Московский государственный университет имени М.В.Ломоносова
Факультет Вычислительной математики и кибернетики

СУПЕРКОМПЬЮТЕРЫ И
ПАРАЛЛЕЛЬНАЯ ОБРАБОТКА ДАННЫХ

ВВЕДЕНИЕ В ТЕОРИЮ АНАЛИЗА СТРУКТУРЫ ПРОГРАММ И АЛГОРИТМОВ

(часть 1)

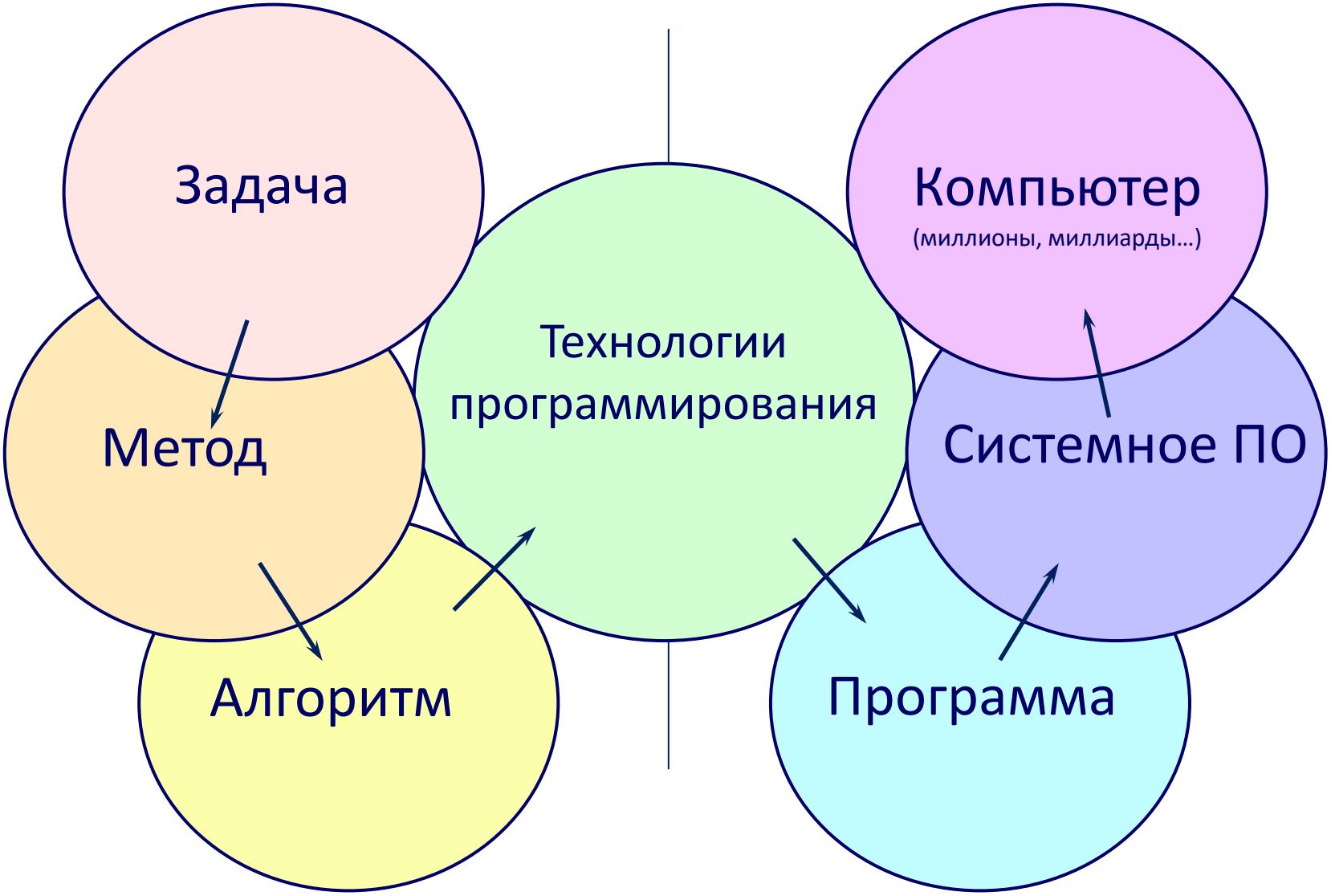
Вл.В.Воеводин

Зав.кафедрой Суперкомпьютеров и квантовой информатики ВМК МГУ
Директор НИВЦ МГУ,
д.ф.-м.н., профессор, чл.-корр. РАН

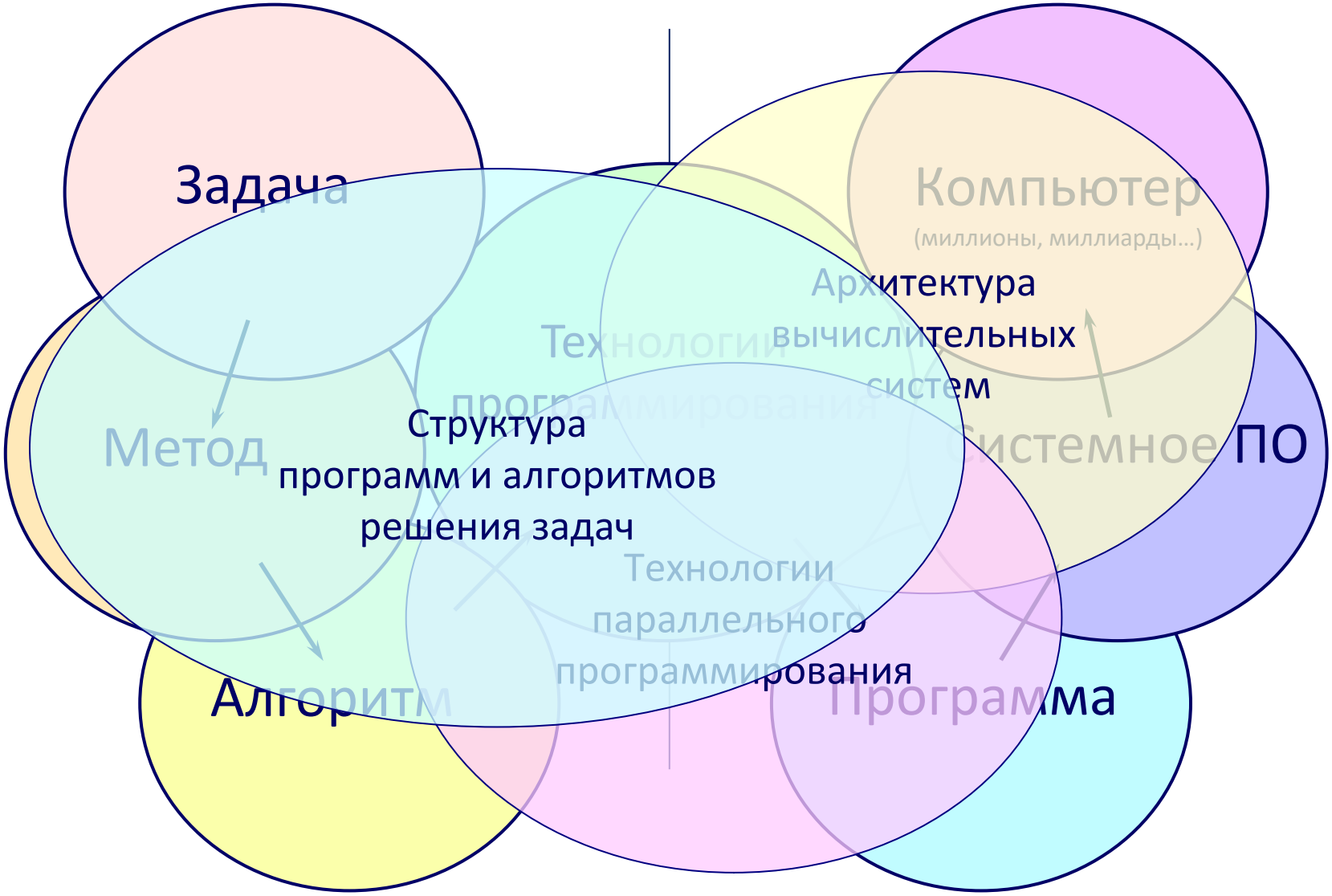
voevodin@parallel.ru

ВМК МГУ, 2020

Решение задачи на компьютере



Решение задачи на компьютере



Почему важно понимать как написаны программы?

Pascal? Fortran? C? C++?

Пользователь: почему?

$A_{ijk} = A_{i-1jk} + B_{jk} + B_{jk}$, i=1,40; j=1,40; k=1,1000

Cray C90, пиковая производительность 960 Mflop/s

```
do k = 1, 1000
   do j = 1, 40
      do i = 1, 40
         A(i,j,k) = A(i-1,j,k)+B(j,k)+B(j,k)
```

Производительность: **20** Mflop/s на Cray C90

Пользователь: почему?

$A_{ijk} = A_{i-1jk} + B_{jk} + B_{jk}$, i=1,40; j=1,40; k=1,1000

Cray C90, пиковая производительность 960 Mflop/s

```
do i = 1, 40, 2
   do j = 1, 40
      do k = 1, 1000
         A(i,j,k) = A(i-1,j,k)+2*B(j,k)
         A(i+1,j,k) = A(i,j,k)+2*B(j,k)
```

Производительность: **700** Mflop/s на Cray C90

Почему важно понимать как написаны программы?

Pascal? Fortran? C? C++?

Почему важно знать как устроены алгоритмы?

Умножение матриц: все ли просто?

Фрагмент исходного текста:

```
for( i = 0; i < n; ++i)
    for( j = 0; j < n; ++j)
        for( k = 0; k < n; ++k)
            A[i][j] = A[i][j] + B[i][k]*C[k][j]
```

Порядок циклов: (i, j, k)

Возможен ли порядок:

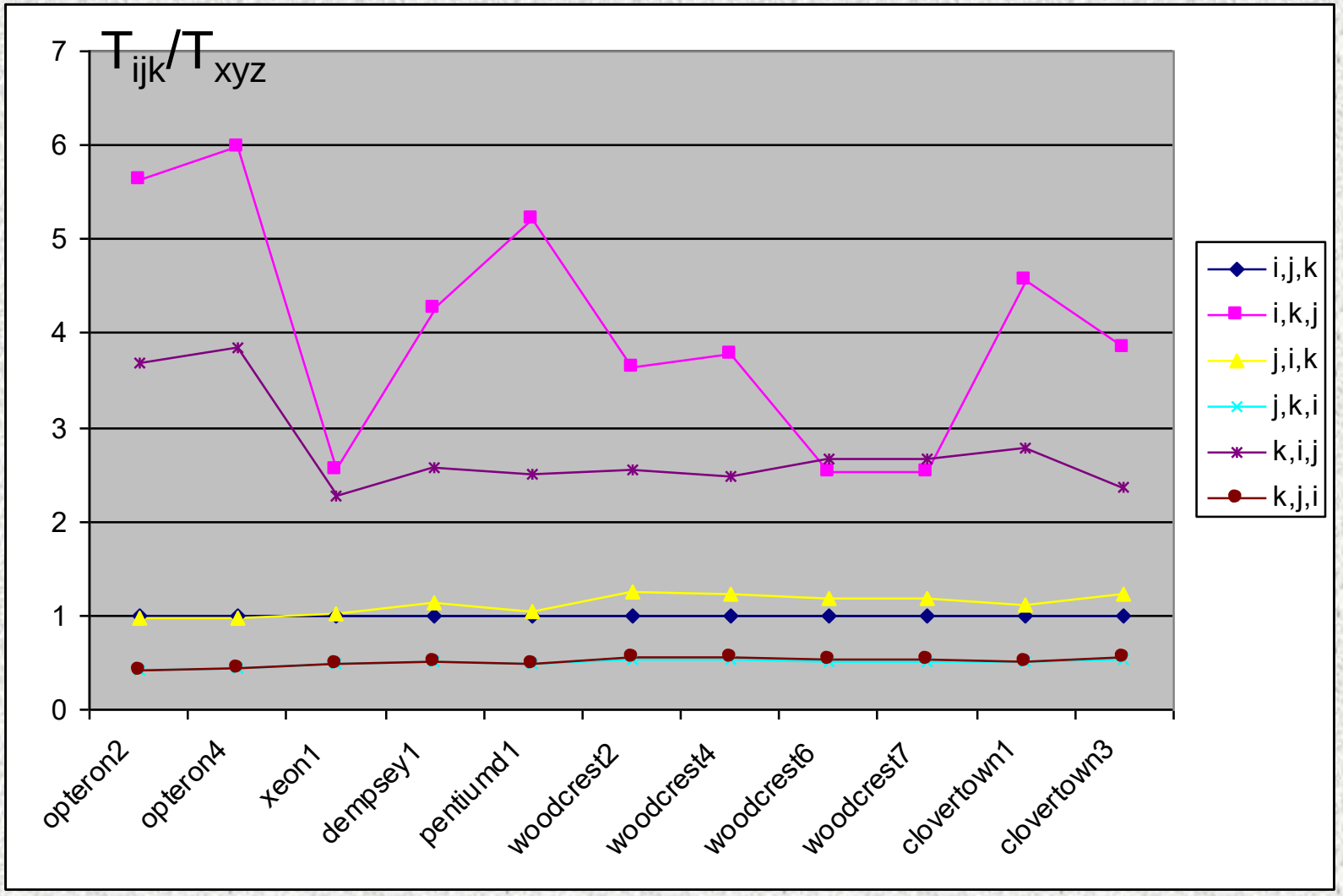
(i, k, j) - ?	ДА
(k, i, j) - ?	ДА
(k, j, i) - ?	ДА
(j, i, k) - ?	ДА
(j, k, i) - ?	ДА

Почему возможен другой порядок?

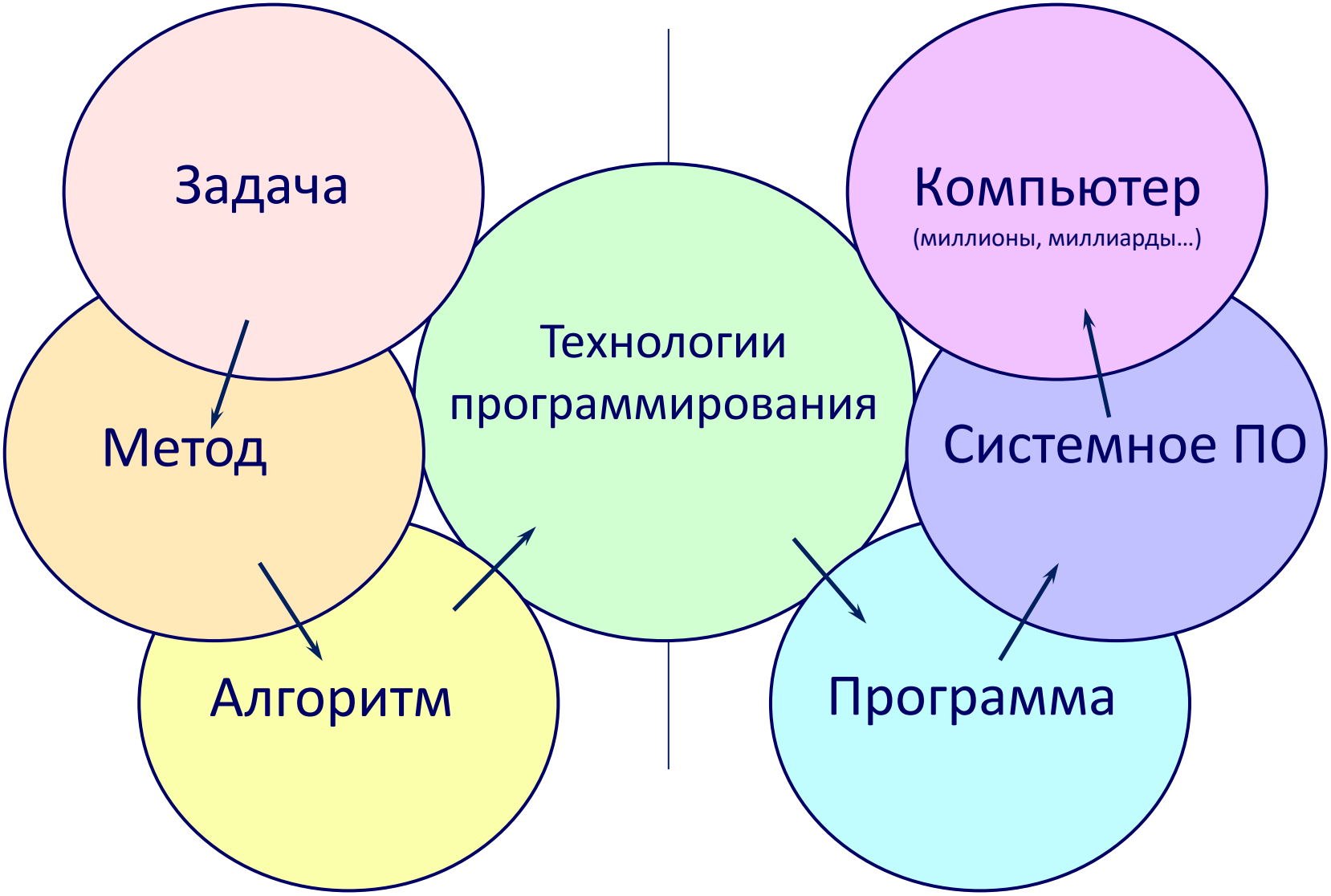
А зачем нужен другой порядок?

Умножение матриц: все ли просто?

(сравнение с порядком (i, j, k))



Решение задачи на компьютере



Графовые модели программ

Будем представлять программы с помощью графов: набор вершин и множество соединяющих их направленных дуг.

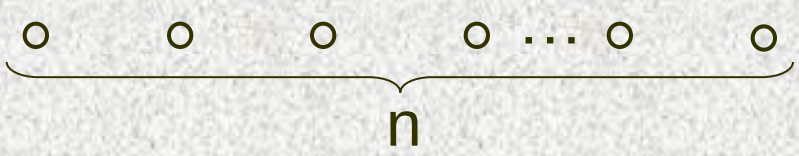
Вершины: процедуры, циклы, линейные участки, операторы, итерации циклов, срабатывания операторов…

Графовые модели программ

Будем представлять программы с помощью графов: набор вершин и множество соединяющих их направленных дуг.

Вершины: итерации циклов.

```
for( i = 0; i < n; ++i) {
    A[i] = A[i – 1] + 2;
    B[i] = B[i] + A[i];
}
```



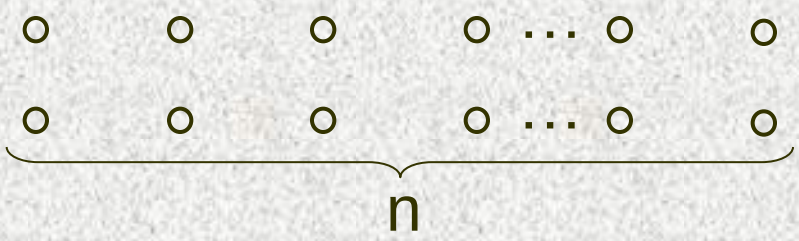
Каждая вершина соответствует двум операторам (телу цикла), выполненным на одной и той же итерации цикла.

Графовые модели программ

Будем представлять программы с помощью графов: набор вершин и множество соединяющих их направленных дуг.

Вершины: срабатывания операторов.

```
for( i = 0; i < n; ++i) {
    A[i] = A[i – 1] + 2;
    B[i] = B[i] + A[i];
}
```



Каждая вершина соответствует одному из двух операторов тела данного цикла, выполненному на некоторой итерации.

Графовые модели программ

Будем представлять программы с помощью графов: набор вершин и множество соединяющих их направленных дуг.

Вершины: процедуры, циклы, линейные участки, операторы, итерации циклов, срабатывания операторов…

Дуги: отражают связь (отношение) между вершинами. Выделяют два типа отношений:

- *операционное отношение,*
- *информационное отношение.*

Графовые модели программ

Будем представлять программы с помощью графов: набор вершин и множество соединяющих их направленных дуг.

Дуги: операционное отношение:

Две вершины A и B соединяются направленной дугой тогда и только тогда, когда вершина B может быть выполнена сразу после вершины A.

Операционное отношение = отношение по передаче управления.

Графовые модели программ

Будем представлять программы с помощью графов: набор вершин и множество соединяющих их направленных дуг.

Дуги: операционное отношение:

x(i) = a + b(i) (1)
y(i) = 2*x(i) – 3 (2)
t1 = y(i)*y(i) + 1 (3)
t2 = b(i) – y(i)*a (4)

Графовые модели программ

Будем представлять программы с помощью графов: набор вершин и множество соединяющих их направленных дуг.

Дуги: информационное отношение:

Две вершины A и B соединяются направленной дугой тогда и только тогда, когда вершина B использует в качестве аргумента некоторое значение, полученное в вершине A.

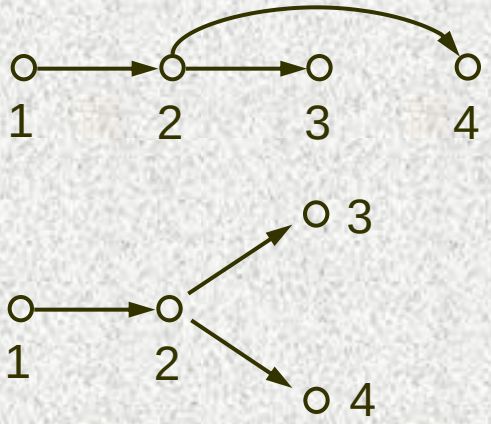
Информационное отношение = отношение по передаче данных.

Графовые модели программ

Будем представлять программы с помощью графов: набор вершин и множество соединяющих их направленных дуг.

Дуги: информационное отношение:

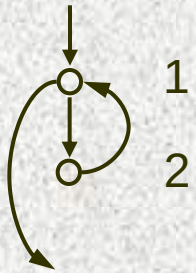
x(i) = a + b(i) (1)
y(i) = 2*x(i) – 3 (2)
t1 = y(i)*y(i) + 1 (3)
t2 = b(i) – y(i)*a (4)



Четыре основные модели программ

Граф управления программы.
Вершины: операторы
Дуги: операционное отношение

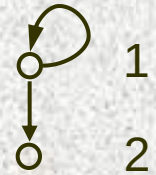
```
for( i = 0; i < n; ++i) {
    A[i] = A[i – 1] + 2;        (1)
    B[i] = B[i] + A[i];         (2)
}
```



Четыре основные модели программ

Информационный граф программы.
Вершины: операторы
Дуги: информационное отношение

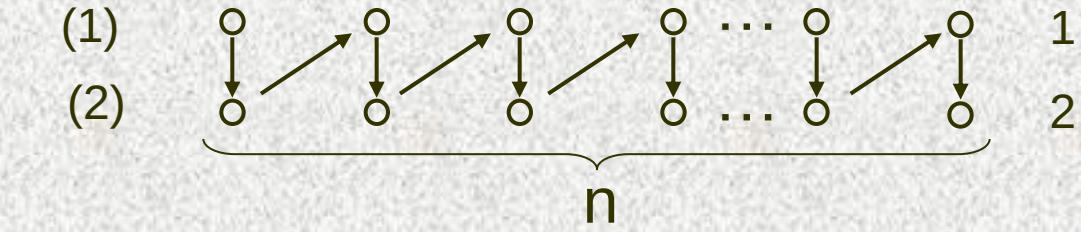
```
for( i = 0; i < n; ++i) {
    A[i] = A[i – 1] + 2;          (1)
    B[i] = B[i] + A[i];           (2)
}
```



Четыре основные модели программ

Операционная история программы.
Вершины: срабатывания операторов
Дуги: операционное отношение

```
for( i = 0; i < n; ++i) {
    A[i] = A[i – 1] + 2;
    B[i] = B[i] + A[i];
}
```

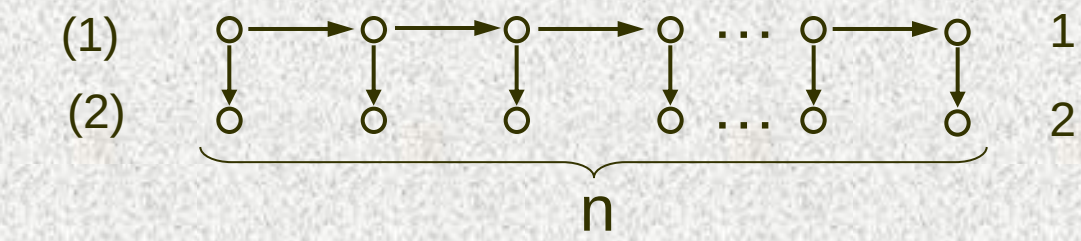


Свойства операционной истории:
- одна начальная вершина, у которой нет входящей дуги,
- одна конечная вершина, у которой нет исходящей дуги,
- у всех остальных вершин есть ровно одна входящая дуга и одна исходящая дуга.

Четыре основные модели программ

Информационная история программы.
Вершины: срабатывания операторов
Дуги: информационное отношение

```
for( i = 0; i < n; ++i) {
    A[i] = A[i – 1] + 2;
    B[i] = B[i] + A[i];
}
```



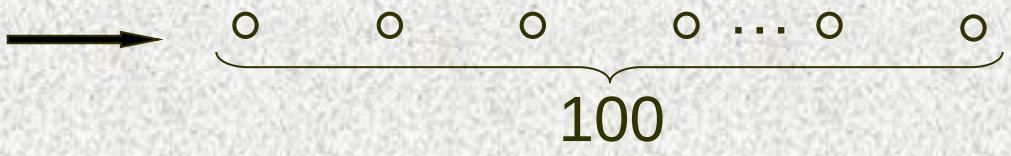
Свойства информационной истории:
- ациклический граф,
- нет кратных дуг.

Несколько вопросов…

Может ли информационная история некоторого фрагмента программы иметь 100 вершин и ни одной дуги?

ДА.

```
for( i = 0; i < 100; ++i)
    A[i] = B[i] + C[i]*x;
```

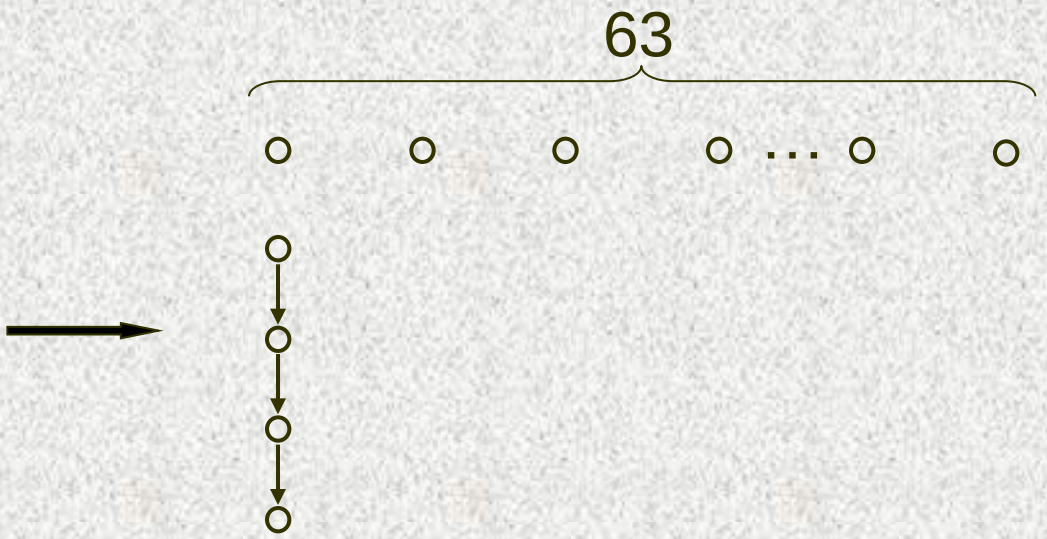


Несколько вопросов…

Может ли информационная история некоторого фрагмента программы иметь 67 вершин и 3 дуги?

ДА.

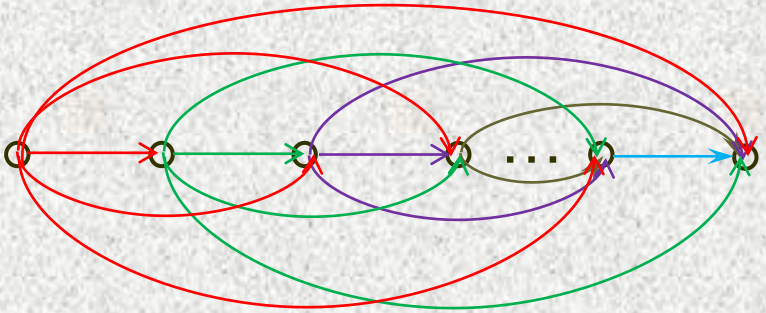
```
for( i = 0; i < 63; ++i)
    A[i] = B[i] + C[i]*x;
x1 = 10;
x2 = x1+1;
x3 = x2+2;
x4 = x3+3;
```



Несколько вопросов…

Может ли информационная история некоторого фрагмента программы иметь 20 вершин и 200 дуг?

Вспомним свойства информационной истории:
- ациклический граф,
- нет кратных дуг.



Макс.число дуг: (n-1) + (n-2) + (n-3) + … + 2 + 1 = n(n-1)/2*

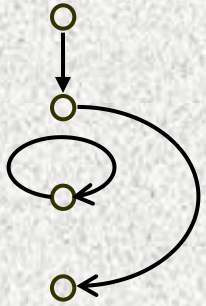
Ответ: НЕТ

Несколько вопросов…

Может ли граф управления некоторого фрагмента программы состоять из нескольких компонент связности?

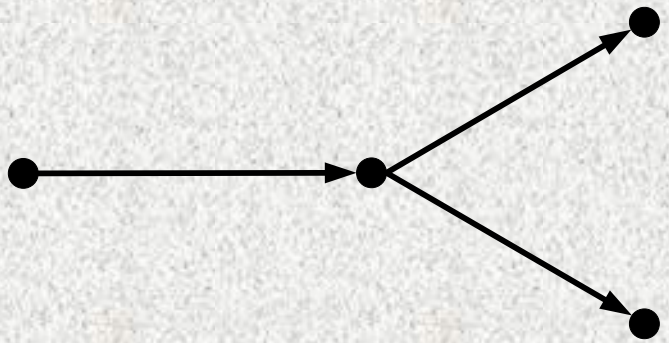
ДА.

```
x1 = 10;
x2 = x1+1; goto A;
B: x3 = x2+2; goto B;
A: x4 = x2+3;
```



Несколько вопросов…

Модель некоторого фрагмента программы в качестве подграфа содержит следующий граф:



Какой моделью могла бы быть исходная модель?

ГУ ИГ ~~ОИ~~ ИИ

Множество графовых моделей программ

(опорные точки)

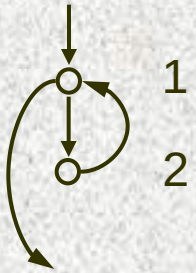
Операционные модели

Информационные модели

Компактные модели

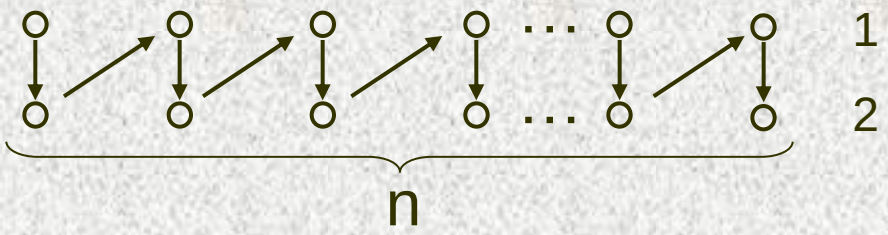
Истории

Граф управления

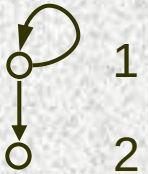


…

Операционная история

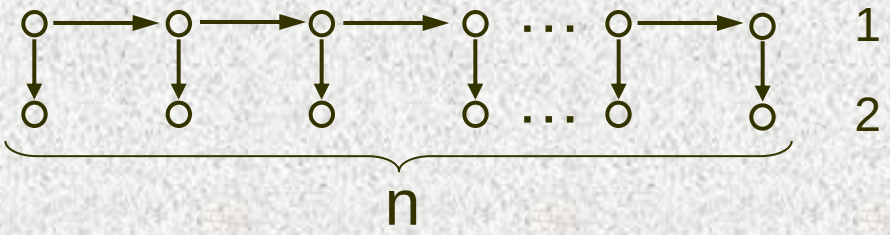


Информационный граф



…

Информационная история



Какое отношение выбрать для описания свойств программ?

Операционное отношение?

x(i) = a + b(i) (1)
y(i) = 2*x(i) – 3 (2)
t1 = y(i)*y(i) + 1 (3)
t2 = b(i) – y(i)*a (4)

Какое отношение выбрать для описания свойств программ?

Информационная структура – это основа анализа свойств программ и алгоритмов.

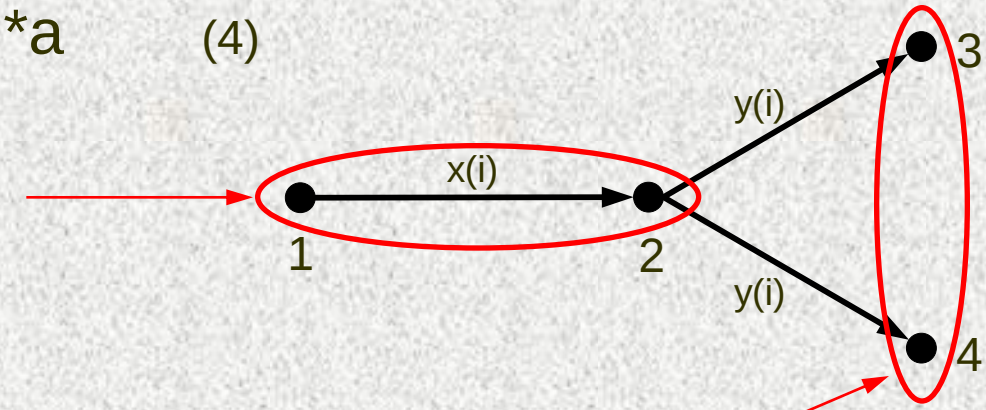
$$x(i) = a + b(i) \quad (1)$$

$$y(i) = 2*x(i) - 3 \quad (2)$$

$$t1 = y(i)*y(i) + 1 \quad (3)$$

$$t2 = b(i) - y(i)*a \quad (4)$$

Исполнять только последовательно !

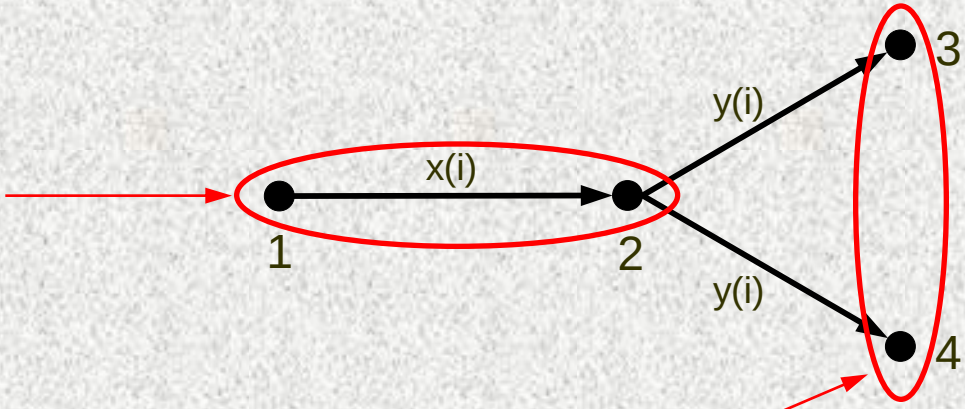


Можно исполнять параллельно !

Какое отношение выбрать для описания свойств программ?

Информационная структура – это основа анализа свойств программ и алгоритмов.

Исполнять только последовательно !



Можно исполнять параллельно !

Информационная зависимость определяет критерий эквивалентности преобразований программ.

Информационная независимость определяет ресурс параллелизма программы.

От компактных до историй: что выбрать для описания свойств программ?

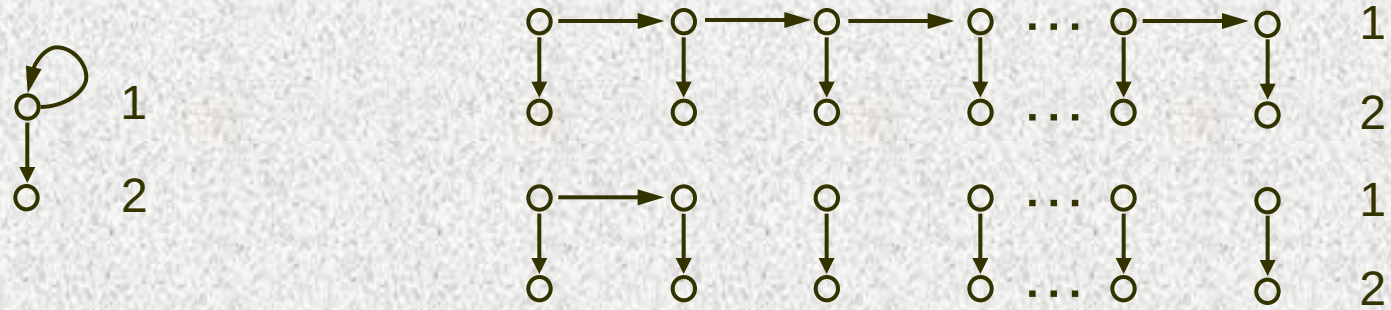
Аргументы для выбора степени компактности модели:

- *компактность описания,*
- *информативность,*
- *сложность построения.*

От компактных до историй: что выбрать для описания свойств программ?

Аргументы для выбора степени компактности модели:

- *компактность описания,*
- *информативность,*



- *сложность построения.*

От компактных до историй: что выбрать для описания свойств программ?

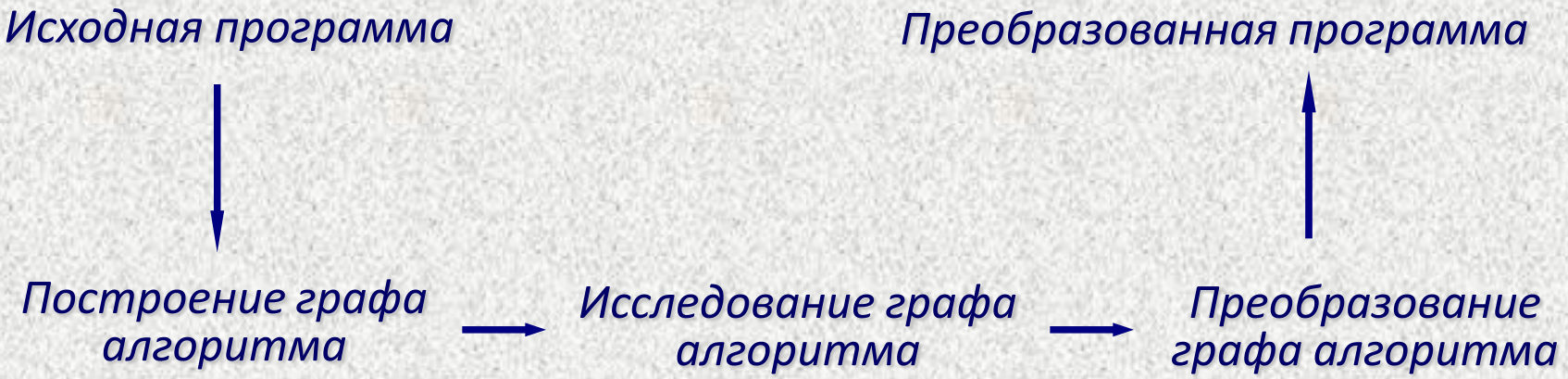
Аргументы для выбора степени компактности модели:

- *компактность описания,* *(компактные +)*
- *информативность,* *(истории +)*
- *сложность построения.* *(компактные +)*

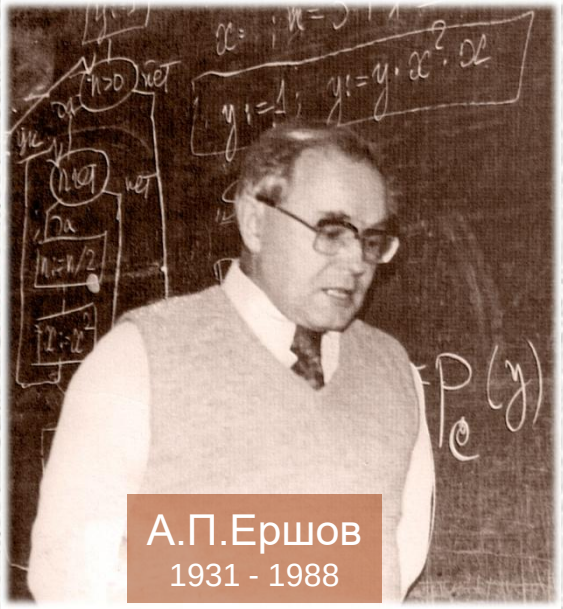
Граф алгоритма – это параметризованная информационная история:

- *компактность описания за счет параметризации,*
- *имеет информативность истории,*
- *разработана методика построения графа алгоритма по исходному тексту программ.*

Общая схема анализа и преобразования структуры программ



Основатели теории анализа структуры программ и алгоритмов



Ершов Андрей Петрович, академик, создатель сибирской школы системного и теоретического программирования. Многие его работы посвящены методам изучения свойств и структуры программ. Еще в 60-х годах он рассматривал задачу преобразования схем программ над общей и распределенной памятью, изучал фундаментальные основы графовых моделей программ.

Воеводин Валентин Васильевич, академик, создатель математической теории информационной структуры программ и алгоритмов. Разработал методы нахождения и описания информационной структуры программ по их исходному тексту, методы определения потенциала параллелизма и эквивалентного преобразования программ.

Теорема о построении графа алгоритма

<u>Теорема.</u> Если фрагмент принадлежит к линейному классу программ, то на основе статического анализа можно построить компактное описание его графа алгоритма в следующем виде:
для каждого входа каждого оператора фрагмента будет указано конечное множество троек вида

$$(N, \Delta(N), F(\Delta, N))_k ,$$

где:
N – линейный выпуклый многогранник в пространстве внешних переменных фрагмента,
$\Delta(N)$ – линейный выпуклый многогранник в пространстве итераций фрагмента,
$F(\Delta, N)$ – линейная векторная функция, описывающая входящие дуги.

Теорема о построении графа алгоритма

(…простыми словами…)

<u>Теорема.</u> Если фрагмент принадлежит к линейному классу программ, то на основе статического анализа можно построить компактное описание его графа алгоритма в следующем виде:
для каждого входа каждого оператора фрагмента будет указано конечное множество троек вида

$$(N, \Delta(N), F(\Delta, N))_k ,$$

где:
N – линейный выпуклый многогранник в пространстве внешних переменных фрагмента,
Δ(N) – линейный выпуклый многогранник в пространстве итераций фрагмента,
F(Δ, N) – линейная векторная функция, описывающая входящие дуги.

аргументы операторов

входные данные программы

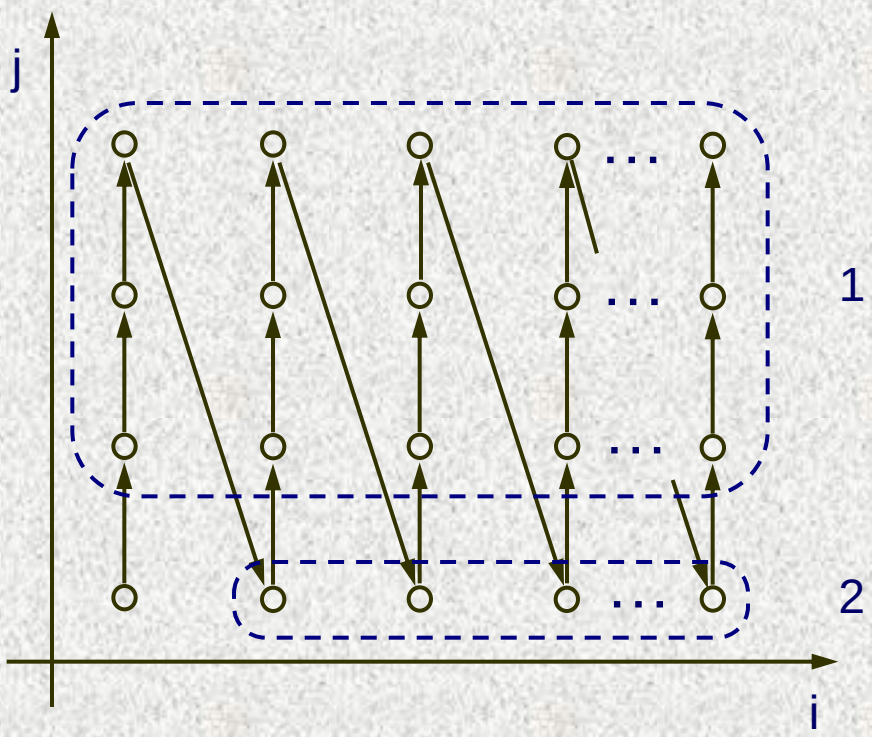
срабатывания операторов

информационное отношение

Как выполняется описание структуры программ на практике?

Программы и их графы алгоритма

```
Do i = 1, n
   Do j = 1, m
      s = s + A(i, j)
```



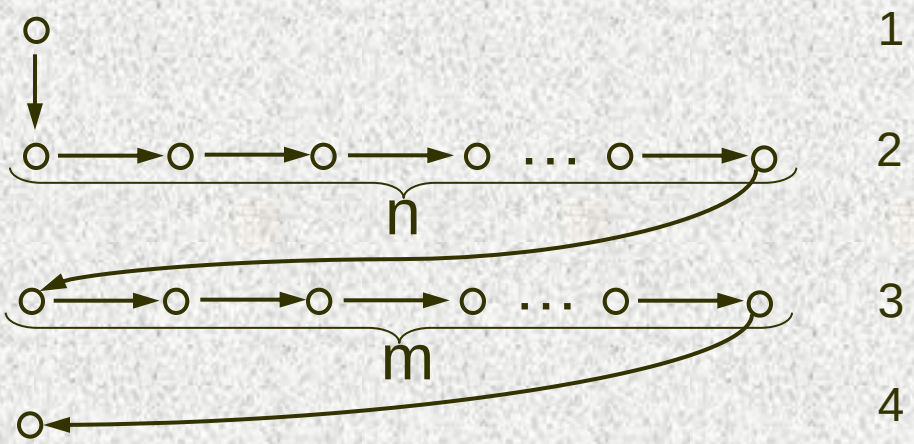
Для входа s:

$$N_1 = \begin{cases} n \ge 1 \\ m \ge 2 \end{cases} \quad I_1 = \begin{cases} 1 \le i \le n \\ 2 \le j \le m \end{cases} \quad F_1 = \begin{cases} i' = i \\ j' = j - 1 \end{cases}$$

$$N_2 = \begin{cases} n \ge 2 \\ m \ge 1 \end{cases} \quad I_2 = \begin{cases} 2 \le i \le n \\ j = 1 \end{cases} \quad F_2 = \begin{cases} i' = i - 1 \\ j' = m \end{cases}$$

Программы и их графы алгоритма

```
s = 0                    (1)
Do i = 1, n
   s = s + 1             (2)
Do i = 1, m
   s = s + 1             (3)
s = s + 1                (4)
```



$$\begin{cases} m \geq 1 \\ j_1 = m \\ из\ 3 \end{cases} \qquad \begin{cases} m < 1 \\ n \geq 1 \\ j_1 = n \\ из\ 2 \end{cases} \qquad \begin{cases} m < 1 \\ n < 1 \\ из\ 1 \end{cases}$$

Вспомним прошлые лекции…

(Умножение матриц: все ли просто?)

Фрагмент исходного текста:

```
for( i = 0; i < n; ++i)
    for( j = 0; j < n; ++j)
        for( k = 0; k < n; ++k)
            A[i][j] = A[i][j] + B[i][k]*C[k][j]
```

Порядок циклов: (i, j, k)

Возможен ли порядок:

(i, k, j) - ?	ДА
(k, i, j) - ?	ДА
(k, j, i) - ?	ДА
(j, i, k) - ?	ДА
(j, k, i) - ?	ДА

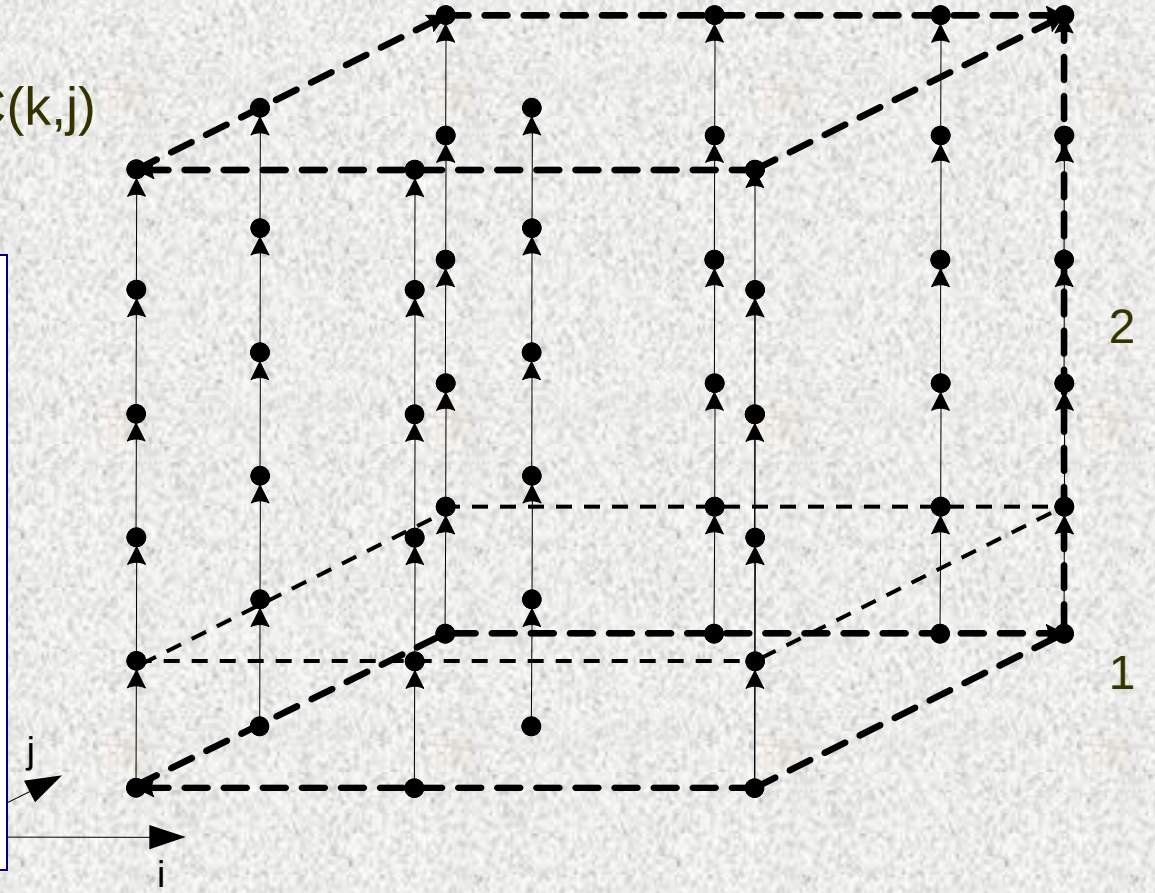
Почему возможен другой порядок?

А зачем нужен другой порядок?

Программы и их графы алгоритма
(умножение матриц)

```
    Do i = 1, n
       Do j = 1, n
1      A(i,j) = 0
         Do k = 1, n
2          A(i,j) = A(i,j) + B(i,k)*C(k,j)
```

$$\begin{cases} 1 \le i \le n \\ 1 \le j \le n \\ k = 1 \\ \begin{cases} i_1 = i \\ j_1 = j \end{cases} \\ из\ (1) \end{cases} \qquad \begin{cases} 1 \le i \le n \\ 1 \le j \le n \\ 2 \le k \le n \\ \begin{cases} i_1 = i \\ j_1 = j \\ k_1 = k - 1 \end{cases} \\ из\ (2) \end{cases}$$



Предварительные объявления о зачете…

- *Крайний срок сдачи отчетов по практическому заданию – 3 декабря.*
- *Заявлены три выступления в формате стенд-ап, которые пройдут на второй половине лекции 8-го декабря, 10 минут на каждое выступление:*

1. Черкасов Владислав, 416 группа. Как суперкомпьютер справляется с приземлёнными задачами?
2. Раева Анастасия, 412 группа. Суперкомпьютеры и их применение.
3. Гринев Тимофей, 416 группа. Введение в теорию суперлюдей (пародия на теорию суперкомпьютеров)

- *Список вопросов к зачету опубликован на https://parallel.ru/vmk20*
- *Консультация по курсу состоится 12 декабря, суббота, в 14 часов. Ссылку вышлем дополнительно.*
- *Итоги по конкурсу изображений, выступлениям в формате стенд-ап и выполнению практических заданий будут подведены к 12 декабря.*
- *Электронное тестирование пройдет 15-го декабря в 14:35, пароли будут высланы старостам групп.*
- *Устные зачеты (экзамены) – 24 и 28 декабря с 10:00, расписание по группам и ссылки будут высланы позже. Необходима включенная камера на стороне студентов и преподавателей.*

Московский государственный университет имени М.В.Ломоносова
Факультет Вычислительной математики и кибернетики

СУПЕРКОМПЬЮТЕРЫ И
ПАРАЛЛЕЛЬНАЯ ОБРАБОТКА ДАННЫХ

ВВЕДЕНИЕ В ТЕОРИЮ АНАЛИЗА СТРУКТУРЫ ПРОГРАММ И АЛГОРИТМОВ

(часть 1)

Вл.В.Воеводин

Зав.кафедрой Суперкомпьютеров и квантовой информатики ВМК МГУ
Директор НИВЦ МГУ,
д.ф.-м.н., профессор, чл.-корр. РАН

voevodin@parallel.ru

ВМК МГУ, 2020