

Графовые модели программ

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

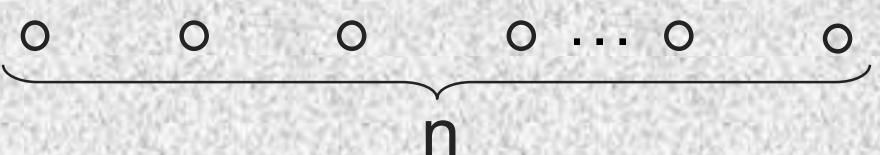
*Вершины: процедуры, циклы, линейные участки,
операторы, итерации циклов, срабатывания
операторов...*

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];  
}
```



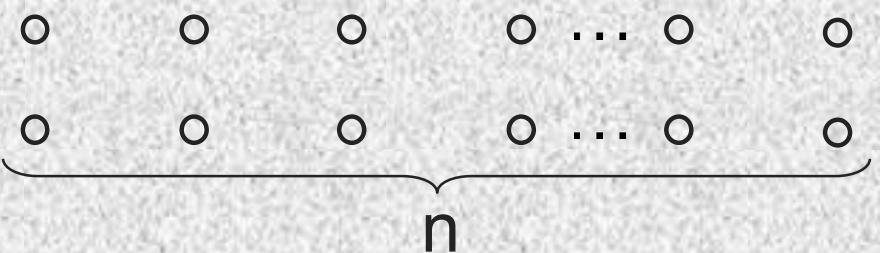
*Каждая вершина соответствует
двум операторам (телу цикла),
выполненным на одной и той же
итерации цикла.*

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 А и В соединяются направленной дугой
тогда и только тогда, когда вершина В может быть
выполнена сразу после вершины А.*

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:

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

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

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

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



Графовые модели программ

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

Дуги: информационное отношение:



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

Информационное отношение = отношение по передаче данных.

Графовые модели программ

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

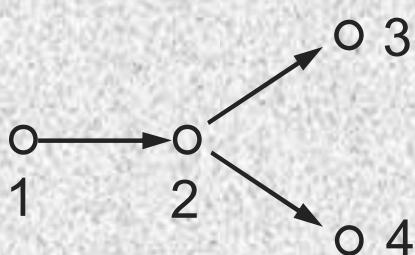
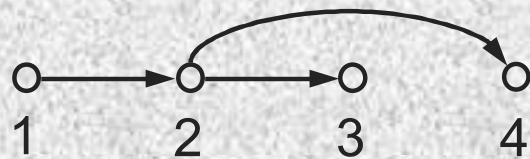
Дуги: информационное отношение:

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

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

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

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



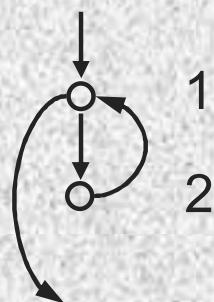
Четыре основные модели программ

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)  
}
```



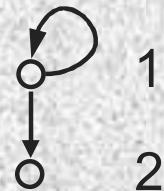
Четыре основные модели программ

Информационный граф программы.

Вершины: операторы

Дуги: информационное отношение

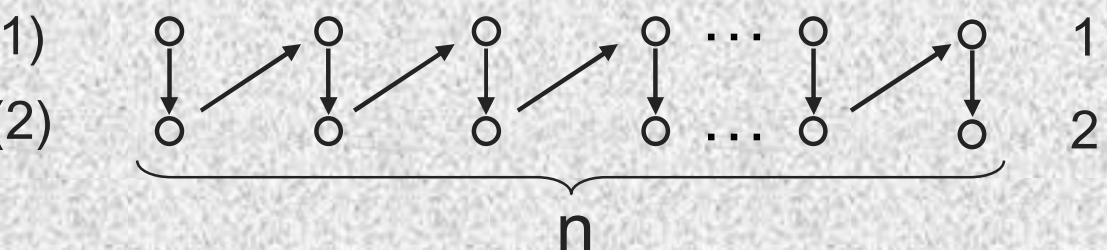
```
for( i = 0; i < n; ++i) {  
    A[i] = A[i - 1] + 2;          (1)  
    B[i] = B[i] + A[i];          (2)  
}
```



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;          (1)  
    B[i] = B[i] + A[i];          (2)  
}
```



Четыре основные модели программ

Информационная история программы.

Вершины: срабатывания операторов

Дуги: информационное отношение

```
for( i = 0; i < n; ++i) {
```

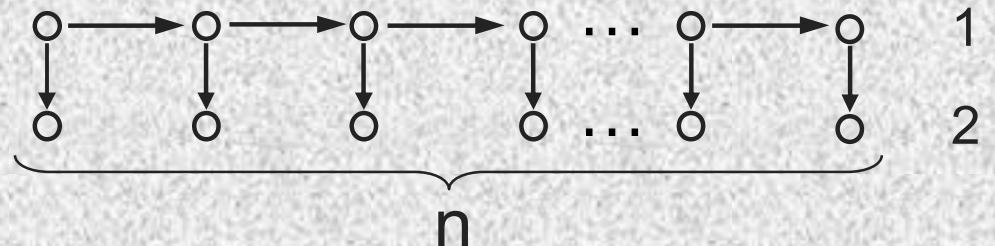
```
    A[i] = A[i - 1] + 2;
```

```
    B[i] = B[i] + A[i];
```

```
}
```

(1)

(2)

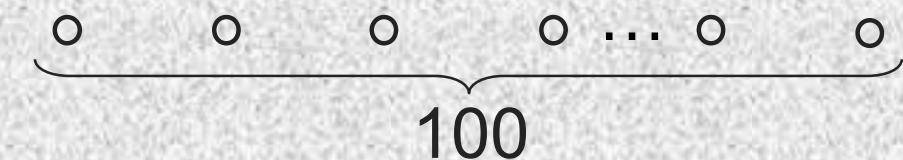


Несколько вопросов...

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;
```

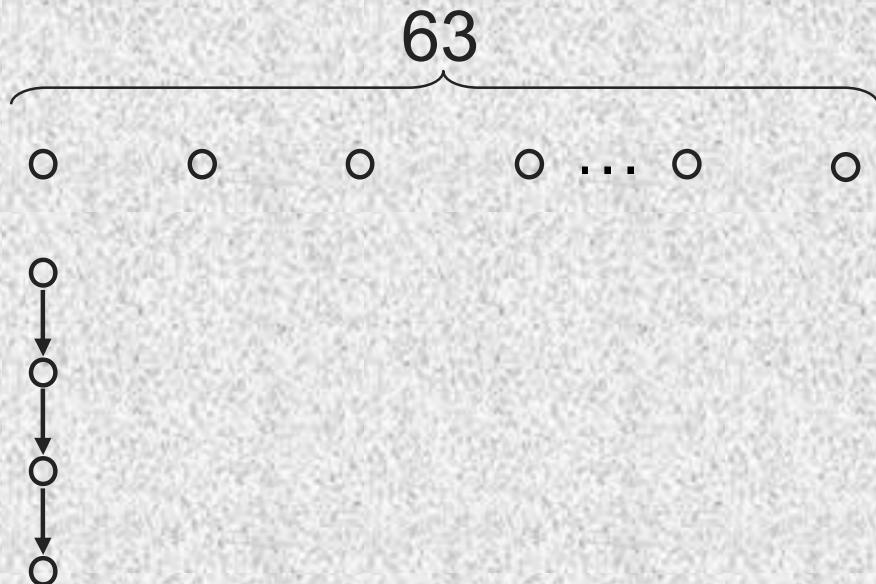


Несколько вопросов...

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;
```



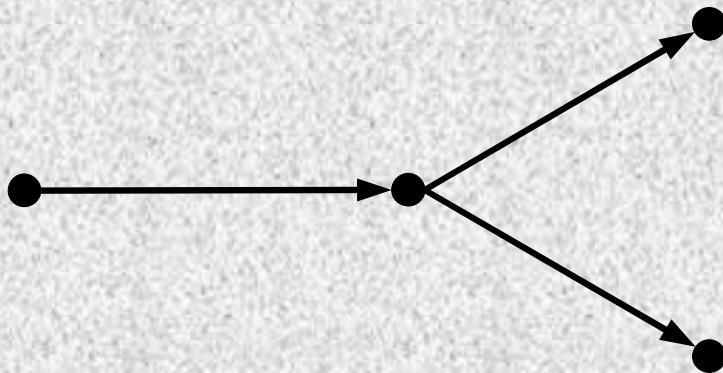
Несколько вопросов...

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

НЕТ.

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)



Какое отношение выбрать для описания свойств программ?

Операционное отношение?

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

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

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

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



Какое отношение выбрать для описания свойств программ?

Информационная структура – это основа анализа свойств программ и алгоритмов.

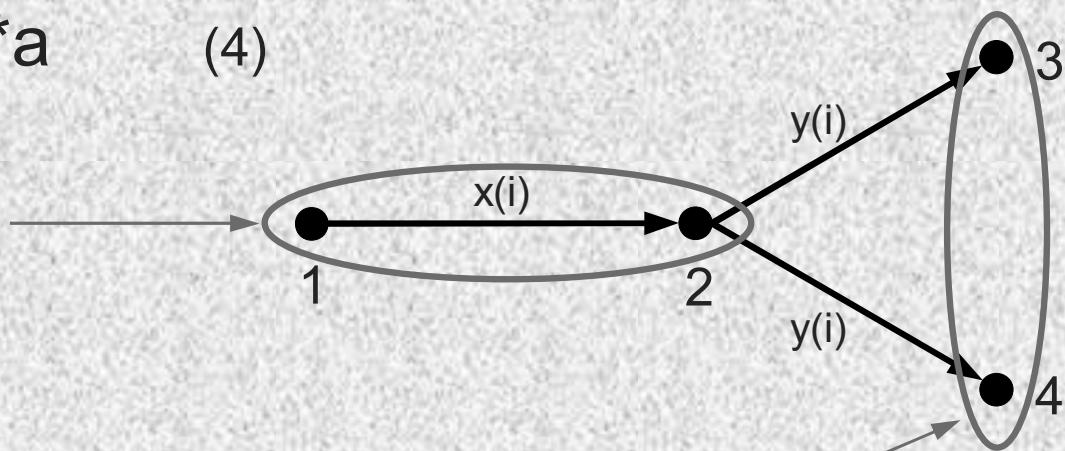
$$x(i) = a + b(i) \quad (1)$$

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

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

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

Исполнять только последовательно !

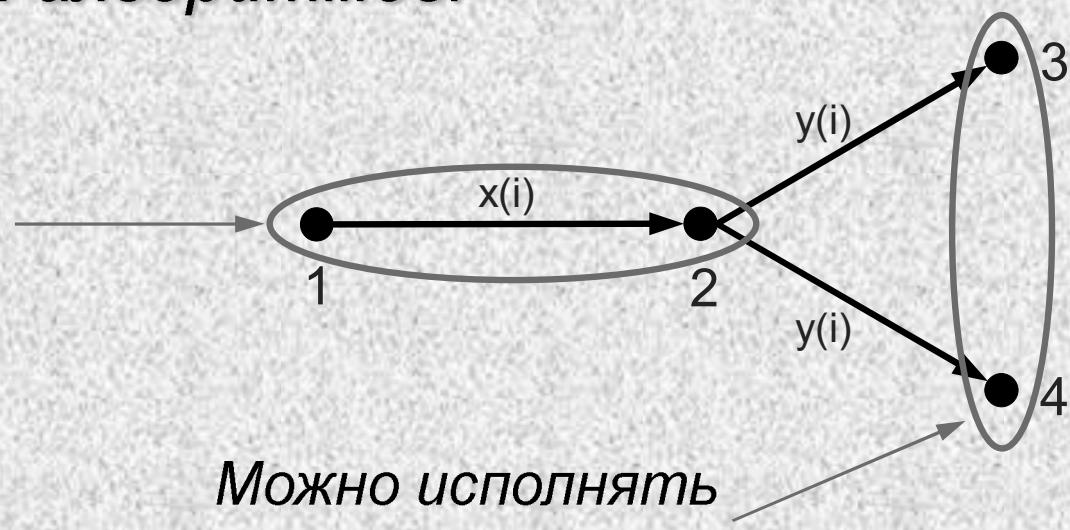


Можно исполнять параллельно !

Информационная структура программ и алгоритмов

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?

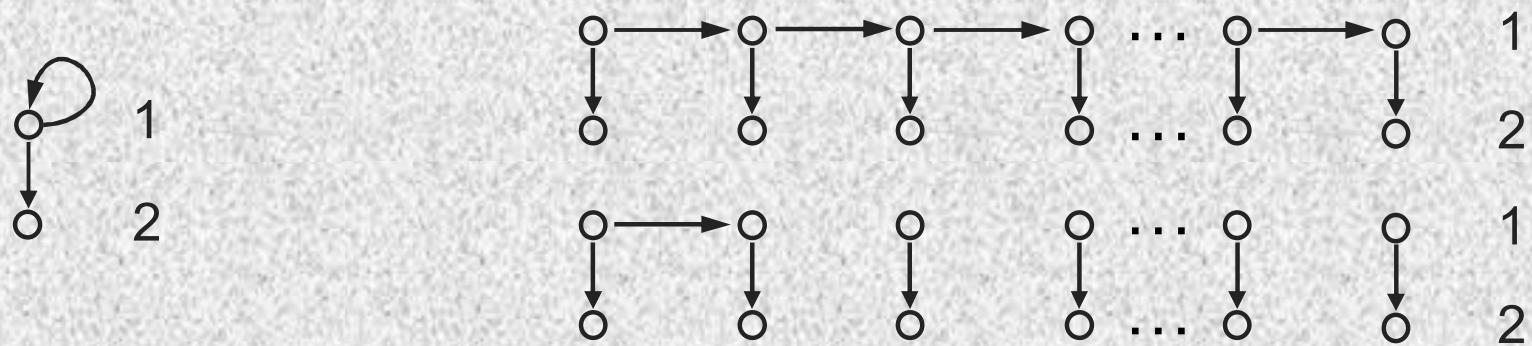
*Аргументы для выбора степени компактности
модели:*

- *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.

Каноническая ЯПФ и степень параллелизма

```
for( i = 0; i < n; ++i)
    for( j = 0; j < m; ++j)
        A[i][j] = A[i][j-1] + C[i][j]*x;
```

Чему, согласно закону Амдала, равно максимальное ускорение, которое можно получить при исполнении данного фрагмента на параллельной вычислительной системе?

Закон Амдала:

$$S \leq \frac{1}{\alpha + \frac{(1-\alpha)}{p}}$$

где:

*α – доля последовательных операций,
 p – число процессоров в системе.*

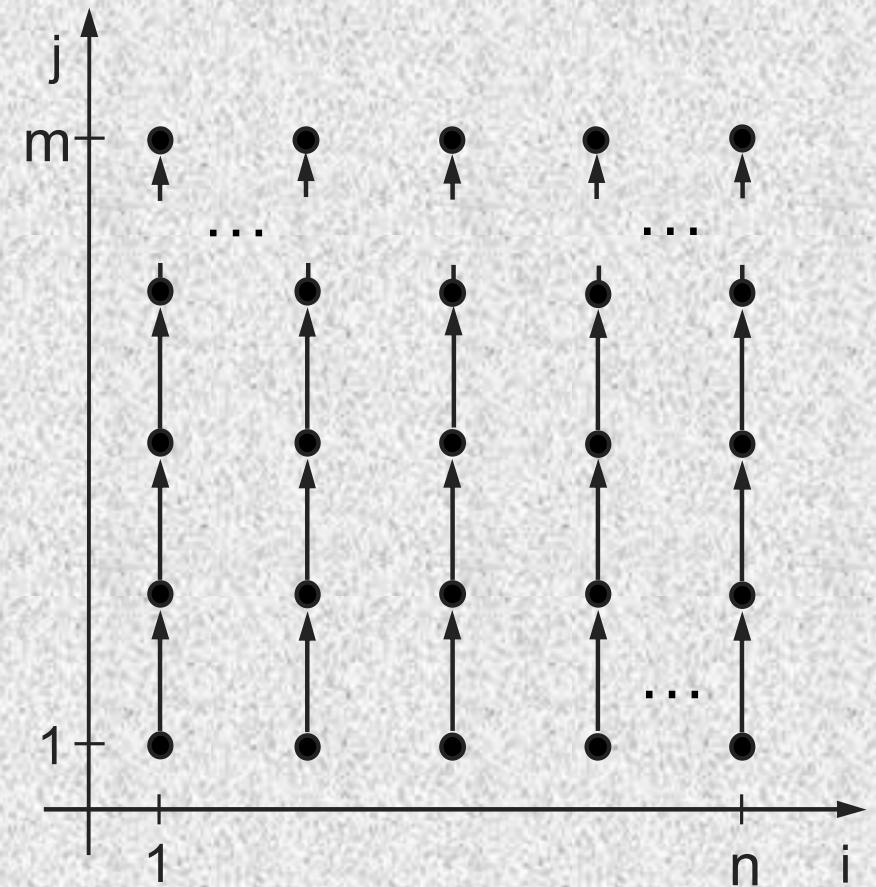
Каноническая ЯПФ и степень параллелизма

```
for( i = 0; i < n; ++i)
    for( j = 0; j < m; ++j)
        A[i][j] = A[i][j-1] + C[i][j]*x;
```

$$S \approx \frac{1}{\alpha}$$

$$\alpha = \frac{\text{Число последовательных операций}}{\text{Общее число операций}} = \frac{m}{n*m} = \frac{1}{n}$$

$$S \approx n$$



*Какого рода параллелизм
встречается в программах?*

Виды параллелизма в алгоритмах и программах



Конечный параллелизм определяется информационной независимостью некоторых фрагментов в тексте программы.

Массовый параллелизм определяется информационной независимостью итераций циклов программы.

Виды параллелизма в алгоритмах и программах

