# Распределенные информационно-аналитические системы

## Лекция № 12.

## Алгоритмы обхода

Профессор кафедры КБ-2: д.т.н. Шатовкин Р.Р.

# *Учебные цели:*

*Изучить основы научных знаний по алгоритмам обхода: последовательному алгоритму опроса (обходу клик); алгоритму обхода для торов; алгоритму обхода для гиперкубов; алгоритму обхода Тарри (обходу связных сетей). Сравнить алгоритмы обхода.*

# *Учебные вопросы:*

***1. Последовательный алгоритм опроса (обход клик).***

***2. Алгоритм обхода для торов.***

***3. Алгоритм обхода для гиперкубов.***

***4. Алгоритм обхода Тарри (обход связных сетей).***

***5. Сравнение алгоритмов обхода.***

Рассмотрим особый класс волновых алгоритмов, а именно, волновые алгоритмы, в которых все события волны совершенно упорядочены каузальным отношением, и в котором последнее событие происходит в том же процессе, где и первое.

**Определение 1.** ***Алгоритмом обхода*** называется алгоритм, обладающий следующими тремя свойствами:

(1) В каждом вычислении один инициатор, который начинает выполнение алгоритма, посылая ровно одно сообщение.

(2) Процесс, получая сообщение, либо посылает одно сообщение дальше, либо принимает решение.

Из первых двух свойств следует, что в каждом конечном вычислении решение принимает ровно один процесс. Говорят, что алгоритм завершается в этом процессе.

(3) Алгоритм завершается в инициаторе и к тому времени, когда это происходит, каждый процесс посылает сообщение хотя бы один раз.

В каждой достижимой конфигурации алгоритма обхода либо передается ровно одно сообщение, либо ровно один процесс получил сообщение и еще не послал ответное сообщение. С более абстрактной точки зрения, сообщения в вычислении, взятые вместе, можно рассматривать как единый объект (маркер), который передается от процесса к процессу и, таким образом, «посещает» все процессы.

Алгоритмы обхода используются для построения алгоритмов выбора и для этого важно знать не только общее количество переходов маркера в одной волне, но и сколько переходов необходимо для того, чтобы посетить первые x процессов.

**Определение 2.** Алгоритм называется ***алгоритмом f-обхода*** (для класса сетей), если:

(1) он является алгоритмом обхода (для этого класса);

(2) в каждом вычислении после f(x) переходов маркера посещено не менее min (N, x + 1) процессов.

Кольцевой алгоритм является алгоритмом обхода, и, поскольку x + 1 процесс получил маркер после x шагов (для x < N), а все процессы получат его после N шагов, это алгоритм x-обхода для кольцевой сети.

# 1. Последовательный алгоритм опроса (обход клик)

Клику можно обойти путем последовательного опроса; алгоритм очень похож на алгоритм опроса, но за один раз опрашивается только один сосед инициатора. Только когда получен ответ от одного соседа, опрашивается следующий.

***Последовательный алгоритм опроса***:

var $rec_p$ : integer init 0 ; (* только для инициатора *)

**Для инициатора:**

(* обозначим $Neigh_p = \{q_1, q_2, .., q_{N-1}\}$ *)

begin while $rec_p$ < # $Neigh_p$ do

begin send <tok> to $qrec_p$ + 1 ;

receive <tok>;

$rec_p$ := recp + 1

end ;

decide

end

**Для не-инициатора:**

begin receive <tok> from q ;

send <tok> to q end

**Теорема 1.** Последовательный алгоритм опроса является алгоритмом 2x-обхода для клик.

*Доказательство.* Легко заметить, что к тому времени, когда алгоритм завершается, каждый процесс послал инициатору ответ. (2x – 1)-е сообщение – это опрос для $q_x$, а (2x)-е – это его ответ. Следовательно, когда было передано 2x сообщений, был посещен x + 1 процесс p, $q_1$, …, $q_x$.

# 2. Алгоритм обхода для торов

***Тором*** m × n называется граф G = (V, E), где

$V = Z_m \times Z_n = \{(i, j) : 0 \le i, j < n\}$

и

$E = \{(i, j)\,(i', j') : (i = i' \ \& \ j = j' \pm 1) \vee (i = i' \pm 1 \ \& \ j = j')\}$.

Граф вида «тор» представляет собой решетку с дополнительными ребрами, соединяющими вершины из верхнего ряда («строки») решетки с вершинами из нижнего ряда, а также с ребрами, соединяющими вершины из левого ряда («столбца») решетки с вершинами из правого ряда. Таким образом, возникают дополнительные циклы. Каждая вершина тора (в отличие от решетки) имеет степень 4, то есть граф является регулярным.

При стандартном изображении такого графа каждая вершина имеет «левого» соседа, «правого» соседа, «верхнего» соседа, «нижнего» соседа (рисунок 1).

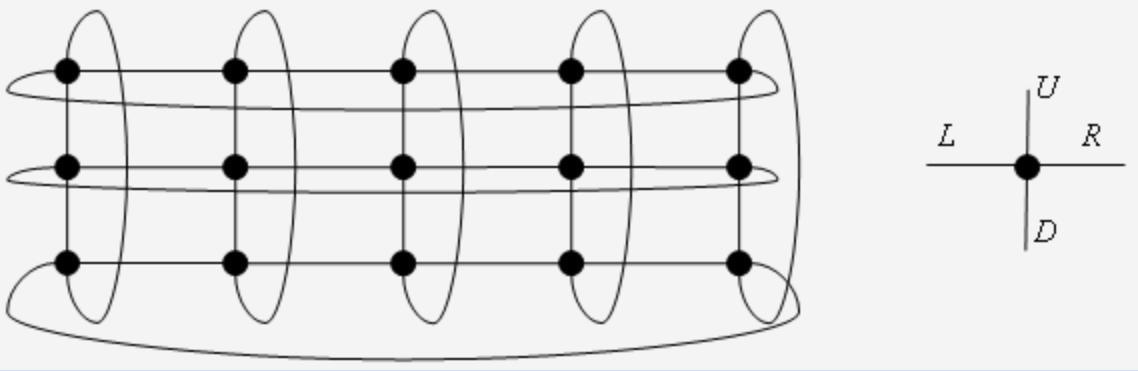


**Рисунок 1 – Тор 3 × 5. Ребра различных направлений**

Тор, построенный на основе решетки m × n, содержит mn узлов и 2mn ребер. Количество ребер можно подсчитать как непосредственно, так и на основе известной теоремы теории графов: «сумма степеней узлов равна удвоенному числу ребер».

Из-за его регулярных свойств тор или его модификации часто используются в различных архитектурах. Например, одна из первых архитектур многопроцессорных ЭВМ ILLIAC-IV использовала торо-подобное соединение процессоров.

Требуется обойти все узлы тора m × n (но не все ребра). Инициатором – начальным узлом при обходе может быть любой из узлов. Алгоритм ориентирован на топологию тора. Предполагается, что тор обладает чувством направления, и имеются имена ребер: L – Left, R – Right, U – Up, D – Down, определяющие направления в соответствии с приведенным выше рисунком. Их можно считать локальным знанием о топологии.

Положим, что в вершине (i, j) канал k (i, j + 1) имеет метку Up, канал k (i, j – 1) – метку Down, канал k (i + 1, j) – метку Right, и канал k (i – 1, j) – метку Left.

Координатная пара (i, j) удобна для определения топологии сети и ее чувства направления, но предполагается, что процессы не знают этих координат; топологическое знание ограничено метками каналов.

***Алгоритм обхода для торов*:**

**Для инициатора (выполняется один раз):**

send <num, 1> to Up

**Для каждого процесса при получении маркера <num, k>:**

begin if k = m∗n then decide

else if n|k then send <num, k + 1> to Up

else send <num, k + 1> to Right

end

На рисунках 2 и 3 показан порядок обхода торов 3 × 4 и 2 × 5. Звездочкой обозначен инициатор обхода, а буквами «Ok» – вершина, сообщающая о завершении обхода. Около ребра записано число – значение k, которое передается по этому ребру от одной вершины к другой.

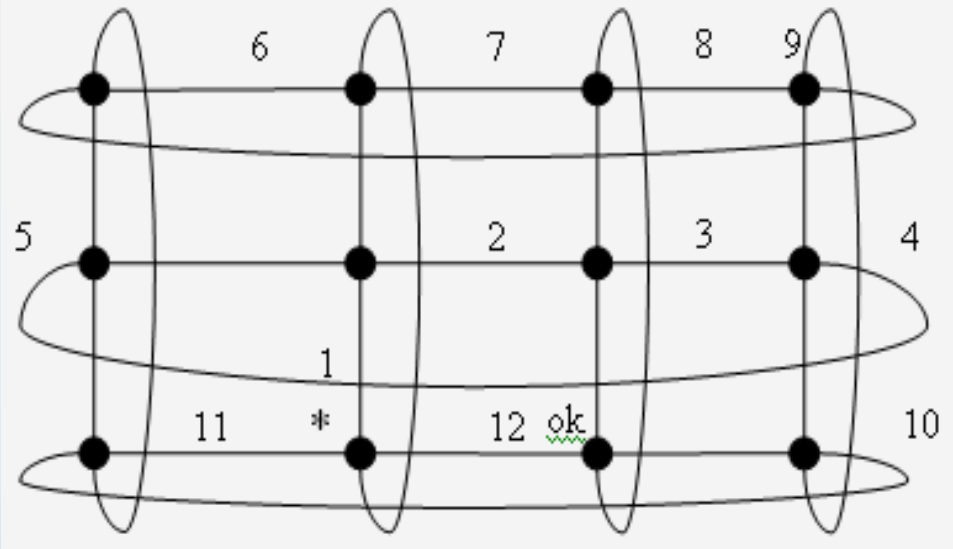


**Рисунок 2 – Порядок обхода тора 3 × 4**

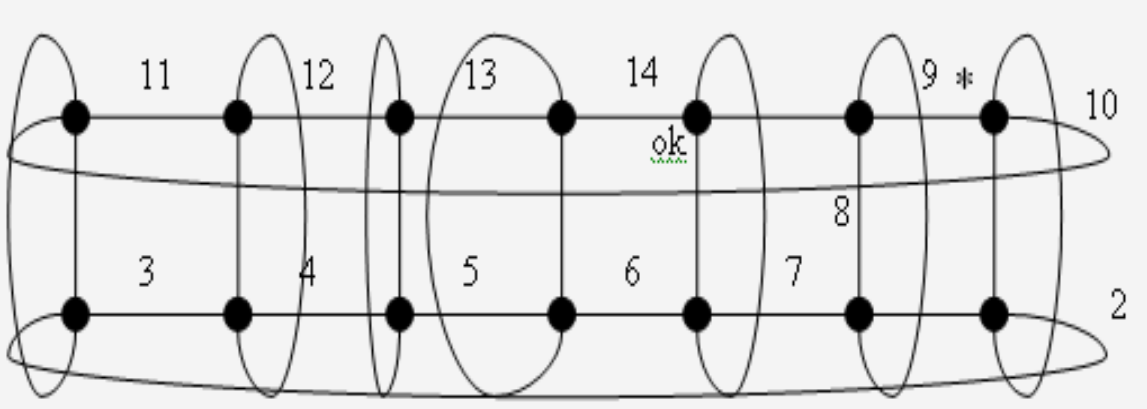


**Рисунок 3 – Порядок обхода тора 2 × 5**

Тор является Гамильтоновым графом, то есть в торе (произвольного размера) существует Гамильтонов цикл, и маркер передается по циклу с использованием алгоритма обхода для торов. После k-го перехода маркера он пересылается вверх, если n|k (k делится на n), иначе он передается направо.

**Теорема 2.** Алгоритм обхода для торов является алгоритмом x-обхода для торов.

*Доказательство.* Как легко заметить из алгоритма, решение принимается после того, как маркер был передан $n^2$ раз. Если маркер переходит от процесса p к процессу q с помощью U переходов вверх и R переходов вправо, то p = q тогда и только тогда, когда (n|U & n|R).

Обозначим через $p_0$ инициатор, а через $p_k$ – процесс, который получает маркер <num, k>.

Из $n^2$ переходов маркера n – переходы вверх, а оставшиеся $n^2 – n$ – переходы вправо. Так как и n, и $n^2 – n$ кратны n, то $p_n^2 = p_0$, следовательно, алгоритм завершается в инициаторе.

Все процессы с $p_0$ по $p_{n^2-1}$ различны; так как всего $n^2$ процессов, то отсюда следует, что каждый процесс был пройден.

Предположим, что $p_a = p_b$ для $0 \le a < b < n^2$. Между $p_a$ и $p_b$ маркер сделал несколько переходов вверх и вправо, и так как $p_a = p_b$, количество переходов кратно n. Изучив алгоритм, можно увидеть, что отсюда следует, что

$$\#\{k : a \le k < b \ \&\ n|k\} \text{ кратно } n,$$

и

$$\#\{k : a \le k < b \ \&\ n \text{ не } |k\} \text{ кратно } n.$$

Размеры двух множеств в сумме составляют b – a, откуда следует, что n|(b – a). Обозначим (b – a) = λ*n, тогда множество {k: a ≤ k < b} содержит λ кратных n. Отсюда следует, что n|λ, а значит $n^2|(b – a)$, что приводит к противоречию.

Так как все процессы с $p_0$ по $p_{n^2-1}$ различны, после x переходов маркера будет посещен x + 1 процесс.

# 3. Алгоритм обхода для гиперкубов

***N-мерным гиперкубом*** называется граф G = (V, E), где

$$V = \{(b_0, ..., b_{n-1}) : b_i = 0, 1\}$$

и

$$E = \{(b_0, ..., b_{n-1}), (c_0, ..., c_{n-1}) : b \text{ и } c \text{ отличаются на 1 бит}\}.$$

Предполагается, что гиперкуб обладает чувством направления, то есть канал между вершинами b и c, где b и c различаются битом с номером i, помечается «i» в обеих вершинах. Предполагается, что метки вершин не известны процессам; их топологическое знание ограничено метками каналов.

Как и тор, гиперкуб является Гамильтоновым графом, и Гамильтонов цикл обходится с использованием алгоритма обхода для гиперкубов. Доказательство корректности алгоритма похоже на доказательство для алгоритма обхода для торов.

***Алгоритм обхода для гиперкубов:***

**Для инициатора (выполняется один раз):**

send <num, 1> по каналу n – 1

**Для каждого процесса при получении маркера <num, k>:**

begin if $k = 2^n$ then decide

else begin пусть l – наибольший номер : $2^l | k$ ;

send <num, k + 1> по каналу l

end

end

**Теорема 3.** Алгоритм обхода для гиперкубов является алгоритмом x-обхода для гиперкубов.

*Доказательство.* Из алгоритма видно, что решение принимается после $2^n$ пересылок маркера. Пусть $p_0$ – инициатор, а $p_k$ – процесс, который получает маркер <num, k>. Для любого $k < 2^n$, обозначения $p_k$ и $p_{k+1}$ отличаются на 1 бит, индекс которого обозначим как $\lambda(k)$, удовлетворяющий следующему условию:

Так как для любого $i < n$ существует равное количество $k \in \{0, ..., 2^n\}$ с $\lambda(k) = i$, то $p_0 = p_{2^n}$ и решение принимается в инициаторе. Аналогично доказательству Теоремы 2, можно показать, что из $p_a = p_b$ следует, что $2^n | (b - a)$, откуда следует, что все $p_0, ..., p_{2^n - 1}$ различны.

Из всего этого следует, что, когда происходит завершение, все процессы пройдены, и после x переходов маркера будет посещен x + 1 процесс.

# 4. Алгоритм обхода Тарри (обход связных сетей)

Алгоритм обхода для произвольных связных сетей был дан Тарри.

Алгоритм сформулирован в следующих двух правилах:

R1. Процесс никогда не передает маркер дважды по одному и тому же каналу.

R2. Не-инициатор передает маркер своему родителю (соседу, от которого он впервые получил маркер), только если невозможна передача по другим каналам, в соответствии с правилом R1.

***Алгоритм обхода Тарри:***

var $used_p$ [q] : boolean init false для всех $q \in Neigh_p$ ;

(* Признак того, отправил ли p сообщение q *)

$father_p$ : process init udef ;

**Для инициатора (выполняется один раз):**

begin $father_p := p$ ; выбор $q \in Neigh_p$ ;

$used_p$ [q] := true ; send <tok> to q ;

end

**Для каждого процесса при получении <tok> от $q_0$ :**

begin if $father_p$ = udef then $father_p := q_0$ ;

if $\forall q \in Neigh_p : used_p$ [q]

then decide

else if $\exists q \in Neigh_p : (q \neq father_p \ \& \ \neg used_p$ [q])

then begin выбор $q \in Neigh_p \setminus \{father_p\}$ с $\neg used_p$ [q] ;

$used_p$ [q] := true ; send <tok> to q

end

else begin $used_p$ [$father_p$] := true ;

send <tok> to $father_p$

end

end

**Теорема 4.** Алгоритм Тарри является алгоритмом обхода.

*Доказательство.* Так как маркер передается не более одного раза в обоих направлениях через каждый канал, всего он передается не более 2|E| раз до завершения алгоритма. Так как каждый процесс передает маркер через каждый канал не более одного раза, то каждый процесс получает маркер через каждый канал не более одного раза. Каждый раз, когда маркер захватывается не-инициатором p, получается, что процесс p получил маркер на один раз больше, чем послал его.

Отсюда следует, что количество каналов, инцидентных p, превышает количество каналов, использованных p, по крайней мере, на 1. Таким образом, p не принимает решение, а передает маркер дальше. Следовательно, решение принимается в инициаторе.

Далее, за 3 шага будет доказано, что, когда алгоритм завершается, каждый процесс передал маркер:

(1) Все каналы, инцидентные инициатору, были пройдены один раз в обоих направлениях. Инициатором маркер был послан по всем каналам, иначе алгоритм не завершился бы. Инициатор получил маркер ровно столько же раз, сколько отправил его; так как инициатор получал маркер каждый раз через другой канал, то маркер пересылался через каждый канал по одному разу.

(2) Для каждого посещаемого процесса p все каналы, инцидентные p были пройдены один раз в каждом направлении. Предположив, что это не так, выберем в качестве p первый встретившийся процесс, для которого это не выполняется, и заметим, что из пункта (1) p не является инициатором. Из выбора p, все каналы, инцидентные $father_p$ были пройдены один раз в обоих направлениях, откуда следует, что p переслал маркер своему родителю. Следовательно, p использовал все инцидентные каналы, чтобы переслать маркер; но, так как маркер в конце остается в инициаторе, p получил маркер ровно столько же раз, сколько отправил его, то есть p получил маркер по одному разу через каждый инцидентный канал. Мы пришли к противоречию.

(3) Все процессы были посещены и каждый канал был пройден по одному разу в обоих направлениях. Если есть непосещенные процессы, то существуют соседи p и q такие, что p был посещен, а q не был. Это противоречит тому, что все каналы p были пройдены в обоих направлениях. Следовательно, из пункта (2), все процессы были посещены и все каналы пройдены один раз в обоих направлениях.

Каждое вычисление алгоритма Тарри определяет остовное дерево сети. В корне дерева находится инициатор, а каждый не-инициатор p в конце вычисления занес своего родителя в дереве в переменную $father_p$. Желательно, чтобы каждый процесс также знал (в конце вычисления), какие из его соседей являются его сыновьями в дереве. Этого можно достигнуть, посылая родителю $father_p$ специальное сообщение.

# 5. Сравнение алгоритмов обхода

В Таблице 1 для сравнения приведены рассмотренные алгоритмы обхода.

**Таблица 1 – Алгоритмы обхода**

| Алгоритм | Топология | C/D | T | Сообщения | Время |
|---|---|---|---|---|---|
| Последовательный опрос | клика | C | да | 2N – 2 | 2N – 2 |
| Для торов | тор | C | да | N | N |
| Для гиперкубов | гиперкуб | C | да | N | N |
| Тарри | произвольная | C | да | 2\|E\| | 2\|E\| |

В столбце «C/D» отмечено, является ли алгоритм централизованным (C) или децентрализованным (D); столбец «T» определяет, является ли алгоритм алгоритмом обхода; в столбце «Сообщения» дана сложность сообщений; в столбце «Время» дана временная сложность. В этих столбцах N – количество процессов, |E| – количество каналов.

# Выводы

*В ходе лекции рассмотрены следующие вопросы:*

*- последовательный алгоритм опроса (обход клик);*

*- алгоритм обхода для торов;*

*- алгоритм обхода для гиперкубов;*

*- алгоритм обхода Тарри (обход связных сетей);*

*- сравнение алгоритмов обхода.*

## Задание на самостоятельную работу

*1. Конспект лекций.*

## Вид и тема следующего занятия

***Практическое занятие №12. Состояние приложения. Куки. Сессии. Обработка ошибок***