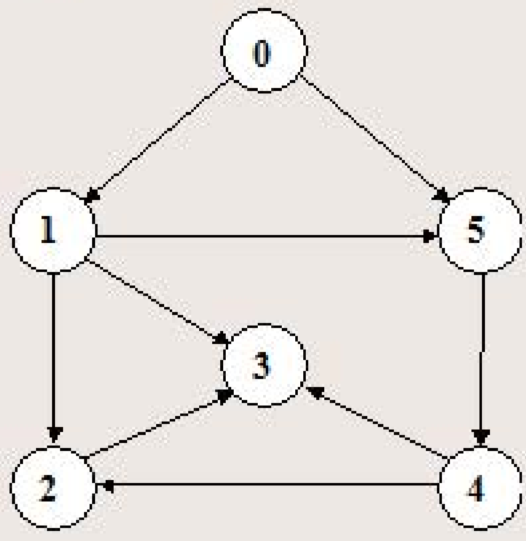
1. Понятие графа…

Определение 5.2. *Ориентированный граф* **G** представляет собой набор

$$G = (V, R)$$

где

- $V = \{ v_1, v_2, \ldots, v_n \}$ есть множество *вершин* графа,
- $R = \{ (i,j): i,j \in V \}$ есть множество *дуг* (*ребер*) графа



5.3. Представление графовых моделей на ЭВМ

1. Понятие графа

Понятия теории графов

- Вершина-родитель (предок) и вершина-потомок
- Входящие/исходящие дуги
- Путь, длина пути, размер (глубина) графа

Дополнительные понятия

- Размеченный граф (вершины имеют метки-значения)
- Взвешенный граф (дугам графа приписывается вес)
- Длина пути с учетом весов дуг
- Кратчайший путь

2. Выбор структуры хранения графа

- Использование матрицы смежности (для неориентированных графов можно применить верхние треугольные матрицы)
- Представление наборов исходящих дуг в виде множест смежных вершин
- Использование списков исходящих дуг для вершин графа

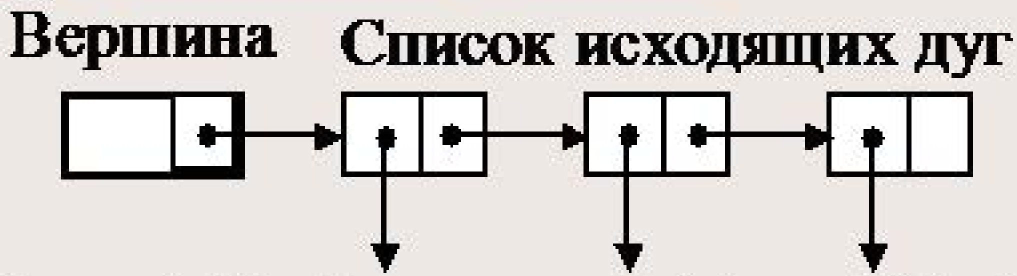
Оценка походов:

- объем используемый памяти
- эффективность выполнения основных операций обработки (вставка/удаление вершин/вершин, поиск вершины/дуги, реализация обходов)

↳Используем для учебного изучения *структуру хранения на основе списков исходящих дуг*

3. Реализация структуры хранения графа...

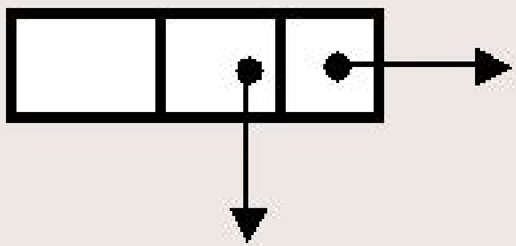
Структура хранения вершины




```
class TGraphNode : public TDatValue {
 protected:
  string    Name;  // имя вершины графа
  int       Value; // метка (значение)
  int       Index; // индекс (номер)
  TDatList Links; // список исходящих дуг
 public:
  // вставка/исключение дуг
  void InsLink(TGraphNode *pGn, int val);
  void DelLink(string NodeName);
};
```

3. Реализация структуры хранения графа...

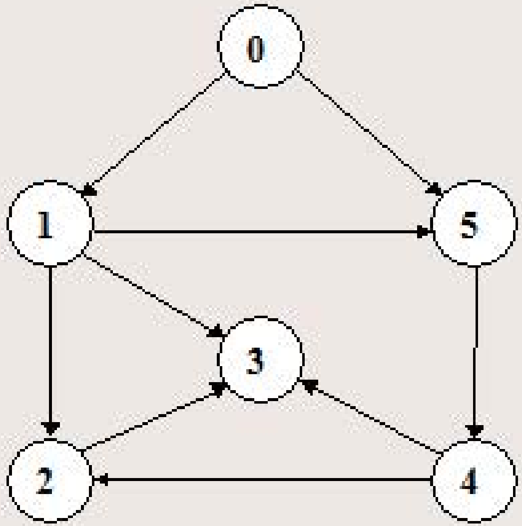
Структура хранения дуги

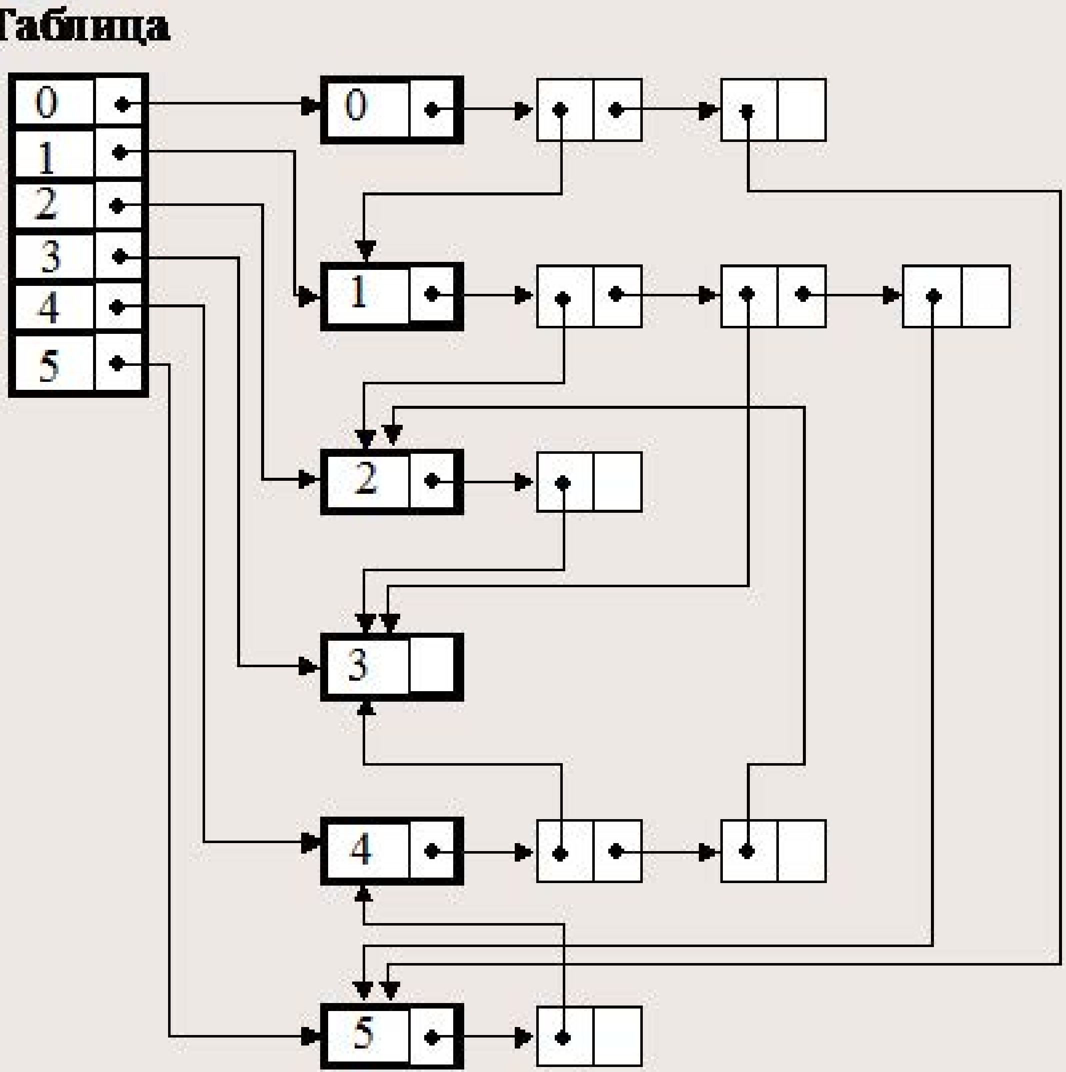


```
class TGraphLink : public TDatValue {
 protected:
  int Value; // вес дуги
  // начальная и конечная вершины
  PTGraphNode pFirst, pLast;
};
```

3. Реализация структуры хранения графа…

Структура хранения дуги





3. Реализация структуры хранения графа

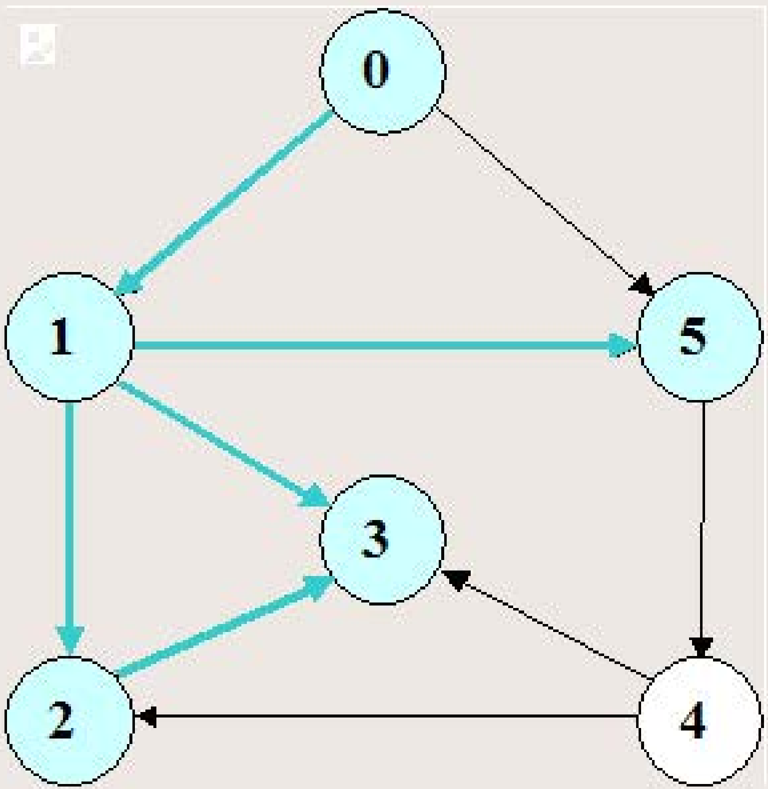
Структура хранения графа

```
class TGraph : public TDatValue {
 protected:
  int NodeNum;                // количество вершин
  int LinkNum;                // количество дуг
  TTreeTable Nodes;           // множество вершин графа
 public:
  TGraph ();
  int GetNodeNum (void) { return NodeNum; }
  int GetLinkNum (void) { return LinkNum; }
  // встава верши и дуг графа
  void InsNode ( string nm, int val=0 );
  void InsLink ( string fn, string ln, int val=1);
};
```

4. Алгоритмы обхода…

Поиск в глубину (depth-first search)

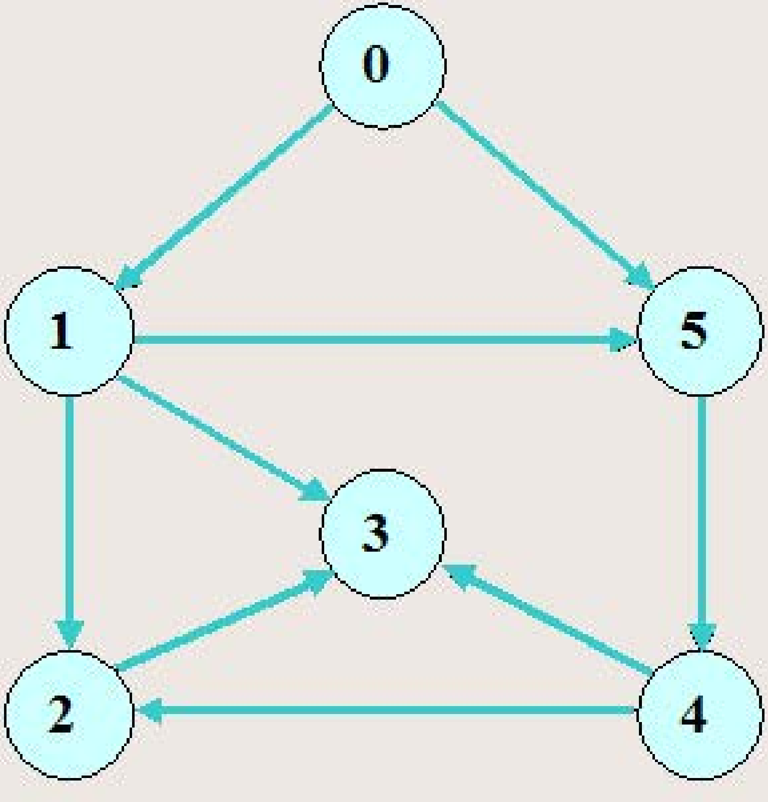
Обработка очередной вершины графа продолжается рекурсивной процедурой обработки смежных вершин



4. Алгоритмы обхода…

Поиск в ширину (breadts-first search)

Для каждой вершины сначала выполняется обработка непосредственно смежных вершин



4. Алгоритмы обхода…

- Для исключения циклов и запоминания набора уже обработанных вершин можно использовать множество TSet
- Для запоминания набора достигнутых, но еще не обработанных вершин, можно использовать структуры:
  - Стек – для алгоритма поиска в глубину
  - Очередь – для алгоритма поиска в ширину

4. Алгоритмы обхода (итератор)…

```
TSearchMode Mode;          // способ обхода
PTGraphNode pCurrNode; // текущая вершина
// достигнутые, но не обработанные вершины
TDataRoot *pStream;
// множество вершин, достигнутых в ходе обхода
TBitField *pFound;
```

4. Алгоритмы обхода (итератор)…

Инициализация (Reset)

- Инициализация структур
- Вставка первой вершины в поток pStream и ее отметка в множестве pFound
- Выполнение метода GoNext

4. Алгоритмы обхода (итератор)

Переход к следующей вершине графа (GoNext)

- Получить вершину из потока pStream
- Поместить смежные вершины, если они еще не достигнуты, в поток pStream
- Отметить смежные вершины в множестве pFound

Пример: программа, приложение

5. Поиск кратчайшего пути…

Алгоритм Дейкстры (Dijkstra)

- В ходе работы алгоритма определяются расстояния кратчайших путей от заданной вершины до всех достижимых вершин графа
- Для запоминания длин найденных путей используется массив pDist (если до вершины нет найденного пути, соответствующее значение массива устанавливается в ∞)
- Вершины, уже вошедшие в построенные кратчайшие пути, фиксируются при помощи множества pFound

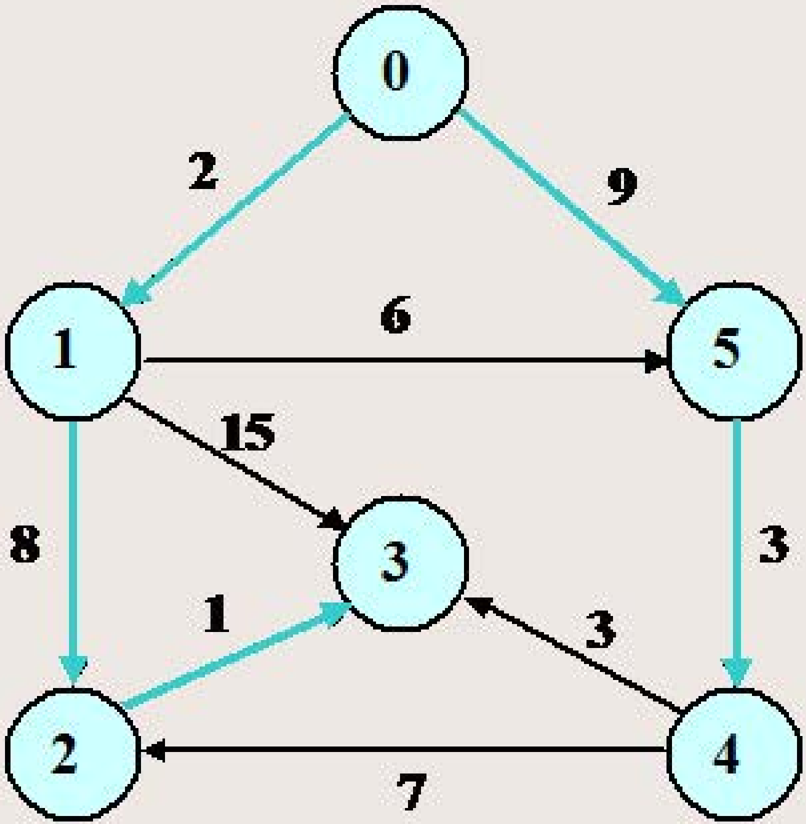
5. Поиск кратчайшего пути…

Алгоритм Дейкстры (Dijkstra)

- На каждой итерации алгоритма строится кратчайший путь до вершины, до которой этот путь еще не был определен (т.е. до вершины, не принадлежащей множеству pFound); данная вершина определяется минимальным расстоянием в массиве pDist среди вершин, до которых еще не найден кратчайший путь
- Найденная вершина фиксируется в множестве pFound; далее из этой вершина строятся пути до вершин, не вошедших в pFound и, если этот длина этих путей меньше, чем значения в массиве pDist, значения длин путей в pDist корректируется

5. Поиск кратчайшего пути…

Алгоритм Дейкстры (Dijkstra)



0	1	2	3	4	5
0	2	10	11	11	8

5. Поиск кратчайшего пути…

Алгоритм Дейкстры (Dijkstra)

Для восстановления найденных кратчайших путей для каждой вершины определим массив pPred, в котором будем запоминать для вершин номера предшествующих по кратчайшему пути вершин

Пример: программа, приложение

5. Поиск кратчайшего пути…

Обоснование алгоритма Дейкстры

Теорема 5.2. Пусть $G=(V,E)$ – взвешенный ориентированный граф с неотрицательной весовой функцией $w:E\to R$ и исходной вершиной s. Тогда после применения алгоритма Дейкстры

$$\forall\, v\in V \Rightarrow d[v] = \delta_{min}(s,v)$$

Доказательство,

Покажем, что на любой итерации цикла

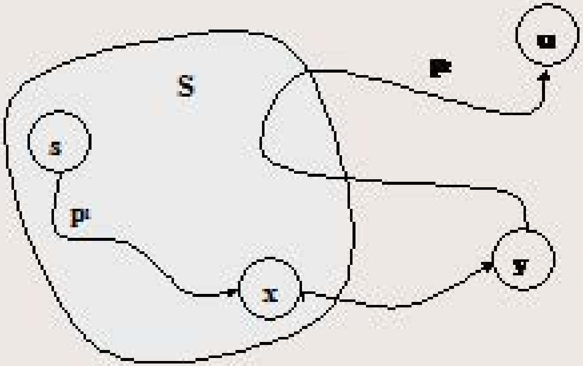
- Для вершин $v\in S$ (**S** – множество уже обработанных вершин) уже выполняется утверждение теоремы, т.е. $d[v] = \delta_{min}(s,v)$, и, кроме того, к этим вершинам существуют кратчайшие пути только из вершин **S**.
- Для вершин $v\in Q=V\setminus S$ значение **d[v]** равно наименьшему весу пути из **s** в **v**, если учитывать только те *особые пути*, в которых все вершины (кроме последней) лежат в **S** (если таких путей нет, то $d[v] = \infty$)

5. Поиск кратчайшего пути…

Обоснование алгоритма Дейкстры

Для начальной итерации, когда в **S** только одна вершина **s**, свойства 1 и 2 справедливы. Покажем выполнимость этих утверждений для любой итерации.

Пусть **u** есть вершина из **Q=V\S** с минимальным значением **d[u]** Покажем, что особый путь из **s** в **u** веса **d[u]** является кратчайшим для **u**.



Пусть существует другой путь из **s** в **u**. Рассмотрим первую вершину **y**$\notin$**S**. По свойству 1 и предположения индукции вес пути из **s** в **y** не меньше **d[y]** и, кроме того, **d[y]≥d[u]** (условие выбора **u**). Это означает, что вершину **u** можно добавить в **S** и свойство 1 не нарушится.

5. Поиск кратчайшего пути…

Обоснование алгоритма Дейкстры

Расширение множества $\mathbf{S'=S\cup\{u\}}$ сопровождается уточнением расстояний до вершин множества $\mathbf{Q}$

$$\forall v \in Q \Rightarrow d'[v] = \min(d[v], d[u] + w(u,v))$$

Оба значения, из которых определяется величина $\mathbf{d'[v]}$, есть веса особых путей из $\mathbf{s}$ в $\mathbf{v}$. Покажем, что вес любого особого пути из $\mathbf{s}$ в $\mathbf{u}$ не меньше $\mathbf{d'[v]}$. На самом деле, пусть $\mathbf{x \in S'}$ есть вершина, предшествующая $\mathbf{v}$. Тогда либо $\mathbf{x \in S}$ и тогда вес пути не меньше $\mathbf{d'[v]}$ по свойству 1, либо $\mathbf{x=u}$ и необходимое условие следует из правила вычисления $\mathbf{d'[v]}$.

Итак, условия 1 и 2 выполняются на всех итерациях цикла, в том числе, и по завершении работы алгоритма Дейкстры.

5. Поиск кратчайшего пути…

Оценка сложности алгоритма Дейкстры

- Количество операций пересчета весов путей до вершин графа пропорционально числу вершин **N**
- Количество итераций алгоритма также совпадает с **N**

$$\Rightarrow T_{max} = O(N^2)$$

Если для представления вектора весов **d** использовать приоритетную очередь на основе двоичной кучи, то можно получить оценку

$$\Rightarrow T_{max} = O(M \log N),$$

где **M** есть общее количество вершин в графе.

Заключение

- Графы является одной из самых важных моделей описания сложных структур
- Для представления графов на ЭВМ можно использовать матрицы смежности или списковые структуры хранения
- Для обхода графов основными алгоритмами являются поиски в глубину или в ширину
- Поиск кратчайших путей в графе может быть выполнен при помощи алгоритма Дейкстры