**Лекция 10**

**ОПТИМИЗАЦИОННЫЕ АЛГОРИТМЫ НА ГРАФАХ**

**Алгоритм поиска в ширину**

**(BFS,** breadth-first search**)**

**BFS(G, s)**

Идея поиска в ширину состоит в том, чтобы посещать вершины в порядке их удаленности от некоторой заранее выбранной или указанной стартовой вершины *a*. Иначе говоря, сначала посещается сама вершина *a*, затем все вершины, смежные с *a*, то есть находящиеся от нее на расстоянии 1, затем вершины, находящиеся от на расстоянии 2, и т.д.

Рассмотрим алгоритм поиска в ширину с заданной стартовой вершиной *a*. Вначале все вершины помечаются как новые. Первой посещается вершина *a*, она становится единственной открытой вершиной. В дальнейшем каждый очередной шаг начинается с выбора некоторой открытой вершины *x*. Эта вершина становится активной. Далее исследуются ребра, инцидентные активной вершине. Если такое ребро соединяет вершину *x* с новой вершиной *y*, то вершина *y* посещается и превращается в открытую. Когда все ребра, инцидентные активной вершине, исследованы, она перестает быть активной и становится закрытой. После этого выбирается новая активная вершина, и описанные действия повторяются. Процесс заканчивается, когда множество открытых вершин становится пустым.

Основная особенность поиска в ширину, отличающая его от других способов обхода графов, состоит в том, что в качестве активной вершины выбирается та из открытых, которая была посещена раньше других. Именно этим обеспечивается главное свойство поиска в ширину: чем ближе вершина к старту, тем раньше она будет посещена. Для реализации такого правила выбора активной вершины удобно использовать для хранения множества открытых вершин очередь - когда новая вершина становится открытой, она добавляется в конец очереди, а активная выбирается в ее начале.

**0**

**1**

**3**

**4**

**3**

**4**

**2**

**4**

**22**

**2**

**BFS(G, s)**

**0**

**1**

**2**

**3**

**3**

**4**

**Алгоритм поиска в глубину**

**(DFS, depth-first search)**

Поиск в глубину - вероятно, наиболее важная ввиду многочисленности приложений стратегия обхода графа. Идея этого метода - идти вперед в неисследованную область, пока это возможно, если же вокруг все исследовано, отступить на шаг назад и искать новые возможности для продвижения вперед. Метод поиска в глубину известен под разными названиями, например, "бэктрекинг", "поиск с возвращением".

Понятия новой, открытой, закрытой и активной вершин для поиска в глубину имеют такой же смысл, как и для поиска в ширину. Отметим, что всегда имеется не более чем одна активная вершина.

Обход начинается с посещения заданной стартовой вершины *a*, которая становится активной и единственной открытой вершиной. Затем выбирается инцидентное вершине *a* ребро (*a,y*) и посещается вершина *y*. Она становится открытой и активной. Заметим, что при поиске в ширину вершина *a* оставалась активной до тех пор, пока не были исследованы все инцидентные ей ребра. В дальнейшем, как и при поиске в ширину, каждый очередной шаг начинается с выбора активной вершины из множества открытых вершин. Если все ребра, инцидентные активной вершине *x*, уже исследованы, она превращается в закрытую. В противном случае выбирается одно из неисследованных ребер (*x,y*), это ребро исследуется. Если вершина *y* новая, то она посещается и превращается в открытую.

Главное отличие от поиска в ширину состоит в том, что при поиске в глубину в качестве активной выбирается та из открытых вершин, которая была посещена последней. Для реализации такого правила выбора наиболее удобной структурой хранения множества открытых вершин является стек: открываемые вершины складываются в стек в том порядке, в каком они открываются, а в качестве активной выбирается последняя вершина.

**1 /**

**1 /**

**2 /**

**1 /**

**2 /**

**3 /**

**1 /**

**2 /**

**3 /**

**4 /**

**1 /**

**2 /**

**3 /**

**4 /**

**5 /**

**1 /**

**2 /**

**3 /**

**4 /**

**5 /6**

**1 /**

**2 /**

**3 /**

**4 /6**

**5 /6**

**1 /**

**2 /**

**3 /**

**4 /7**

**5 /6**

**1 /**

**2 /**

**3 /8**

**4 /7**

**5 /6**

**1 /**

**2 /9**

**3 /8**

**4 /7**

**5 /6**

**1 /10**

**2 /9**

**3 /8**

**4 /7**

**5 /6**

**1 /**

**1 /**

**2 /**

**1 /**

**2 /**

**3 /**

**1 /**

**2 /**

**3 /**

**1 /**

**2 /**

**3 /**

**4 /**

**1 /**

**2 /**

**3 /**

**4 /5**

**1 /**

**2 /**

**3 /6**

**4 /5**

**1 /**

**2 /7**

**3 /6**

**4 /5**

**1 /8**

**2 /7**

**3 /6**

**4 /5**

**1 /8**

**2 /7**

**3 /6**

**4 /5**

**9 /**

**1 /8**

**2 /7**

**3 /6**

**4 /5**

**9 /10**

**ТОПОЛОГИЧЕСКАЯ СОРТИРОВКА**

***Топологическая сортировка***  - алгоритм упорядочивания вершин безконтурного ориентированного графа согласно ***линейного порядка*** (***частичный порядок*** – отношение со свойствами рефлективности, антисимметричности и транзитивности, необязательно на всем множестве), заданного дугами на множестве вершин.

Топологическая сортировка вершин орграфа G=(V,E) заключается в присвоении его вершинам номеров 1,..,n т.о., чтобы для любой дуги этого графа выполняется условие: (Vi,Vj) є Ei. Это возможно в том случае если граф не имеет контуров, является ациклическим.

Топологическая сортировка может рассматриваться как процесс определения линейного порядка на множестве вершин, в которые м.б. вложен частичный порядок, заданный множеством дуг. Топологическая сортировка начинается с нахождения вершины из которой не выходит ни одной дуги. Такая вершина всегда существует, если в графе нет контуров. Ей присваивается наибольший номер N и она удаляется из графа вместе с входящими в нее дугами. Т.к. оставшийся граф также не содержит контуров, процесс повторяется и новой вершине из которой не выходят дуги, присваивается наибольший номер (n-1) и т.д.

Рассмотренный подход нетрудно реализовать на матрице смежности, но потребуется n^2 операций. Трудоемкость можно уменьшить, если исключить повторяющиеся действия по нахождению вершин, из которых не выходят дуги. Это может быть достигнуто применением поиска в глубину. Для ациклического графа эта процедура обеспечивает последовательный выбор вершин, не имеющих выходящих дуг (при возврате из рекурсии).

При реализации топологической сортировки с помощью алгоритма поиска в глубину используется массив меток вершин Label[1..n], с помощью которого моделируется удаление вершин из графа и сохраняются новые номера вершин.

**1/**

**1/**

**2/**

**1/**

**2/**

**3/**

**1/**

**2/**

**3/**

**4/**

**1/**

**5**

**2/**

**3/**

**4/5**

**1/**

**2**

**2/**

**3/6**

**4/5**

**5**

**1/8**

**2**

**2/7**

**3/6**

**4/5**

**5**

**1**

**0**

**1/8**

**2**

**2/7**

**3/6**

**4/5**

**5**

**1**

**0**

**9/**

**1/8**

**2**

**2/7**

**3/6**

**4/5**

**5**

**1**

**0**

**9/**

**10/**

**1/8**

**2**

**2/7**

**3/6**

**4/5**

**5**

**1**

**0**

**9/**

**10/**

**11/**

**1/8**

**2**

**2/7**

**3/6**

**4/5**

**5**

**1**

**0**

**9/**

**10/**

**11/12**

**4**

**1/8**

**2**

**2/7**

**3/6**

**4/5**

**5**

**1**

**0**

**9/**

**10/13**

**11/12**

**4**

**6**

**1/8**

**2**

**2/7**

**3/6**

**4/5**

**5**

**1**

**0**

**9/14**

**10/13**

**11/12**

**4**

**6**

**3**

**1/8**

**2**

**2/7**

**3/6**

**4/5**

**5**

**1**

**0**

**9/14**

**10/13**

**11/12**

**4**

**6**

**3**

**15/**

**1/8**

**2**

**2/7**

**3/6**

**4/5**

**5**

**1**

**0**

**9/14**

**10/13**

**11/12**

**4**

**6**

**3**

**15/16**

**7**

**1/8**

**2**

**2/7**

**3/6**

**4/5**

**5**

**1**

**0**

**9/14**

**10/13**

**11/12**

**4**

**6**

**3**

**15/16**

**7**

**17/**

**1/8**

**2**

**2/7**

**3/6**

**4/5**

**5**

**1**

**0**

**9/14**

**10/13**

**11/12**

**4**

**6**

**3**

**15/16**

**7**

**17/**

**18/**

**1/8**

**2**

**2/7**

**3/6**

**4/5**

**5**

**1**

**0**

**9/14**

**10/13**

**11/12**

**4**

**6**

**3**

**15/16**

**7**

**17/**

**18/19**

**9**

**1/8**

**2**

**2/7**

**3/6**

**4/5**

**5**

**1**

**0**

**9/14**

**10/13**

**11/12**

**4**

**6**

**3**

**15/16**

**7**

**17/20**

**18/19**

**9**

**8**

**2**

**5**

**1**

**0**

**4**

**6**

**3**

**7**

**9**

**8**