**Практическая работа** **№** **2**

ТЕМА: Представление графа для обработки с помощью компьютера

      ЦЕЛЬ: Изучить способы представления г рафив как информационных   о 'является тов для Компьютерная обработки.

Задача: Выбрать согласно вашего номера в списке группы соответствующий граф и выполнить следующее:

1)      вручную подать граф при всех способах;

2)      запрограммировать представление произвольного конечного графа.

**ТЕОРИЯ:**

1)        Представление графов в памяти вашего компьютера зависит от структуры данных, которые допускает алгоритмических язык и типа ЭВМ.

а) Представление с помощью матрицы смежности, порядок которой спивподае с числом вершин,

где элемент (i - j) -й равен 1, если i-и вершина смежная с j -й вершиной, и (i - j) -й элемент равен 0 в протележному случае.

Для графов с большим количество и дуг это достаточно компактное представление, а для графов с небольшим числом дугматимо достаточно разжижению матрицу. Приведем примеры таких представлений для неоринтовного графа: G 1 и орграфа G 2:

                                x3

G1: x2 x5

010100

01100

x1 x4 x6 A (G1) = 0100

011

01

0

011000

x2 000000

x4 010110

x6 A (G2) = 000010

G2: x1 000001

000100

x3 x5

б) представление с помощью матрицы инцедентностей определяет граф однозначно потому имеет порядок nxm, где n-количество вершин, а m-количество ребер; элементы матрицы определяют наличие или отсутствие отношение инцидентности между вершинами и ребрами.

Используется редко из-за отсутствия алгоритмов обработки работающие с такой структурой.

в) Представление с помощью списков смежности является главной альтернотивою представлена с помощью матриц. Список смежности для вершины v является списком конечных дуг, исходящих из этой v вершины орграфа, или просто списком всех смежных с v вершиннеориентованого графу.

Приведем пример списочного представления графов, имели наводнения выше матричное представление для неориентовоного G 1:

x1: x2, x4; x3: x2, x4; x4: x1, x2, x3, x5, x6;

x2: x1, x2, x4; x5: x4, x6; x6: x5, x6;

представление и ориентированного графа G 2:

x1: x2, x3; x2: nil; x3: x2, x4, x5; x4: x5; x5: x6; x6: x4;

г) Представление с помощью списка дуг используют для сохранения различной информации о дуги. При этом способе каждой дужзи придают тройку чисел (u, x, y), где u = (x, y), х-начало, в-конец дуги. Вес дуги возможно представить как четвертое число в этой тройки.

**Код Харари**определяют с помощью матрицыA(G) - матрицы смежности путем пометовного записи строк из тех элементов, расположенных над главной диагональю, один за другим.Таким образом получим двоичное число, величина которого залежитеме от нумерации вершин.Больше изцих чисел будет будет кодом Хирари данного графуG.Нумерция вершин,

что соответствует кода Хирари называется канонично.

**Код Прюфера**используется для представления деревьев.Пусть Т-дерево с множеством вершин{V1,v2, ...vn}, где номер вершиныviравна и.Припишем дереву Т последовательность (a1,a2, ...an-2) построенную по следующему правиломЖ

1)        i = 1;

2)        в последовательности 1, 2, ... n путем просмотра слева направо ищем номер первой висячей вершины. Пусть это bi.

3)        Щука вершину что смежная с b i. Пусть это аи.

4)        В последовательности из пункта 2) вычеркиваем bi.

5)        В дереве Твидаляемо вершину bi.

6)        i = i + 1;

7)        если и <n-1, то переходим к 2), иначе вида {ai ... an 1}

Это и будет код Прюфера. Например для дерева:

   7 6 марта

1 8 2 4 5

код Прюфера Матей вид (8,4,4,4,2,2).

В случае ордерева построение кода Прюфера выполняется аналогично.Необхидно только на последнем месте писать корневую вершину и при декодировании кода недописуваты эту вершину. Так для ордерева:

Мать кодПрюфера равным (7,4,4,4,2,2,7).

Код Прюфера е оптимальным с точки зрения экономии памяти "памяти и доведен теореме Келли:

n-2

числом замеченных n - вершиндерев = n.

**№2-2** **Г** **лобальний анализ графов.**

Означает выделение структуры графа и определения характеристик выделенных структуры для решения задачи. Этот анализ заключается в збиранни информации о построении графа путем обхода вершин и дуг (ребер) графа. Информация полученная таким путем оформлюеться в виглядипидходящои нумерации вершин графа.

**1**           Нумеция, что обнаруживает логическую структуру графа.

1.1.           нумерация F будем называть приписывание вершинам графа G различных чисел (номеров) из множества натуральных чисел N, то F: V (G) 🡨 N. Из большого кассу нумерации наиболее важливишимы е нумерация построена на поиске в глубину (базисная нумерация), прямая нумерация и еранжировка. Иногда эти нумерации зовут линейными.

Поиск в глубину это обход вершин графа по следующим правилам:

1)        Знаходячись в вершине х надо двигаться в любю другую, ранее пройденное, если такая найдется, одновременно запомнить "ятовуючи дугу по которой впервые попали к вершине;

2)         Если с вершины х невозможно попасть в ранее пройденной вершины или таковой вообще нет, то повертаемося к вершине z с которой впервые попали в х и продолжает поиски в глубену с вершена z.

При выполнении обхода графа поцим правилам мы стараемся проникнуть в глубь графа насколько это возможно, затем отступает на шаг назад и снова пытаемся пройти в перед. При поиске в глубь орграфа можливопопасты в вершину y с вершины х тильки благодаря наличию дуги (x, y), то мы должны двигаться вперед только в направлении ориентации дуг, а возвращаться в протележному направлении.Внеорентованому графе таких ограничений нет. Будем называть М-нумерацию вершин графа ту нумерацию что соответствует порядка их обхода при поиске в глубину. Путь м = (g = x 1, x 2, ..., xn = p) называть М-путем, если для каждой и, i = 1 (1) n, выполняется условие:

М (хи) <М (хи + 1), то номер хи меньше номера хи + 1; Вершина г. называется М-достижимой с вершины g, если существует М-путь с g в p

1.2     Алгоритм поиска в глубину и построение М-нумерации в ориентированный граф имеет следующий вид:

Вход: Граф G = (V, E) задан списками смежности A (v), где v - вершина из множества V, A (v) ее список смежных вершин.

Выход: М-нумерация вершин и разбиение множеств E на четыре класса: деревянных дуг T, прямых дуг F, обращенных дуг B и поперечных дуг C.

начало

прцедуры ПОИСК (V, N);

начало;

1. присвоить виршини v номер M (v) = N;
2. N = N + 1;
3. для w Е A (v) // вершина w выбрана из списка A (v)

Выполнить цикла:

1. если w не имеет М-номера то
2. начало;
3. добавить дуги (v, w) в T;
4. Поиск (w, N);
5. конец;
6. иначе;
7. начало
8. если М-номер вершины w больше М-номера вершины v   то доьавиты (v, w) до F,
9. иначе ежели существует М -путь С w в v   то
10. добавить (v , W) до В;
11. иначе    добавить (v , W) до С;
12. конец // если с g)
13. конец // цикла
14. T = 0; F = 0; B = 0; C = 0; N = 1;
15. для v Е V выполнить в цикле
16. заметит вершину v как имея номера
17. пока существует вершина v без М-номера выполнить
18. ПОИСК (v, N);
19. конец цикла пока. // Конец алгоритма.

Пример М нумерации построенной поиском вглубь:

Где жирными дугами замечено деревьяни дуги из множества Т, тонкими дугами замечено прямые дуги из множества F, штриховыми линиями помичення обратные дуги из множества В, а пунктиром помичення поперечные дуги из множества С.

Использование стека упрощает реализацию алгоритма. Присвоении М номера происходит в тот момент, когда вершина вводится в стек; удаление вершины происходит в тот момент, когда оказываются пройденными все дуги, выходящие из данной вершины.

     N нумерация вершин графа, имеющего n -вершина, называется присвоении первый выброшенный из стека вершине номера n, а второй викинутй вершине присваемое номер n-1.

Поиск в глубину в неориентованому графе видризняеться от поиска в глубину в орграфе тем, что все ребра розбаваються на три класса:

Класс деревьяних (остовных) ребер, класс ребер соприкосновения и класс прямых ребер, где класс ребер соприкосновения соответствует класамобернених и поперечных дуг в орграфе т.то В + С. Этим вичерпууться изменения в приведенном алгоритме.

Поиск в глубину и неорентованому графе превращает входной граф G на орграф G ', путем наведения на каждом ребре ориентации в направлении следования.

1.3.

Аранжировкой или А-нумерации будем называть такую нумерацию когда каждый простой путь с входом А однохидного графа называется А-путем. Граф называется аранжируемим если он допускает аранжировку.

Пример аранжуруемого приведем граф G 1:

А граф G 2 е неаранжуемим:

Не факт неаранжуемости указывает наличие двух путей от входа s = v 0 в вершину v 2, один из которых содержит вершину v 3, а второй нет.

2-4

Построение аранжировки для ацикличного графа осуществляется по следующему алгоритму:

Вход: Граф G = (V, E) задан стеками смежности A (v), где V 0 множество входных вершин.

Выход: Аранжировка (А-нумерация) вершин графа.

1. начало
2. процедура Аранжировка (v, N);
3. начало
4. присвоить вершине v номер N;
5. N = N + 1;
6. для w Е А (v) цикла
7. Выполнить Аранжировка (w, V), если все предыдущие вершины w имеют в-номера
8. конец
9. N = 1
10. Для v Е V0 цикла
11. Заметить вершину v как имеющую А-номера; конец цикла
12. Покой существует вершина v Е V 0 без А- номера выполнить цикл аранжировки ( v, N);
13. Конец цикла пока
14. конец алгоритма

1.4.

   Прямая нумерация выполняется на базе М-нумерации и заключается в определении формальных циклов графа, где под формальным циклом понимают граф порождением вершинами простого пути F (i,.. K) и обратной дуги (Vk, Vi) считаем что от ш-й вершины в k - той растут (i <j <k).

**№** **2** **-4** **Логический анализ графов.** **Линейные компоненты.**

2.1.          Матрица досягаемости и транзетивне замыкания.

В случае, когда необходимо установить Лише факт наличия путей между вершинами, возникает задача о построении транзитивного замыкания орграфа. Транзетивним замыканием G \* графа G называется орграф с тиеюже множеством вершин что и G, но в нем присутствует дуга (xi xj) тогда и только тогда, когда вершина xj достижима с xi, .т. то существует путь (xi - xy). Матрицей смежности транзетивного замыкания G \* орграфа G служит матрица досягаемости R (G) графа G, т. То квадратная матрица порядка n, n = / V (G) /, с элементами rij = 1, если вершина xj достижима с x i, также rij = 0.

Самым простым алгоритмом построения матрицы R (G) имеет вид:

Вход: Матрица А (G) смежности орграфа G, имеет строки А1, А2, .. А n.

Выход: Матрица досягаемости R (G), что имеет строки R 1, R 2, .. Rn.

Начало алгоритма:

1. для и от 1 до n шаг 1 выполнить цикла
2. построить множество J <-> {J}, таких индексов, aij = 1;
3. Ri = Ai; K = 0;
4. Пока J <-> 0 (J непустое множество) выполнить цикл
5. Выбрать j Е J;
6. Ri = Ri + Aj, где + - логическая операция "и", то есть обеднаня матриц одного порядка, то есть 1 или в одной или в другой матрицы дает А в результативной матрицы.
7. J = J \ {y};
8. K = KU {j};
9. Сформировать множеству J и - индексов К таким, что а jk = 1;
10. J = JU (Jj \ K)
11. Конец цикла;
12. Выводим Ri;
13. Конец цикла;

Конец алгоритма.

Например, для графа G и его матрицы А (G):

0110

1000

: A (G) = 0,001

0010

имеем матрицу R (G) и G \* -транзетивне замыкания:

1111

1111

         R (G) = 0011

0011

Более эффективным является алгоритм Уоршела (Warahall), которой находит матрицу R путем вычисления последовательности квадратных матриц

0 1 n

В, В ... В порядка n по следующим правилам:

           0 e e

1. В = А (G) // В = {В ij};
2. Для L = 1 до n кроком1 выполнить цикл

L L-1 L-1

1. B ij = B ij + | B ij & Bij | ; // + Логическое сложение диеты множество ребер из числа присутствующих или в одном или во втором
2. конец цикла

               n

1. B = B;

Конец алгоритма.

е e

Элементам В и j предоставить тот смысл, что В ij = 1 тогда и только тогда, когда ершина хи и х j связаны путем, проходящей через вершины хи, х ... х j.

0

Тогда начальный шаг В = А (G) будет означать пути без прмижних вершин, а последний шаг

          n

  R = B означает наличие путей через любые промежуточные вершины. Цикл виконувате провирки факта, что хи и х j звязанни путем, промежуточные вершины которого будут принадлежать множеству {xi, ... xj} и удовлетворяют случаев:

1)        существует путь от хи к xj, проходящей через (xi ... x L 1};

2)        существует путь от хи к х L и от х L к xj, все промежуточные вершины которого принадлежат множеству (xi ... x L 1}.

4-2

2.2.

**№ 2-3** **Отыскания бы** **компонент орграфа.**

Б и компонентой называют самый большой (по включением) подграф в котором любая пара вершин достижима из каждой вершины другой пары вершин. Эффективный алгоритм нахождения биокомпонент на поиске "вглубину" имеет следующий вид:

Вход: Граф G = (V, E), заданный списками смежности A (v).

Выход: Список В с биокомпонент.

   начало алгоритма

1. Процедура биокомп (End (TT))
2. начало процедуры
3. V = End (TT)
4. Для w Е А (v) выполнить цикл
5. Если А (v) не равно 0, то

6-8. если S (w) = 0, то 1) ТТ = w; S (w) = 1

2) БИКОМП (w)

2. иначе // w размещения в стеке ТТ

11-12.                    взыскать хвост стека ТТ починаючи с вершины w к вершине w ', т.то удалять, починаючи с вешины w хвост ТТ, запоминать удалении вершины и вместо w взять w' для которой список смежности е списком получений путем обеднання списков смежности стянуть вершин.

13. БИКОМП (w ');

1. иначе 1) видаиты v с стека ТТ и занести в В, т.то в список биокомпонент вершину v;

2) БИКОМП (End (TT))

1. конец цикла;
2. Возвращают В;
3. Конец процедуры БИКОМП ();
4. конец алгоритма.

7-1

7.2.

**Минимизация памяти при исчислении арифметических выражений.**

Арифметические выражения могут отображаться как ордерева, в каждую вершину которых входит не более двух дуг. Начальные данные соответствуют висячим вершинам дерева, а промежуточные результаты вычислений.Каждый внутренний вершина отражает бинарную операцию над оргументамы, представленными вершинами- предшественниками. Порядок в котором надо выберите аргументы несуттевий в простых моделях.Наприклад процесс вычисления арифметического выражения ((a + b) - c \* d) / (e \* (f - g)) изображений следующим образом:

Выбор возможного порядка операций означает топологически упорядочить дерево, т.то розтошуваты его вершины в "целочисленных" точках числовой прямой в такой последовательности где вершина v предшествует вершины v 'если существует дуга из v' в v, т.то дуга (v ', v). Это условие эквивалентно требованию, чтобы промежуточный результат исчислялся раньше использовался. Пример топологического упорядочения дерева, для приведенного выше выражения имеет вид:

1 2 3 4 5 6 7 8 9 10 11 12 0

Отметим, что величины которые не используются в качестве операндов- аргументов операции, выполняется в текущий момент времени, должны храниться в памяти.

На рисунке топологического упорядочения дерева тиким велечин соответствуют дуги, которые проходят над вершиной что обозначает текущую операцию.

Таким образом возникает оптимизационная задача:

Минимизувты килькистькомирок памяти для организации вычисления арифметических выражений.

Эта задача сводится к построению топологических впорядкувань ордерева с минимальной расширенных.

**Генерация оптимального кода для арифметических выражений.**

Пусть имеем машину с неограниченным памяти и Т -регистры и допустимыми е команды следующих типов:

1)        переправки: память 🡪 регистр;

2)        переправки: регистр 🡪 память;

3)        Операция: (регистр, память) 🡪 регистр;

4)        Операция: (регистр, регистр ) 🡪 регистр.

Заметим на то, что в Перацим: (регистр, память) 🡪 регистр; е запрещенной. Под ценой вычислений будем понимать количество операторов (программных шагов) необходимых для полного процесса обчислень. Для того, чтобы определить наймеьшу количество потребных регистров, а также оптимальную последовательность операций будем использовать разметку вершин, учитывая некоммутативными оерации. Разметку выполнит следующим образом:

1)        если вершина v висящая и е левым потоком некоторой вершины, то положим L (v) = 1,

если она является правым потоком, то будем считать L (v) = 0 /

2)        если вершина v имеет потоков с метками L 1, L 2, то при L 1 # L 2 положим,

что L (v) = max (L1, L2), а при L 1 = L 2 = L считать, что L (v) = L 1.

Алгоритм розвязку мае наступний вигляд:

Початок алгоритму

1. процедура Т ( v );
2. початок процедури
3. якщо f( v ) то

4-5.        якщо v - вісяча вершина , то заслати значення вершини v у доступний регістр В m із найменьшим номером; // в цьому випаку вешина v е лівим потоком свого предка //

1. інакше T( &( v ) ) ;

7-8. інакше   якщо   min ( L (&( v ) ) , L ( p( v ) ) ) => N то Т ( p( v ) );

// користуемося позначенням &( v ) для лівого left ( v ), а через P ( v ) позначмо right ( v ) – правого потомка для v .

1. інакше   якщо   L (&( v ))# L ( P ( v ) ) то
2. початок
3. w := потомок вершини v із найбільшою міткою.
4. T( w );
5. кінец;
6. інакше T(P( w ));
7. якщо ( L ( v )=1 ) ^( v-не е висячою вершиною ) то
8. початок
9. обчислити вершину v , взявши значення лівого потомка із регістру Bm , а значення правого потомка із памяті
10. заслати значення вершини v до регістру Bm ;
11. кінец;
12. інакше початок
13. T(&( v ));
14. виконати операцію , що відповідае вершині v , беручі значення лівого потомка з регістру Bm , а значення правого потомка з регістру Bm +1;

// на цьому кроці доступними е регістри Bm , Bm +1;

23. я кщо   ( L ( v ) => N )^( v - правий птомок свого предка )   то

24. початок

1. значення вершини v заслати в память;
2. звільнити регистри Bm , Bm+1;
3. кінець;
4. кінець // якщо з номером 3
5. кінец процедури T( v )
6. v = корень дерева Т.

кінець

Розглянемо приклад роботи цього алгоритму над виразом

(( ab-c) /( d+c ))/( g+i )/( j+k )\*Lm)).

Дерево для цього виразу мае вигляд:

Для N =2 алгоритм генеруе такий код:

1. f 🡪 B1;
2. B1+k 🡪 B1;
3. B1\*L 🡪 B1;
4. B1-m 🡪 B1;
5. G 🡪 B2;
6. B2+I 🡪 B2;
7. B1/B2 🡪 ( память );
8. D 🡪 B1;
9. B1+e 🡪 B1;
10. A 🡪 B2;
11. B2\*B 🡪 B2;
12. B2-c 🡪 B2;
13. B1/B2 🡪 B1;
14. B1/( память ) 🡪 B1;