le x < u \land p(x) \le R\}$

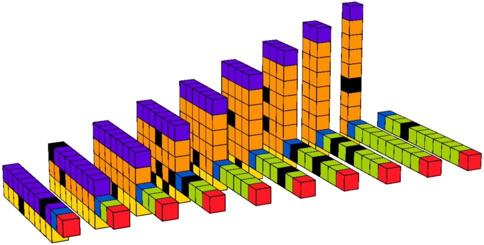
```
def dyn_unrank (n, RANK):
    i = unisolve ([0, 2n, −3 + n, −2], 0, n, RANK)
    if RANK < 1 + 2in − 3i^{\underline{2}} + i^{\underline{2}}n − 2i^{\underline{3}}:
        return ([i], [0, 0])
    elif RANK < 1 + i + 2in − 3i^{\underline{2}} + i^{\underline{2}}n − 2i^{\underline{3}}:
        j = unisolve ([1 + 2in − 3i^{\underline{2}} + i^{\underline{2}}n − 2i^{\underline{3}}, 1], 0, i, RANK)
        return ([i, j], [0, 1, 0])
    elif RANK < 2 + i + 2in − 3i^{\underline{2}} + i^{\underline{2}}n − 2i^{\underline{3}}:
        return ([i], [0, 2])
    else:
        j = unisolve ([−3i + 2in − 5i^{\underline{2}} + i^{\underline{2}}n − 2i^{\underline{3}}, 2 + i], 1 + i, n, RANK)
        if RANK < 1 − 3i + 2in − 5i^{\underline{2}} + i^{\underline{2}}n − 2i^{\underline{3}} + 2j + ji:
            return ([i, j], [0, 3, 0])
        elif RANK < 1 − 2i + 2in − 5i^{\underline{2}} + i^{\underline{2}}n − 2i^{\underline{3}} + 2j + ji:
            k = unisolve ([1 − 3i + 2in − 5i^{\underline{2}} + i^{\underline{2}}n − 2i^{\underline{3}} + 2j + ji, 1], 0, i, RANK)
            return ([i, j, k], [0, 3, 1, 0])
        else:
            return ([i, j], [0, 3, 2])
```

```
for i = 0 to n
    do
        S1
        for j = 0 to i
            do S2 done
        S3
        for j = 1+i to n
            do
                S4
                for k = 0 to i
                    do S5 done
                S6
            done
    done
```

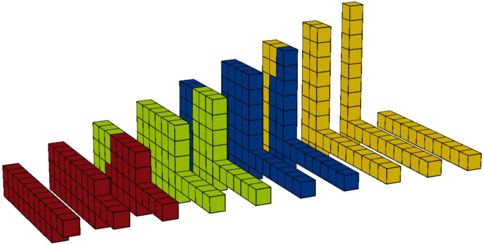
- Uniform random sampling: 10% with $n = 10 \longrightarrow 27$ out of 275



- Uniform random sampling: 10% with $n = 10 \longrightarrow 27$ out of 275



- Slicing: 4-way with $n = 10 \longrightarrow 275 = 3 \times 69 + 68$



- Pros
  - simple mathematical foundations
  - efficient algorithms
  - lightweight implementation
- Cons: strict restrictions on loops
  1. unit step

  $\rightarrow$ a fundamental difference with Barvinok/Ehrhart

  2. bounds coherence

  $\rightarrow$ an anecdotal difference, can be delegated

- Topics not covered in this talk
  - multivariate integer polynomials
  - symbolic algebraic operations
- Some trivial extensions:
  - polynomial bounds
  - weighted instructions