Федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»

Институт компьютерных наук и технологий
Высшая школа искусственного интеллекта

В.Г. Пак

Структуры и алгоритмы компьютерной обработки данных

Слайды видеолекций
для студентов заочного бакалавриата направления подготовки высшего образования «Прикладная информатика»

Санкт-Петербургский политехнический университет Петра Великого
2022

Федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»

Институт компьютерных наук и технологий
Высшая школа искусственного интеллекта

ЛЕКЦИЯ №2

Нелинейные структуры данных. Графы

Санкт-Петербургский политехнический университет Петра Великого
2022

Содержание

III. Нелинейные структуры данных

§1. Графы

1.1. Основные определения

Определение. *Графом* называется пара $\mathrm{G} = \langle V, E \rangle$, где V - не более чем счётное множество, E – множество неупорядоченных пар элементов V, т.е. двухэлементных подмножеств V. Множество V называется *множеством вершин* (*узлов*), его элементы – *вершинами* (*узлами*). Множество E называется *множеством рёбер*, его элементы называются *рёбрами*. Число $n = |V|$ называется *порядком графа*.

Определение. Граф $\mathrm{G} = \langle V, E \rangle$ называется *ориентированным (орграфом)*, если элементы E – упорядоченные пары вершин, т.е. $E \subseteq \subseteq V^2$ - бинарное отношение на V. Рёбра орграфа называются *дугами*.

1.1. Основные определения теории графов

$$x = \{u, v\}$$

Вершины u, v *смежны.*
Ребро x *инцидентно* вершинам u, v.
Вершины u, v – концевые вершины ребра x.

$$x = \langle u, v \rangle$$

Вершины u, v *смежны.*
Дуга x *инцидентна* вершинам u, v.
Вершины u, v – концевые вершины дуги x.
Вершина u – начальная, v – конечная для дуги x.
Дуга x *исходит из* u и *входит в* v.

$$x = \{u, v\}, y = \{v, w\}$$

Рёбра (дуги) называются *смежными*, если они инцидентны общему узлу.

1.1. Основные определения теории графов

Определение. Граф называется *пустым*, если $V = \emptyset$, $E = \emptyset$ (обозначается Λ). Граф называется *полным*, если его любые два несовпадающих узла смежны. Полный граф порядка n обозначается K_n.

$x = \{u, u\}$ $(x = \langle u, u\rangle)$

↑

Петля

$x = \{u, v\}$, $y = \{u, v\}$ $(x = \langle u, v\rangle$, $y = \langle u, v\rangle)$

↑

Кратные рёбра (дуги)

$x = \langle u, v\rangle$, $y = \langle v, u\rangle$

↑

Противоположные дуги

1.1. Основные определения теории графов

Определение. *Звездой узла* v называется множество инцидентных v рёбер (дуг). Обозначается $\delta(v)$.

Определение. *Степенью узла* v называется число $deg(v) = |\delta(v)|$, т.е. количество инцидентных v рёбер (дуг).

Определение. *Полустепенью захода* $\delta_+(v)$ *узла* v орграфа называется число входящих в v дуг, *полустепенью исхода* $\delta_-(v)$ - число исходящих из v дуг.

Определение. Узел v называется *концевым узлом графа (орграфа)*, если $deg(v) = 1$, *изолированным*, если $deg(v) = 0$.

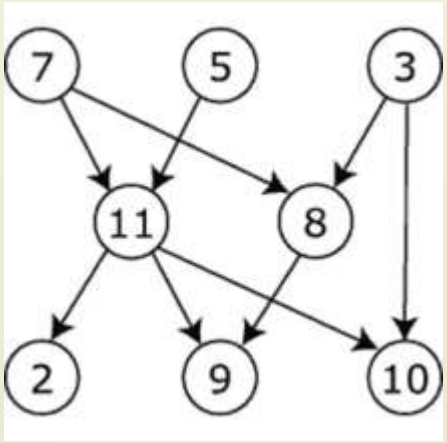
1.2. Способы задания графов

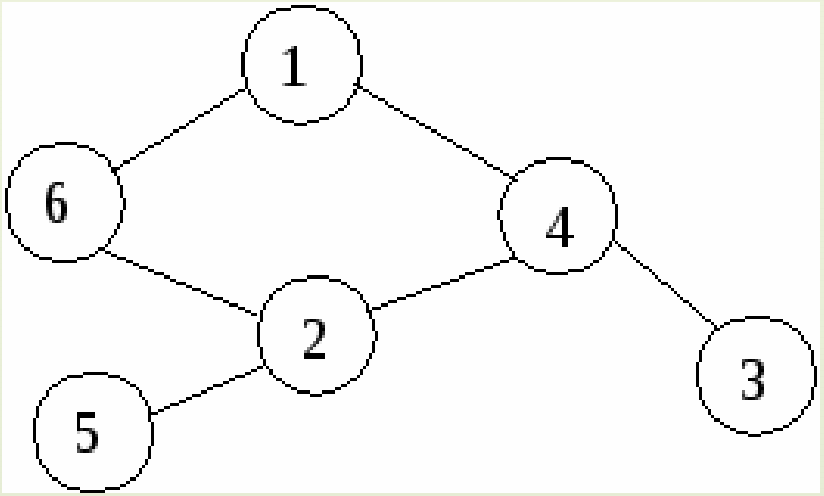
1. Теоретико-множественные:
   a) по определению, т.е. перечислением множеств V, E;
   b) списки инцидентности;

v_1: $x_1, x_2, \ldots$ ← Списки инцидентных рёбер

v_2: $y_1, y_2, \ldots$ ← (исходящих и входящих дуг)

2. Графический: *диаграмма;*





1.2. Способы задания графов

3. Матричные:

а) матрица смежности вершин $\Omega(G)$.

Для графа:

$$\Omega_{ij} = \begin{cases} 1, \text{если } v_i, v_j \text{ смежны}, \\ 0, \text{если } v_i, v_j \text{ несмежны}. \end{cases}$$

Если есть кратные рёбра, то Ω_{ij} равно их числу.

Для орграфа:

$$\Omega_{ij} = \begin{cases} 1, \text{если есть дуга } \langle v_i, v_j \rangle, \\ 0, \text{если такой дуги нет}. \end{cases}$$

Если есть кратные дуги, то Ω_{ij} равно их числу.

1.2. Способы задания графов

b) матрица инцидентности $\varepsilon(G)$.

Для графа:

$$\varepsilon_{ij} = \begin{cases} 1, \text{если узел } v_i \text{ инцидентен ребру } x_j, \\ 0, \text{если } v_i \text{ не инцидентен } x_j. \end{cases}$$

Для орграфа:

$$\varepsilon_{ij} = \begin{cases} 1, \text{если } v_i - \text{конечный узел дуги } x_j, \\ -1, \text{если } v_i - \text{начальный узел } x_j, \\ 0, \text{если } v_i \text{ и } x_j \text{ неинцидентны.} \end{cases}$$

Для петель применяют какой-либо особый символ, например, ± 1.

1.3. Представление графов структурами данных

1.3.1. Матрицы и списки

Наиболее общей структурой данных для представления графов является матрица смежности булевого или символьного типа. Возможно также представление матрицей инцидентности.

Недостаток матричных структур: требуют $\Omega(n^2)$ или $\Omega(nm)$ объёма памяти (n - число узлов, m – число рёбер (дуг)). Поэтому для чтения матрицы или нахождения в ней нужного элемента требуется время порядка $O(n^2)$ или $O(nm)$, что не позволяет создавать алгоритмы с временем $O(n)$.

1.3.1. Матрицы и списки

Матричные структуры относятся к *статическим структурам данных.*

Определение. *Статическими структурами данных* называются такие структуры, память под которые выделяется во время компиляции и сохраняется в течение всей работы программы.

Примеры: массивы, строки.

Линейные структуры данных являются статическими (но не обязательно).

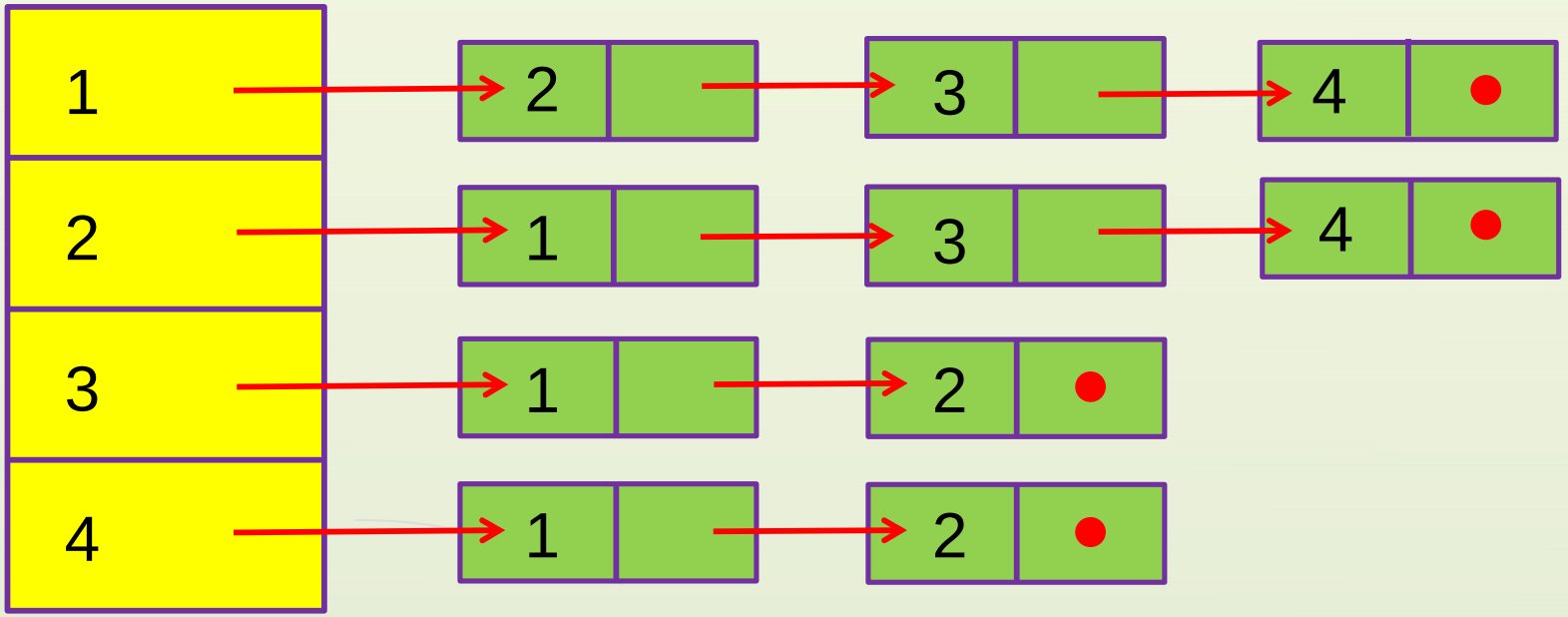
Определение. *Динамическими структурами данных* называются такие структуры, память под которые распределяется во время работы программы.

Из динамических структур в программах чаще всего используются *линейные списки, стеки, очереди и бинарные деревья.*

Нелинейные структуры данных, как правило, являются динамическими. Работа с динамическими величинами связана с использованием указателей — данных ссылочного типа.

1.3.1. Матрицы и списки

Списком смежности вершины v графа называется определённым образом упорядоченный список смежных с v вершин.

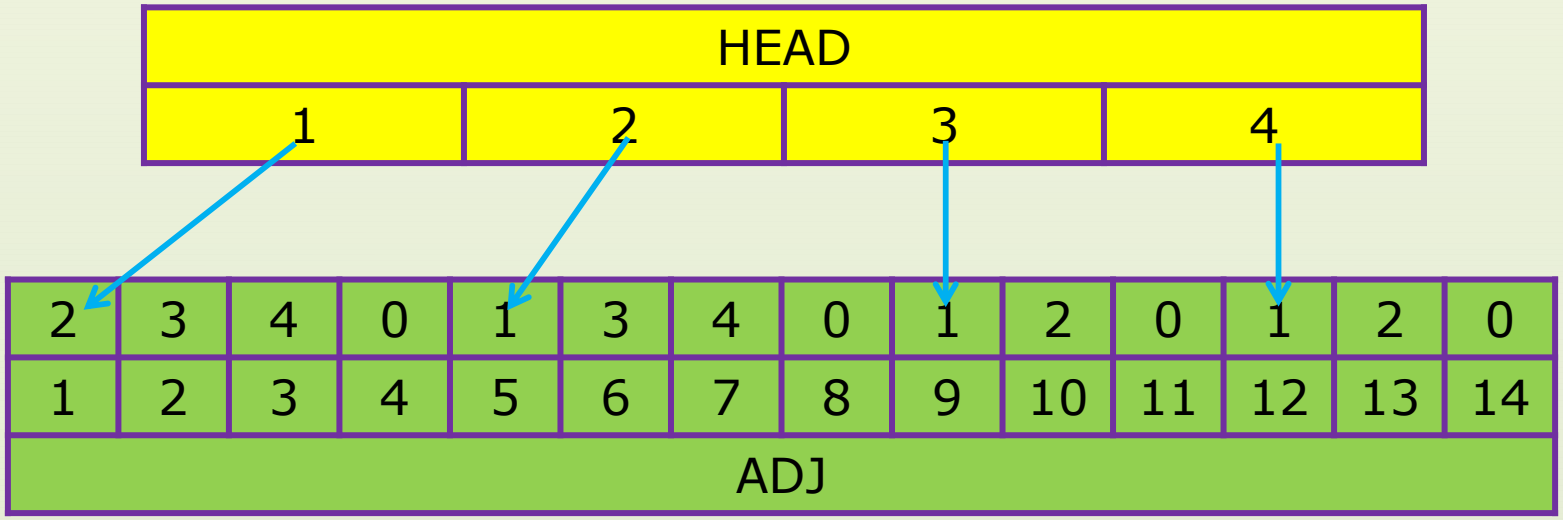


1.3.1. Матрицы и списки

Списком смежности узла v орграфа называется определённым образом упорядоченный список узлов, достижимых из v по дуге.

Для вставки и удаления элементов в списках смежности необходимо иметь массив указателей на ячейки заголовков списков смежности.

Но если известно, что граф не будет подвергаться изменениям (или они будут незначительны), то предпочтительнее другая структура списка смежности:



1.3.2. Структуры Вирта

Для представления орграфов используется предложенная Н.Виртом динамическая *структура* со многими связями. Вершины орграфа представляются связанным списком заголовочных узлов, каждый из которых содержит четыре поля. Если указатель P указывает на заголовочный узел, представляющий вершину v графа, то:

1. Поле $P.Key$ содержит идентификатор вершины v.
2. Поле $P.Count$ содержит количество «предшествующих» v вершин, т.е. тех, из которых v достижима по дуге.
3. Поле $P.Next$ содержит указатель на заголовочный узел, представляющий следующий узел орграфа (если таковой есть) в списке заголовочных узлов.

1.3.2. Структуры Вирта

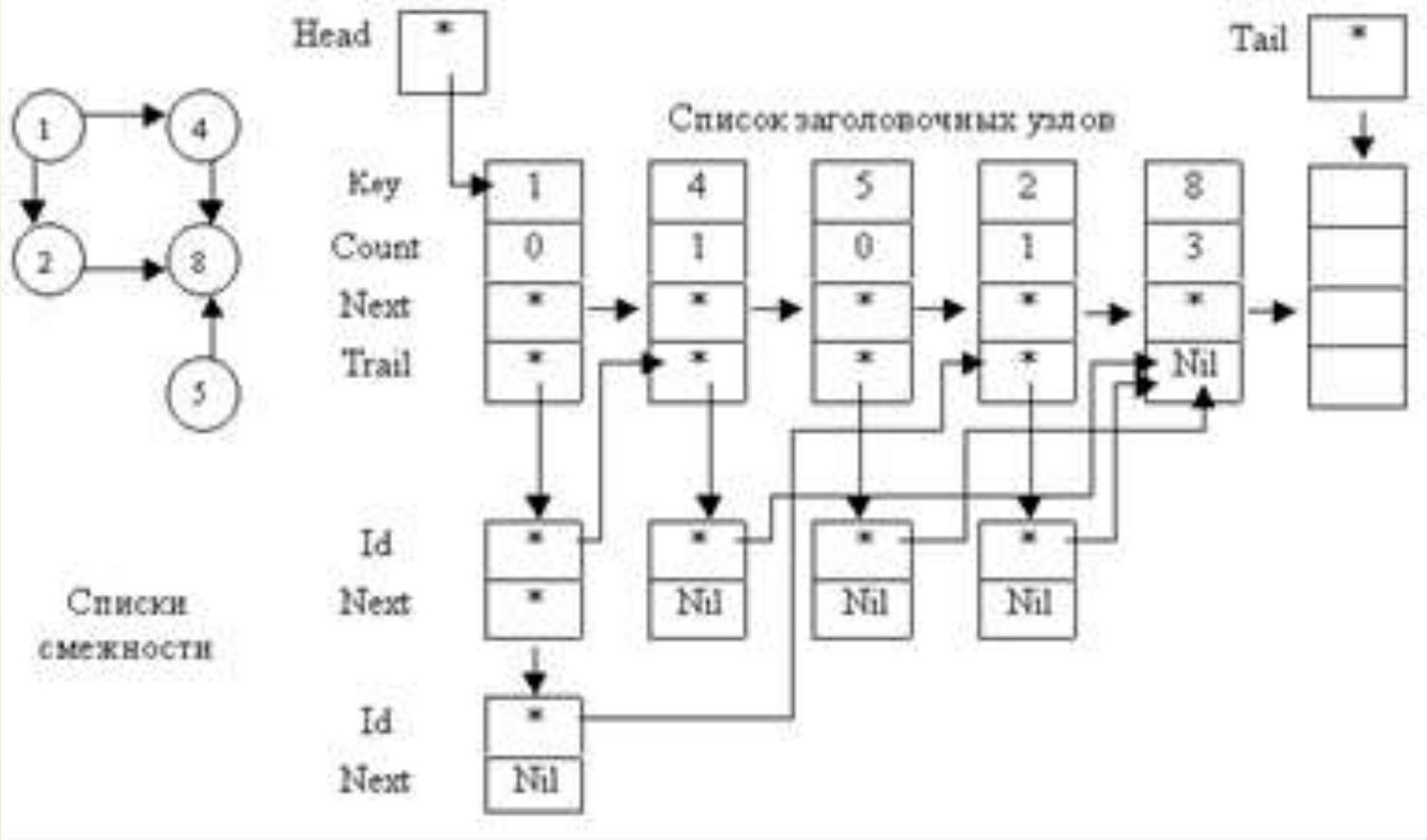
4. Каждый заголовочный узел является заглавным звеном списка узлов второго типа, называемых *дуговыми узлами*. Такой список называется *списком смежности*. Поле $P.Trail$ указывает на список смежности, представляющий выходящие из v дуги.

Каждый узел списка смежности содержит два поля: Id и $Next$. Если Q указывает на дуговой узел, представляющий дугу $\langle u, w \rangle$, то:

1. Поле $Q.Id$ указывает на заголовочный узел, представляющий вершину w орграфа.
2. Поле $Q.Next$ указывает на дуговой узел, представляющий следующую дугу, выходящую из вершины u (если таковая есть).

Каждый дуговой узел содержится в единственном списке смежности, представляющем все дуги, выходящие из данной вершины.

1.3.2. Структуры Вирта



1.3.2. Структуры Вирта

Модифицированная структура Вирта используется в тех случаях, когда для быстрой работы с орграфом желательно иметь списки «предшествующих» узлов. Для этого к структуре Вирта добавляются такие списки.

Если указатель P указывает на заголовочный узел, представляющий вершину v графа, то:

1. Поле $P.Key$ содержит идентификатор вершины v.
2. Поле $P.Count$ содержит количество «предшествующих» v вершин, т.е. тех, из которых v достижима по дуге.
3. Поле $P.Count1$ содержит количество «следующих» за v вершин, т.е. тех, которые достижимы из v по дуге.
4. Поле $P.Next$ содержит указатель на заголовочный узел, представляющий следующий узел орграфа (если таковой есть) в списке заголовочных узлов.
5. Каждый заголовочный узел является заглавным звеном двух списков узлов второго типа, по-прежнему называемых *дуговыми узлами*. Такие списки также называются *списками смежности*.

1.3.2. Структуры Вирта

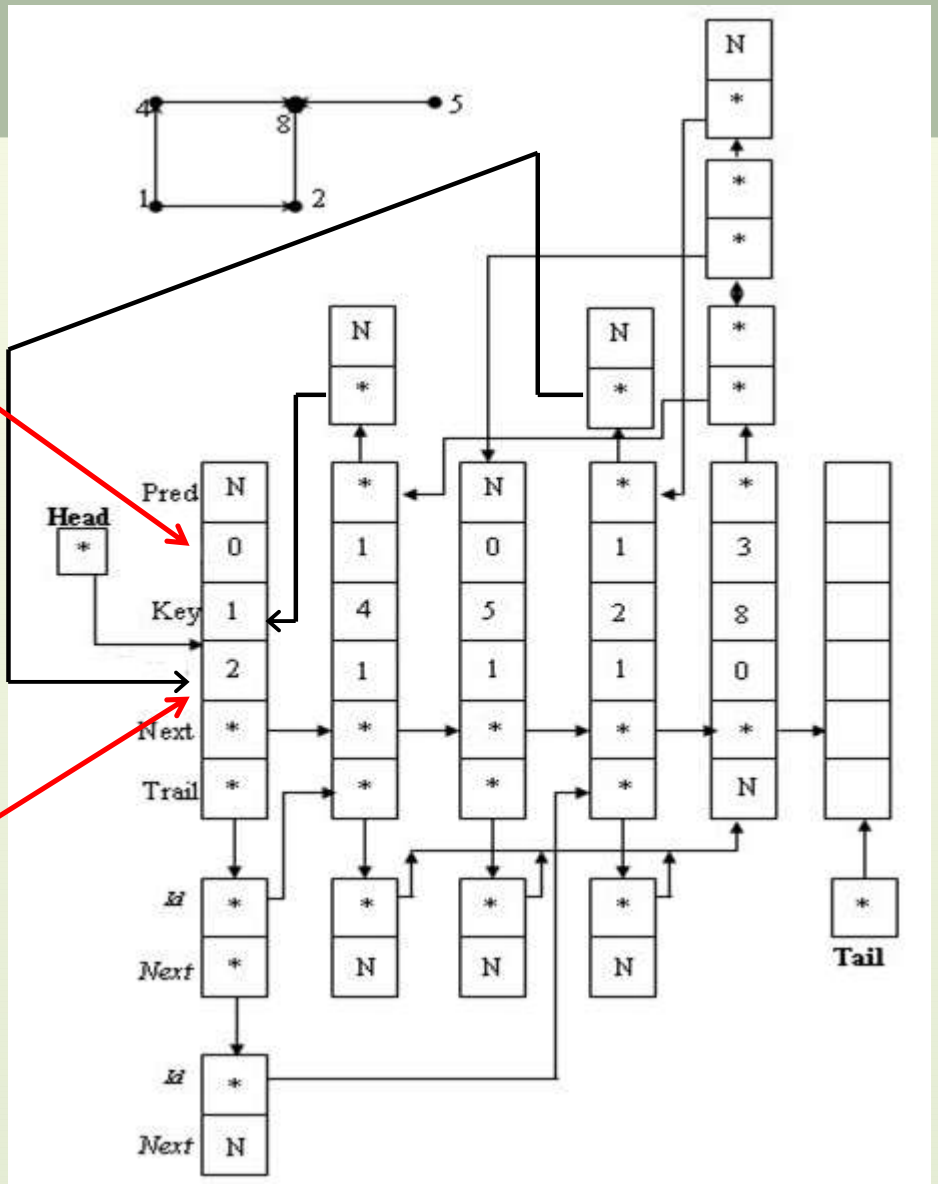
Каждый узел списков смежности представляет дугу орграфа. Поле $P.Trail$ указывает на первый список смежности, представляющий выходящие из v дуги. Он такой же, как в структуре Вирта.

Второй список смежности представляет собой список узлов, представляющих «предшественники» v. Поле $P.Pred$ содержит указатель на этот список.

Каждый узел второго списка смежности также содержит два поля: Id и $Next$. Если Q указывает на дуговой узел, представляющий дугу $\langle u, w \rangle$, то:

1. Поле $Q.Id$ указывает на заголовочный узел, представляющий узел u орграфа.
2. Поле $Q.Next$ указывает на дуговой узел, представляющий следующую дугу, входящую в узел w (если таковая есть).

1.3.2. Структуры Вирта



1.3.3. АТД графов

Наиболее общие операторы АТД для графов:

- чтение меток (идентификаторов) узлов и рёбер (дуг);
- вставка и удаление узлов и рёбер (дуг);
- перемещение по последовательностям рёбер (дуг).

Для просмотра множества смежных с v узлов (достижимых из v по дуге) необходимо реализовать операторы:

1. FIRST(v) возвращает индекс первого узла, смежного с v (достижимого из v по дуге).
2. NEXT(v, i) возвращает индекс смежного с v (достижимого из v по дуге) узла, следующего за индексом i. Если i – индекс последнего такого v узла, то возвращается пустой индекс (nil, null, Λ).
3. VERTEX(v, i) возвращает узел с индексом i из множества смежных с v узлов (достижимых из v по дуге).

1.3.3. АТД графов

Примеры:

v – индекс узла

```
Function FIRST(v)
Begin
    for i:=1 to n do
        if A[v, i] then
            return (i)
    return (0)
End
```

A – логическая матрица смежности узлов

```
Function NEXT(v, i)
Begin
    for j:=i+1 to n do
        if A[v, j] then
            return (j)
    return (0)
End
```

Последовательный просмотр узлов, смежных с v:

```
i:= FIRST(v)
While i<>0 do begin
    w:=VERTEX(v, i)
    Действия с узлом w
    NEXT(v, i)
End
```

1.4. Обходы графов

Постановка задачи.

Дано: граф (орграф) G.

Требуется систематически обойти (пометить) все узлы G.

1.4.1. Алгоритм поиска в глубину в орграфе

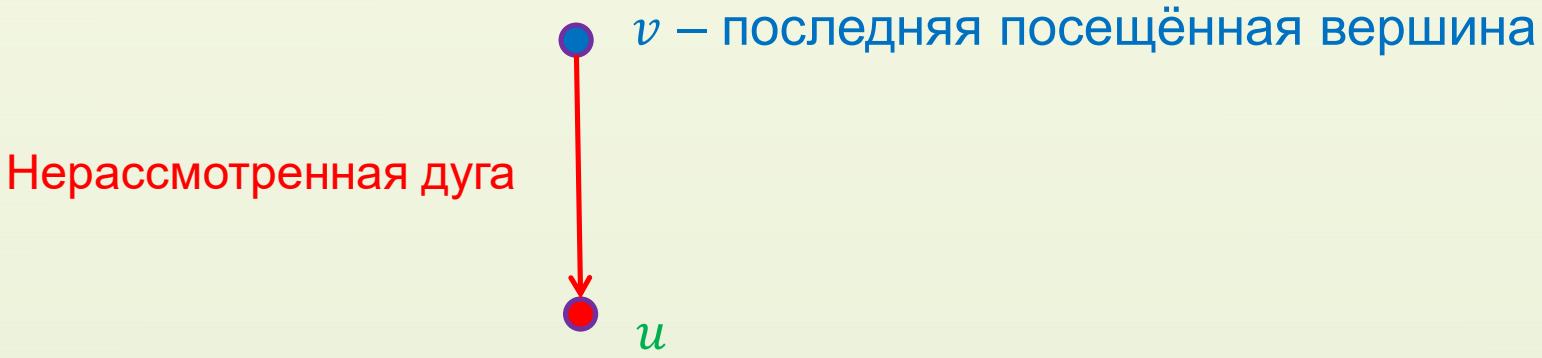
Словесное описание алгоритма.

Сначала все вершины помечаются как непосещённые (–). Выбирается некоторая начальная непосещённая вершина v и помечается как посещённая (+). Далее к каждой смежной с v непосещённой вершине рекурсивно применяется процедура поиска в глубину. Когда все вершины, которые можно достичь из v, посещены, алгоритм применительно к v завершается.

Если остались непосещённые вершины, то любая из них выбирается как начальная и к ней применяется алгоритм. Продолжается до тех пор, пока не будут посещены все вершины.

1.4.1. Алгоритм поиска в глубину в орграфе

Поиск идёт вглубь



Если u помечена $(+)$, то переход к следующей нерассмотренной дуге, если u помечена $(-)$, то она помечается $(+)$, алгоритм переходит в u, и поиск продолжается от u.

После рассмотрения всех путей от u происходит возврат в v, т.е. первую вершину, из которой была достигнута u.

Затем продолжается выбор нерассмотренных дуг, исходящих из v.

И так до тех пор, пока не будут исчерпаны все исходящие из v дуги.

1.4.1. Алгоритм поиска в глубину в орграфе

Псевдокод: Структура данных – список смежности вершин L

```
Procedure Поиск в глубину
For v:=1 to n do
      M[v]:='-'          M – массив меток вершин
For v:=1 to n do
      if M[v]='-' then
         search(v)       процедура поиска в глубину от вершины v
End Поиск в глубину
Procedure search(v)
M[v]:='+'
For u∈L[v] do
if M[u]='-' then
         search(u)
End search
```

1.4.1. Алгоритм поиска в глубину в орграфе

Алгоритм имеет вычислительную сложность $O(m), m \geq$ $\geq n$.

1.4.2. Глубинный остовный лес

Определение. *Маршрутом* в графе (орграфе) называется последовательность вершин и рёбер (дуг) вида $v_0 x_1 v_1 x_2 \dots x_n v_n$, где $v_i \in$ $\in V$, $x_i \in E$, ребро (дуга) x_i инцидентно v_{i-1}, v_i. Маршрут от u к v будем называть (u, v)-маршрутом. Если $v_0 \neq v_n$, маршрут называется *открытым*, если $v_0 = v_n$ - *замкнутым*.

Определение. Маршрут называется *простым*, если все его рёбра (дуги) различны, *цепью*, если все его вершины (кроме, может быть, крайних) различны.

Определение. *Циклом* называется замкнутая цепь.

Определение. Граф (орграф) без циклов называется *лесом.*

Определение. Граф (орграф) называется *связным*, если для любых его несовпадающих вершин u, v существует (u, v)-цепь.

Определение. Связный граф (орграф) без циклов называется *деревом.*

1.4.2. Глубинный остовный лес

Дуги, ведущие к непосещённым вершинам в процессе поиска в глубину, называются *дугами дерева* и формируют *глубинный остовный лес*, построенный методом поиска в глубину.

Другие типы дуг, определяемые в процессе обхода в глубину:

1) *обратные дуги*, ведущие от потомков к предкам (т.е. от позже посещённых к ранее посещённым при поиске из одной начальной вершины);
2) *прямые дуги*, ведущие от предков к истинным потомкам (т.е. от ранее посещённых к позже посещённым при поиске из одной начальной вершины), но не являющиеся дугами дерева;
3) *поперечные дуги*, соединяющие узлы, не являющиеся ни предками, ни потомками друг друга.

Глубинная нумерация – нумерация в порядке посещения при обходе вершин:

```
Number[v]:=count
count:=count+1
```

1.4.2. Глубинный остовный лес

Как различить эти четыре типа дуг?

- Дуги дерева ведут к ранее не посещённым вершинам.
- Прямые дуги ведут от вершин с более низкими номерами к вершинам с более высокими номерами.
- Обратные дуги ведут от вершин с более высокими номерами к вершинам с более низкими номерами.
- Поперечные дуги также ведут от вершин с более высокими номерами к вершинам с более низкими номерами.

1.4.3. Алгоритм поиска в глубину в графе

Алгоритм такой же, как для орграфа.

Определение. *Подграфом графа* $G = \langle V, X \rangle$ называется граф $G' = \langle V', X' \rangle$, в котором $V' \subseteq V$, $X' \subseteq X$.

Определение. Пусть $V' \subseteq V$, $X(V')$ - подмножество рёбер X, обе концевые вершины которых принадлежат V'. Тогда подграф $G' = \langle V', X(V') \rangle$ называется *подграфом графа* $G = \langle V, X \rangle$, *порождённым множеством вершин* V'.

Определение. Пусть на множестве вершин V графа (орграфа) G определено отношение эквивалентности R: $uRv \Leftrightarrow$ в G существует (u, v)-цепь. Классы эквивалентности V по R $V_1, \dots, V_k$ порождают подграфы $G_1 = \langle V_1, X(V_1) \rangle$, $G_2 = \langle V_2, X(V_2) \rangle$, … , $G_k = \langle V_k, X(V_k) \rangle$, которые называются *компонентами связности графа* G. Если $k = 1$, граф называется связным.

1.4.3. Алгоритм поиска в глубину в графе

Свойства глубинного остовного леса, построенного при обходе графа в глубину:

1. Каждое дерево глубинного остовного леса соответствует одной компоненте связности исходного графа.
2. Выделяются два типа рёбер исходного графа:
   a. Рёбра дерева $\{u, v\}$, где узел u уже достигнут при обходе, а v ещё не посещался.
   b. Обратные рёбра $\{u, v\}$, где узел u уже достигнут при обходе, а v уже посещался (естественно, ребро, по которому достигнут узел u, не относится к обратным).

1.4.4. Алгоритм поиска в ширину в графе

Идея алгоритма.

Во время обхода любой вершины v просматриваются все смежные с v вершины.

Словесное описание алгоритма.

Сначала все вершины помечаются как непосещённые (−). Выбирается некоторая начальная непосещённая вершина v и помечается как посещённая (+). Все смежные с v вершины помечаются как посещённые. Затем процедура поиска в ширину применяется для каждой смежной с v вершины. Далее происходит переход в одну из смежных с v вершин и применение к ней процедуры, затем переход в следующую смежную вершину и т.д.

Если остались непосещённые вершины, то любая из них выбирается как начальная и к ней применяется алгоритм.

Продолжается до тех пор, пока не будут посещены все вершины.

1.4.4. Алгоритм поиска в ширину в графе

При обходе также строится *остовный лес*. Его свойства:

1. Строение и нумерация узлов такие же, как у глубинного остовного леса.
2. Рёбра исходного графа делятся на:
   a) рёбра дерева, входящие в остовный лес (определяются так же, как в глубинном остовном лесе);
   b) поперечные рёбра, не входящие в остовный лес (определяются так же, как в алгоритме поиска в глубину; соединяют узлы, не связанные отношением наследования).

1.4.4. Алгоритм поиска в ширину в графе

Псевдокод: Структура данных – список смежности вершин L

Procedure Поиск в ширину

T:=∅ T – массив остовного леса. Сначала пустой, потом содержит рёбра деревьев

Q:=∅ Q- очередь узлов, содержит посещённые узлы

For v∈ V do

M[v]:='-' M – массив меток узлов

For v∈ V do

while M[v]='-' do

search(v) процедура поиска в ширину от начальной вершины v

End Поиск в ширину

1.4.4. Алгоритм поиска в ширину в графе

```
Procedure search (v)
M[v]:='+'
Q:=Q∪{v}    посещённый узел помещается в конец очереди
while not EMPTY(Q) do begin проверка Q=∅ (оператор АТД множеств)
                        x:=FIRST(Q) извлечение первого узла из
                                    очереди
                        For y∈L[x] do
                            if M[y]='-' then begin
                               M[y]:='+'
                               Q:=Q∪{y}   посещённый узел помещается в
                                          конец очереди
                               T:=T∪{{x, y}} ребро {x, y} помещается в
                                             массив T рёбер деревьев
                            End
End
End search
```

1.4.4. Алгоритм поиска в ширину в графе

Алгоритм имеет вычислительную сложность $O(m), m \geq$ $\geq n$.

Применение алгоритма: нахождение связных компонент (деревья в остовном лесу), поиск циклов (поперечные рёбра в лесу).