Распределенные алгоритмы

ЛЕКТОР: В.А. Захаров

Лекция 8.

Задача избрания лидера.

Основные определения и допущения.

Волновые алгоритмы избрания лидера.

Выборы лидера на дереве.

Выборы в кольцах.

Алгоритм Ле-Ланна.

Алгоритм Ченя – Робертса.

Алгоритм Петерсона/Долева–Клейва–Роде.

Эффект угасания.

Задача избрания лидера

Задача избрания лидера состоит в том, чтобы, из конфигурации, в которой все процессы пребывают в одном и том же состоянии, достичь такой конфигурации, в которой ровно один процесс будет находиться в состоянии **leader** , а все остальные процессы — в состоянии **lost** .

Выборы лидера среди процессов проводятся, когда нужно выполнить какой-нибудь централизованный алгоритм, и нет заранее заданного кандидата, который мог бы выступить инициатором этого алгоритма.

Такое происходит, когда инициализация выполняется в начале работы или после выхода системы из строя.

Поскольку множество активных процессов может быть заранее неизвестно, нельзя назначить раз и навсегда один из процессов на роль лидера.

Основные определения и допущения

Определение

Алгоритмом избрания лидера называется алгоритм, который обладает следующими свойствами.

1. Каждый процесс наделен одним и тем же локальным алгоритмом.
2. Алгоритм является децентрализованным, т.е. вычисление может быть инициировано произвольным непустым подмножеством процессов.
3. Алгоритм достигает заключительной конфигурации в каждом вычислении, и в каждой заключительной конфигурации существует ровно один процесс, который находится в состоянии **leader** а все остальные процессы при этом пребывают в состоянии **lost**.

Основные определения и допущения

Последнее свойство иногда бывает ограничено лишь требованием того, чтобы ровно один процесс находился в состоянии **leader**. Если задан алгоритм, удовлетворяющий этому ослабленному требованию, то его можно легко дополнить потоком сообщений, инициированным лидером, благодаря которому все процессы будут оповещены о результатах выборов.

Во всех алгоритмах процесс p наделен переменной $state_p$, множество значений которой включает в себя **leader** и **lost**. Иногда предполагается, что значением $state_p$ является **sleep** до того, как p выполнил хоть один шаг нашего алгоритма, и **cand** как только p присоединился к вычислению, но еще не был осведомлен о том, выиграл или проиграл он эти выборы.

Основные определения и допущения

Задача о выборах изучается в рамках следующих допущений.

1. **Система вполне асинхронна.** Процессы не имеют доступа к общим часам, и передача сообщений может длиться сколь угодно долго или коротко.

   Допущение о синхронной передаче сообщений почти не влияет на результаты, относящиеся к задаче о выборах: все рассмотренные алгоритмы можно применять в системах с синхронной передачей сообщений.

   Допущение о глобальном отсчете времени (процессы могут отслеживать реальное время, задержка передачи сообщения ограничена и пр.) действительно оказывает важное влияние на решение задачи о выборах.

Основные определения и допущения

2 **Каждый процесс опознается по уникальному имени, и это имя изначально известно самому процессу.**

Имена процессов (отличительные признаки) образуют линейно упорядоченное множества $\mathcal{P}$.

При проектировании алгоритма избрания можно постулировать, что на выборах победит процесс с наименьшим (наибольшим) по порядку отличительным признаком.

Так возникает задача отыскания наименьшего отличительного признака посредством децентрализованного алгоритма — задача отыскания экстремумов .

Основные определения и допущения

3 **Каждое сообщение может содержать $O(w)$ бит.**

В каждом сообщении может содержаться лишь константное число отличительных признаков процессов. Это допущение сделано для того, чтобы справедливо сопоставлять сложность по числу обменов сообщениями для различных алгоритмов.

Волновые алгоритмы избрания лидера

Отличительные признаки процессов можно использовать для нарушения симметрии между процессами, и алгоритм избрания можно спроектировать так, чтобы выделять процесс с наименьшим отличительным признаком. Наименьший отличительный признак может быть вычислен процессами по ходу одной волны (почему?).

Волновые алгоритмы избрания лидера

Отличительные признаки процессов можно использовать для нарушения симметрии между процессами, и алгоритм избрания можно спроектировать так, чтобы выделять процесс с наименьшим отличительным признаком. Наименьший отличительный признак может быть вычислен процессами по ходу одной волны (почему?).

Значит, выборы можно провести, запустив волну вычисления наименьшего отличительного признака, после чего процесс с наименьшим признаком становится лидером. Так как алгоритм избрания должен быть децентрализованным, этот принцип можно применять только к децентрализованным волновым алгоритмам.

Выборы лидера на дереве

Задача.

Доказать, что алгоритм избрания лидера на основе задачи отыскания экстремумов является волновым алгоритмом, если событие избрания процесса лидером рассматривать как событие решения.

Выборы лидера на дереве

Если топология сети имеет вид дерева, или в сети есть остовное дерево, то для выборов можно воспользоваваться древесным волновым алгоритмом. Но здесь требуется, чтобы инициаторами были все листовые вершины.

Выборы лидера на дереве

Если топология сети имеет вид дерева, или в сети есть остовное дерево, то для выборов можно воспользоваваться древесным волновым алгоритмом. Но здесь требуется, чтобы инициаторами были все листовые вершины.

Чтобы запустить алгоритм, когда лишь часть процессов являются инициаторами, к нему добавлен этап побудки . Процессы, затевающие выборы, наводняют сеть сообщениями **wakeup**. После получения сообщение **wakeup** по всем каналам, процесс запускает древесный волновой алгоритм, который снабжен приставкой для вычисления наименьшего отличительного признака.

Выборы лидера на дереве

Если топология сети имеет вид дерева, или в сети есть остовное дерево, то для выборов можно воспользоваться древесным волновым алгоритмом. Но здесь требуется, чтобы инициаторами были все листовые вершины.

Чтобы запустить алгоритм, когда лишь часть процессов являются инициаторами, к нему добавлен этап побудки . Процессы, затевающие выборы, наводняют сеть сообщениями **wakeup**. После получения сообщение **wakeup** по всем каналам, процесс запускает древесный волновой алгоритм, который снабжен приставкой для вычисления наименьшего отличительного признака.

Когда процесс принимает решение, он знает имя лидера; если это имя совпадает с отличительным признаком самого процесса, то он становится лидером, а в противном случае он получает статус **lost**.

Выборы лидера на дереве

var ws_p : **bool** **init** false ;
wr_p : **integer** **init** 0 ;
$rec_p[q]$: **bool** для каждого $q \in Neigh_p$ **init** false ;
v_p : $\mathcal{P}$ **init** p ;
$state_p$: ($sleep$, $leader$, $lost$) **init sleep** ;

(* Побудка *)
begin if p is initiator **then**
begin $ws_p := true$;
forall $q \in Neigh_p$ **do** send $\langle$**wakeup**$\rangle$ to q
end;
while $wr_p < \#Neigh_p$ **do**
begin receive $\langle$**wakeup**$\rangle$; $wr_p := wr_p + 1$;
if not ws_p **then**
begin $ws_p := true$;
forall $q \in Neigh_p$ **do** send $\langle$**wakeup**$\rangle$ to q
end
end;

Выборы лидера на дереве

(* Здесь начинает работать древесный алгоритм *)
while $\#\{q : \neg rec_p[q]\} > 1$ **do**
begin
receive $\langle \mathbf{tok}, r \rangle$ from q ; $rec_p[q] := true$; $v_p := \min(v_p, r)$
end;
send $\langle \mathbf{tok}, v_p \rangle$ to q_0 with $\neg rec_p[q_0]$;
receive $\langle \mathbf{tok}, r \rangle$ from q_0 ;
$v_p := \min(v_p, r)$; (* принять решение с ответом v_p *)
if $v_p = p$ **then** $state_p := \mathbf{leader}$ **else** $state_p := \mathbf{lost}$;
forall $q \in Neigh_p$, $q \neq q_0$ **do** send $\langle \mathbf{tok}, v_p \rangle$ to q
end

Выборы лидера на дереве

Теорема 8.1.

Предложенный алгоритм решает задачу о выборах на древесных сетях с использованием $O(N)$ обменов сообщениями, затрачивая при этом $O(D)$ единиц времени.

Выборы лидера на дереве

Теорема 8.1.

Предложенный алгоритм решает задачу о выборах на древесных сетях с использованием $O(N)$ обменов сообщениями, затрачивая при этом $O(D)$ единиц времени.

Доказательство.

Когда хоть один процесс запускает данный алгоритм, все процессы отправят сообщение $\langle$**wakeup**$\rangle$ всем своим соседям, и каждый процесс запустит выполнение древесного алгоритма после получения сообщения $\langle$**wakeup**$\rangle$ от каждого соседа. Все процессы завершат выполнение древесного алгоритма с одним и тем же значением переменной v , а именно, наименьшим отличительным признаком среди всех процессов (почему?). И единственный процесс с таким признаком завершит работу в состоянии **leader**, а все прочие процессы — в состоянии **lost**.

Выборы лидера на дереве

Доказательство.

По каждому каналу передаются два сообщения $\langle \mathbf{wakeup} \rangle$ и два сообщения $\langle \mathbf{tok}, r \rangle$, и поэтому сложность по числу обменов сообщениями становится равной $4N - 4$.

Выборы лидера на дереве

Доказательство.

По каждому каналу передаются два сообщения $\langle \mathbf{wakeup} \rangle$ и два сообщения $\langle \mathbf{tok}, r \rangle$, и поэтому сложность по числу обменов сообщениями становится равной $4N - 4$.

По истечении D единиц времени после того, как первый процесс запустил наш алгоритм, каждый процесс уже отправит сообщения $\langle \mathbf{wakeup} \rangle$ и, следовательно, спустя $D + 1$ единиц времени каждый процесс уже запустит волну.

Выборы лидера на дереве

Доказательство.

По каждому каналу передаются два сообщения $\langle \textbf{wakeup} \rangle$ и два сообщения $\langle \textbf{tok}, r \rangle$, и поэтому сложность по числу обменов сообщениями становится равной $4N - 4$.

По истечении D единиц времени после того, как первый процесс запустил наш алгоритм, каждый процесс уже отправит сообщения $\langle \textbf{wakeup} \rangle$ и, следовательно, спустя $D + 1$ единиц времени каждый процесс уже запустит волну.

Первое решение будет принято спустя D единиц времени после начала движения волны, а последнее решение будет принято спустя D единиц времени после первого, и поэтому общее затраченное время становится равным $3D + 1$. $\square$

Волновые алгоритмы избрания лидера

Задачи.

1. Показать, что сложность по времени алгоритма избрания лидера на дереве составляет $2D$.
2. Каким образом можно модифицировать алгоритм избрания лидера на дереве, чтобы число обменов сообщениями существенно уменьшилось?
3. Как можно ли провести выборы лидера в произвольной сети при помощи фазового алгоритма? Какова будет сложность такого алгоритма избрания лидера по числу обменов сообщениями?
4. Как можно ли провести выборы лидера в произвольной сети при помощи алгоритма Финна? Какова будет сложность такого алгоритма избрания лидера по числу обменов сообщениями?

Выборы лидера в кольцах

Задача о выборах в кольцевых сетях была впервые поставлена Ле-Ланном в 1977; ему удалось найти ее решение со сложностью $O(N^2)$ по числу обменов сообщениями.

Выборы лидера в кольцах

Задача о выборах в кольцевых сетях была впервые поставлена Ле-Ланном в 1977; ему удалось найти ее решение со сложностью $O(N^2)$ по числу обменов сообщениями.
Это решение было улучшено Ченем и Робертсом в 1979; они предложили алгоритм, который имеет сложность $O(N^2)$ в наихудшем случае, и сложность $O(N \log N)$ в среднем.

Выборы лидера в кольцах

Задача о выборах в кольцевых сетях была впервые поставлена Ле-Ланном в 1977; ему удалось найти ее решение со сложностью $O(N^2)$ по числу обменов сообщениями.
Это решение было улучшено Ченем и Робертсом в 1979; они предложили алгоритм, который имеет сложность $O(N^2)$ в наихудшем случае, и сложность $O(N \log N)$ в среднем.
Вопрос о существовании алгоритма со сложностью $O(N \log N)$ в наихудшем случае оставался открытым до 1980, когда такой алгоритм был предложен Хиршбергом и Синклейром двунаправленных колец.

Выборы лидера в кольцах

Задача о выборах в кольцевых сетях была впервые поставлена Ле-Ланном в 1977; ему удалось найти ее решение со сложностью $O(N^2)$ по числу обменов сообщениями.
Это решение было улучшено Ченем и Робертсом в 1979; они предложили алгоритм, который имеет сложность $O(N^2)$ в наихудшем случае, и сложность $O(N \log N)$ в среднем.
Вопрос о существовании алгоритма со сложностью $O(N \log N)$ в наихудшем случае оставался открытым до 1980, когда такой алгоритм был предложен Хиршбергом и Синклейром двунаправленных колец. Какое-то время предполагалось, что $\Omega(N^2)$ обменов сообщениями составляют нижнюю оценку для однонаправленных колец, но Петерсон, а также Долев, Клейв и Роде в 1982 независимо предложили решение сложности $O(N \log N)$ для однонаправленного кольца.

Выборы лидера в кольцах

Задача о выборах в кольцевых сетях была впервые поставлена Ле-Ланном в 1977; ему удалось найти ее решение со сложностью $O(N^2)$ по числу обменов сообщениями.
Это решение было улучшено Ченем и Робертсом в 1979; они предложили алгоритм, который имеет сложность $O(N^2)$ в наихудшем случае, и сложность $O(N \log N)$ в среднем.
Вопрос о существовании алгоритма со сложностью $O(N \log N)$ в наихудшем случае оставался открытым до 1980, когда такой алгоритм был предложен Хиршбергом и Синклейром двунаправленных колец. Какое-то время предполагалось, что $\Omega(N^2)$ обменов сообщениями составляют нижнюю оценку для однонаправленных колец, но Петерсон, а также Долев, Клейв и Роде в 1982 независимо предложили решение сложности $O(N \log N)$ для однонаправленного кольца. Нижняя оценка $\approx 0.34 N \log N$ сложности по числу обменов сообщениями в наихудшем случае для двунаправленных колец была обоснована Бодлаендэром в 1988.

Алгоритм Ле-Ланна

Каждый инициатор вычисляет список отличительных признаков всех инициаторов, после чего лидером избирается инициатор с наименьшим признаком.

Алгоритм Ле-Ланна

Каждый инициатор вычисляет список отличительных признаков всех инициаторов, после чего лидером избирается инициатор с наименьшим признаком.

Каждый инициатор отправляет по кольцу маркер со своим отличительным признаком, и все процессы передают далее этот маркер. Каналы поддерживают очередность сообщений.

Алгоритм Ле-Ланна

Каждый инициатор вычисляет список отличительных признаков всех инициаторов, после чего лидером избирается инициатор с наименьшим признаком.

Каждый инициатор отправляет по кольцу маркер со своим отличительным признаком, и все процессы передают далее этот маркер. Каналы поддерживают очередность сообщений.

Когда инициатор p получает свой собственный маркер обратно, маркеры всех инициаторов уже прошли через p , и p будет избран лидером в том и только том случае, если p — наименьший процесс среди всех инициаторов.

Алгоритм Ле-Ланна

var $List_p$: множество имен $\mathcal{P}$ **init** $\{p\}$;
$state_p$;

begin if p is initiator **then**
begin $state_p := \textbf{cand}$; send $\langle \textbf{tok}, p \rangle$ to $Next_p$; receive $\langle \textbf{tok}, q \rangle$;
while $q \neq p$ **do**
begin $List_p := List_p \cup \{q\}$;
send $\langle \textbf{tok}, q \rangle$ to $Next_p$; receive $\langle \textbf{tok}, q \rangle$
end;
if $p = \min(List_p)$ **then** $state_p := \textbf{leader}$
else $state_p := \textbf{lost}$
end
else while *true* **do**
begin receive $\langle \textbf{tok}, q \rangle$; send $\langle \textbf{tok}, q \rangle$ to $Next_p$;
if $state_p = \textbf{sleep}$ **then** $state_p := \textbf{lost}$
end
end

Алгоритм Ле-Ланна

Теорема 8.2.

Алгоритм Ле-Ланна решает задачу о выборах на кольцах с использованием $O(N^2)$ обменов сообщениями за $O(N)$ единиц времени.

Алгоритм Ле-Ланна

Теорема 8.2.

Алгоритм Ле-Ланна решает задачу о выборах на кольцах с использованием $O(N^2)$ обменов сообщениями за $O(N)$ единиц времени.

Доказательство.

Так как порядок следования маркеров по кольцу остается неизменным (почему?), и инициатор q отправляет $\langle \mathbf{tok}, q \rangle$ раньше , чем q получает $\langle \mathbf{tok}, p \rangle$, инициатор p получает $\langle \mathbf{tok}, q \rangle$ до того, как p получит $\langle \mathbf{tok}, p \rangle$ обратно.
Отсюда следует, что каждый инициатор p завершает работу со списком $List_p$ имен всех инициаторов, и инициатор с наименьшим отличительным признаком становится лидером.

Алгоритм Ле-Ланна

Доказательство.

Всего используется не более N различных маркеров, и каждый из них совершает N шагов, что приводит к сложности $O(N^2)$ по числу обменов сообщениями.

Алгоритм Ле-Ланна

Доказательство.

Всего используется не более N различных маркеров, и каждый из них совершает N шагов, что приводит к сложности $O(N^2)$ по числу обменов сообщениями.

Спустя самое позднее $N - 1$ единиц времени после того, как первый инициатор отправил свой маркер, каждый инициатор проделает тоже самое.

При этом каждый инициатор получит свой маркер обратно в течение N единиц времени после создания этого маркера. Значит, наш алгоритм завершит работу не позднее, чем спустя $2N - 1$ единиц времени. □

Алгоритм Ченя – Робертса

Алгоритм Ченя и Робертса улучшает алгоритм Ле-Ланна за счет того, что из кольца изымаются все маркеры тех процессов, относительно которых уже становится ясно, что они проиграют выборы.

А именно, инициатор p изымает маркер $\langle \mathbf{tok}, q \rangle$ из кольца, если $q > p$. Инициатор p обретает статус **lost** , как только получает маркер с отличительным признаком $q < p$, и статус **leader**, как только получает маркер с отличительным признаком p .

Алгоритм Ченя – Робертса

```
var state_p ;

begin if p is initiator then
      begin state_p := cand ; send ⟨tok, p⟩ to Next_p ;
            while state_p ≠ leader do
                  begin receive ⟨tok, q⟩ ;
                        if q = p then state_p := leader
                        else if q < p then
                             begin if state_p = cand then state_p := lost;
                                   send ⟨tok, q⟩ to Next_p end
                  end
      end
      else while true do
           begin receive ⟨tok, q⟩ ; send ⟨tok, q⟩ to Next_p ;
                 if state_p = sleep then state_p := lost
           end
end
```

Алгоритм Ченя – Робертса

Теорема 8.3.

Алгоритм Ченя – Робертса решает задачу о выборах на кольцах с использованием $\Theta(N^2)$ обменов сообщениями в наихудшем случае и за $O(N)$ единиц времени.

Алгоритм Ченя – Робертса

Теорема 8.3.

Алгоритм Ченя – Робертса решает задачу о выборах на кольцах с использованием $\Theta(N^2)$ обменов сообщениями в наихудшем случае и за $O(N)$ единиц времени.

Доказательство.

Обозначим p_0 инициатора с наименьшим именем.
Всякий другой процесс может быть либо не-инициатором, либо инициатором с отличительным признаком, превышающим p_0 , и поэтому все процессы передадут далее маркер $\langle \mathbf{tok}, p_0 \rangle$ выпущенный p_0 . Значит, p_0 получит свой маркер обратно и будет избран лидером.

Алгоритм Ченя – Робертса

Доказательство.

Не-инициаторы не будут избраны, но все они перейдут в состояние **lost** самое позднее к моменту передачи маркера, который выпустил p_0 .

Инициатор p , для которого $p > p_0$, не станет лидером, т.к. p_0 не передаст далее маркер $\langle \mathbf{tok}, p \rangle$, и поэтому p никогда не получит свой собственный маркер.

Такой инициатор p перейдет в состояние **lost** самое позднее в тот момент, когда будет передавать далее $\langle \mathbf{tok}, p_0 \rangle$.

Это и служит обоснованием того, что наш алгоритм решает задачу о выборах.

Алгоритм Ченя – Робертса

Доказательство.

В алгоритме задействовано не более N различных маркеров, и каждый маркер передается не более N раз; этим и обосновывается оценка $O(N^2)$ сложности по числу обменов сообщениями.

Алгоритм Ченя – Робертса

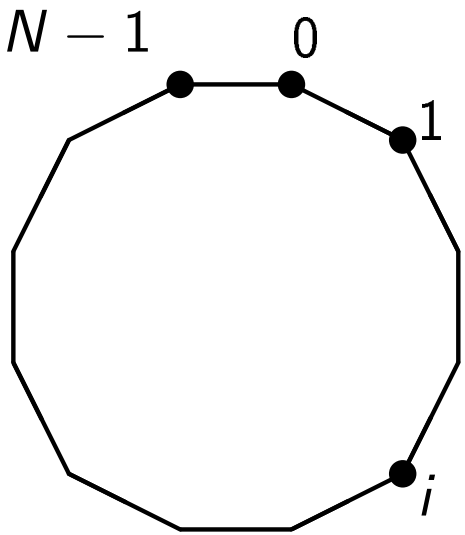
Доказательство.

В алгоритме задействовано не более N различных маркеров, и каждый маркер передается не более N раз; этим и обосновывается оценка $O(N^2)$ сложности по числу обменов сообщениями.

Чтобы убедиться в том, что может иногда может понадобиться передать $\Omega(N^2)$ сообщений, рассмотрим начальную конфигурацию, в которой отличительные признаки расположены в кольце по возрастанию, и каждый процесс является инициатором.

Алгоритм Ченя – Робертса

Доказательство.



Сообщения передаются по часовой стрелке

Алгоритм Ченя – Робертса

Доказательство.

В алгоритме задействовано не более N различных маркеров, и каждый маркер передается не более N раз; этим и обосновывается оценка $O(N^2)$ сложности по числу обменов сообщениями.

Чтобы убедиться в том, что может иногда может понадобиться передать $\Omega(N^2)$ сообщений, рассмотрим начальную конфигурацию, в которой отличительные признаки расположены в кольце по возрастанию, и каждый процесс является инициатором.

Маркер каждого процесса изымается из кольца процессом 0 , и поэтому маркер процесса i совершает $N - i$ переходов; это приводит к тому, что число передач сообщений будет равно $\sum_{i=0}^{N-1} (N - i) = \frac{1}{2} N(N + 1)$.

Алгоритм Ченя – Робертса

Теорема 8.4.

Алгоритму Ченя–Робертса в среднем требуется всего лишь $\approx 0.69N \log N$ обменов сообщениями, когда все процессы являются инициаторами.

Алгоритм Ченя – Робертса

Теорема 8.4.

Алгоритму Ченя–Робертса в среднем требуется всего лишь $\approx 0.69N \log N$ обменов сообщениями, когда все процессы являются инициаторами.

Задачи.

1. Зависит ли корректность алгоритма Ченя–Робертса от очередности передачи сообщений в каналах?
2. Рассмотрим алгоритм Ченя–Робертса, полагая, что каждый процесс является инициатором. При каком расположении отличительных признаков в кольце сложность по числу обменов сообщениями будет минимальной, и сколько обменов сообщениями потребуется в этом случае?

Алгоритм Петерсона/Долева–Клейва–Роде

В данном алгоритме требуется, чтобы в каналах поддерживалась очередность сообщений.

Вначале алгоритм вычисляет наименьший отличительный признак и доводит его до сведения всех процессов; затем процесс с указанным отличительным признаком становится лидером, а все остальные процессы признают свое поражение на выборах.

Алгоритм Петерсона/Долева–Клейва–Роде

В данном алгоритме требуется, чтобы в каналах поддерживалась очередность сообщений.

Вначале алгоритм вычисляет наименьший отличительный признак и доводит его до сведения всех процессов; затем процесс с указанным отличительным признаком становится лидером, а все остальные процессы признают свое поражение на выборах.

Суть этого алгоритма будет проще понять, если взглянуть на него так, как будто алгоритм выполняют не сами процессы, а их отличительные признаки .

Алгоритм Петерсона/Долева–Клейва–Роде

Первоначально каждый отличительный признак является активным , но в каждом туре некоторые отличительные признаки становятся пассивными .

Алгоритм Петерсона/Долева–Клейва–Роде

Первоначально каждый отличительный признак является активным , но в каждом туре некоторые отличительные признаки становятся пассивными .

Вычисление разбито на туры. В каждом туре всякий активный отличительный признак сравнивается с двумя соседними активными отличительными признаками, расположенными по ходу и против хода часовой стрелки. Если этот признак будет минимальным из трех, то он переходит в следующий тур, а иначе он становится пассивным .

Алгоритм Петерсона/Долева–Клейва–Роде

Первоначально каждый отличительный признак является активным , но в каждом туре некоторые отличительные признаки становятся пассивными .

Вычисление разбито на туры. В каждом туре всякий активный отличительный признак сравнивается с двумя соседними активными отличительными признаками, расположенными по ходу и против хода часовой стрелки. Если этот признак будет минимальным из трех, то он переходит в следующий тур, а иначе он становится пассивным .

Так как все отличительные признаки попарно различны, отличительные признаки, расположенные по обе стороны от локального минимума, станут пассивными, и поэтому, по крайней мере, половина отличительных признаков не перейдет в следующий тур.

Алгоритм Петерсона/Долева–Клейва–Роде

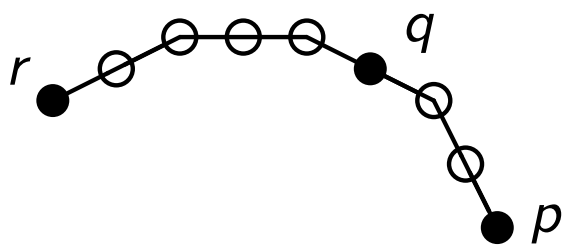
Первоначально каждый отличительный признак является активным , но в каждом туре некоторые отличительные признаки становятся пассивными .

Вычисление разбито на туры. В каждом туре всякий активный отличительный признак сравнивается с двумя соседними активными отличительными признаками, расположенными по ходу и против хода часовой стрелки. Если этот признак будет минимальным из трех, то он переходит в следующий тур, а иначе он становится пассивным .

Так как все отличительные признаки попарно различны, отличительные признаки, расположенные по обе стороны от локального минимума, станут пассивными, и поэтому, по крайней мере, половина отличительных признаков не перейдет в следующий тур.

Следовательно, спустя самое большее $\log N$ туров, останется только один активный отличительный признак, который и будет признан победителем.

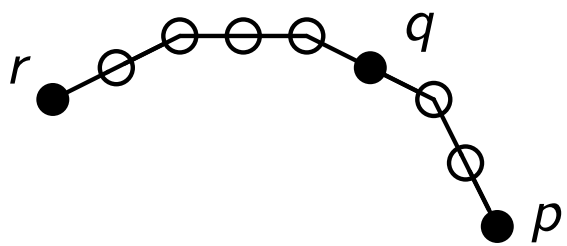
Алгоритм Петерсона/Долева–Клейва–Роде



● Активный процесс
○ Пассивный процесс

В ориентированных кольцах сообщения можно передавать только по часовой стрелке, и это затрудняет получение первого соседнего по ходу часовой стрелки отличительного признака. Отличительный признак q необходимо сравнить с r и p ; но если признак r может быть легко передан q , то признак p вынужден был бы двигаться в направлении, противоположном ориентации каналов, чтобы достичь q .

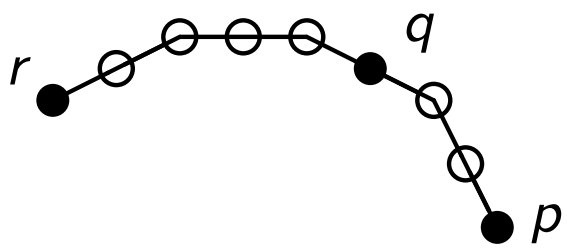
Алгоритм Петерсона/Долева–Клейва–Роде



● Активный процесс
○ Пассивный процесс

Для того, чтобы сделать возможным проведение сравнения с обоими признаками r и p , отличительный признак q передается (по направлению каналов в кольце) тому процессу, который в этот момент является хранителем p , а r переправляется не только процессу, хранящему q , но также и далее процессу, хранящему p .

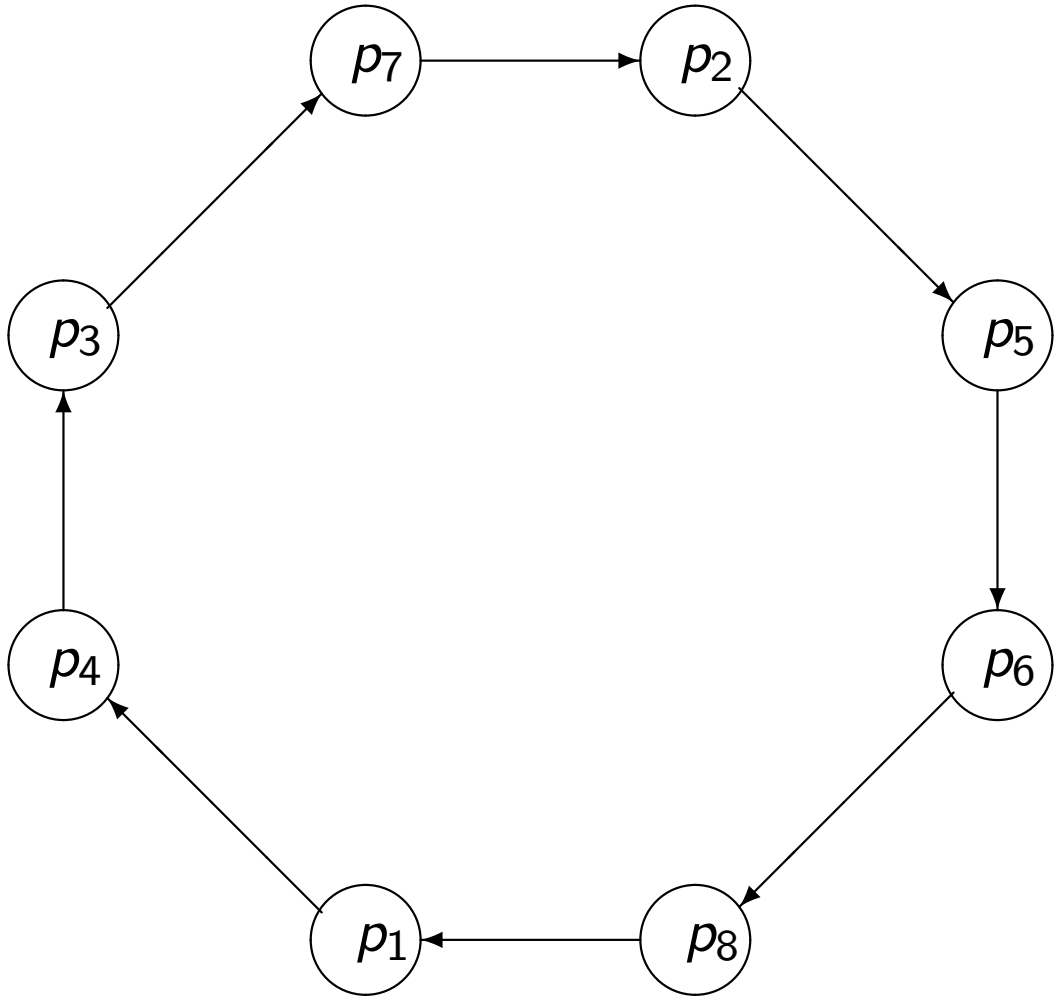
Алгоритм Петерсона/Долева–Клейва–Роде



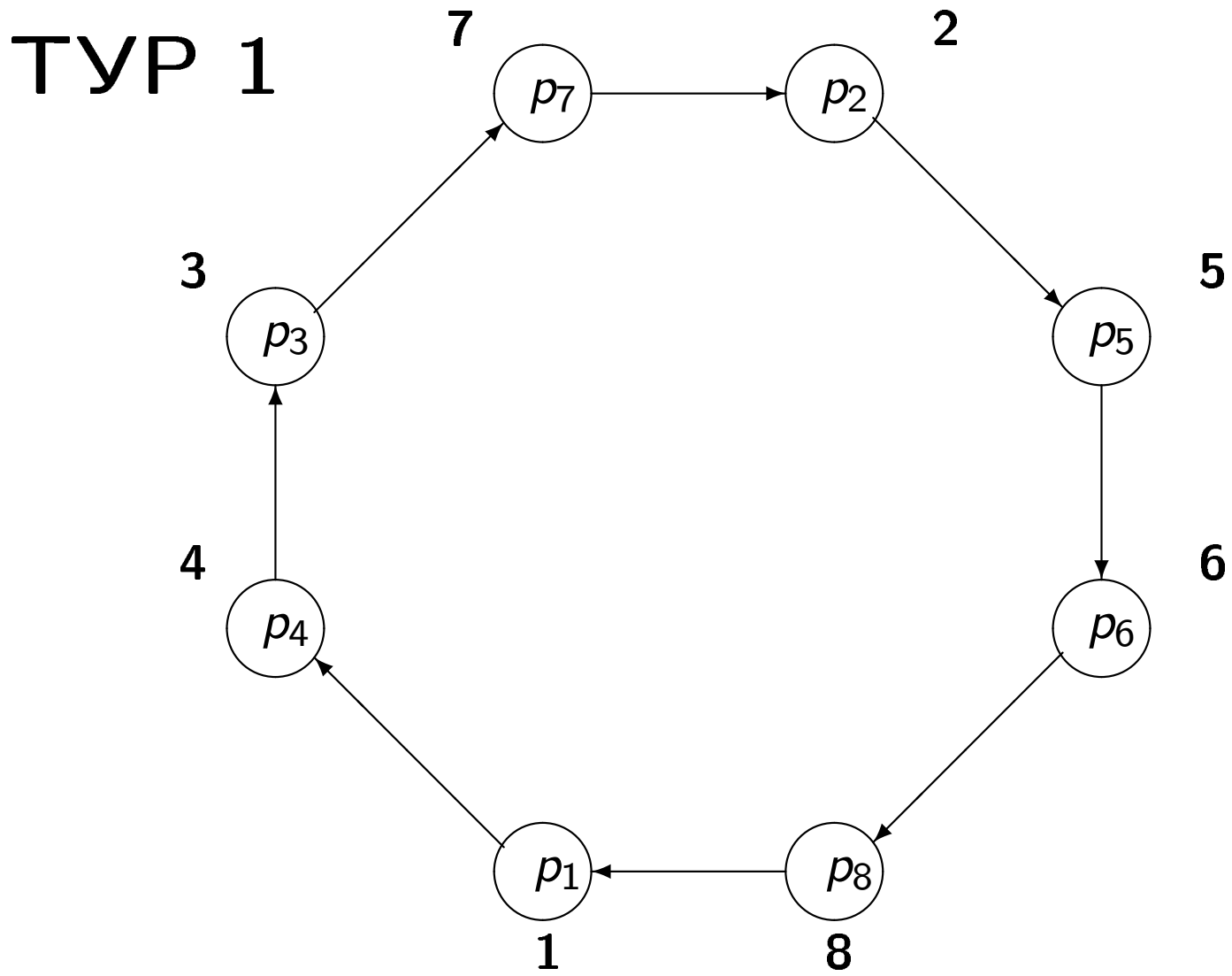
● Активный процесс
○ Пассивный процесс

Если q остается единственным активным отличительным признаком в начале некоторого тура, то первый же признак, который будет передан q в этом туре, будет равен q (т. е. в этом случае $p = q$). Как только сложится такая ситуация, данный отличительный признак становится победителем на выборах.

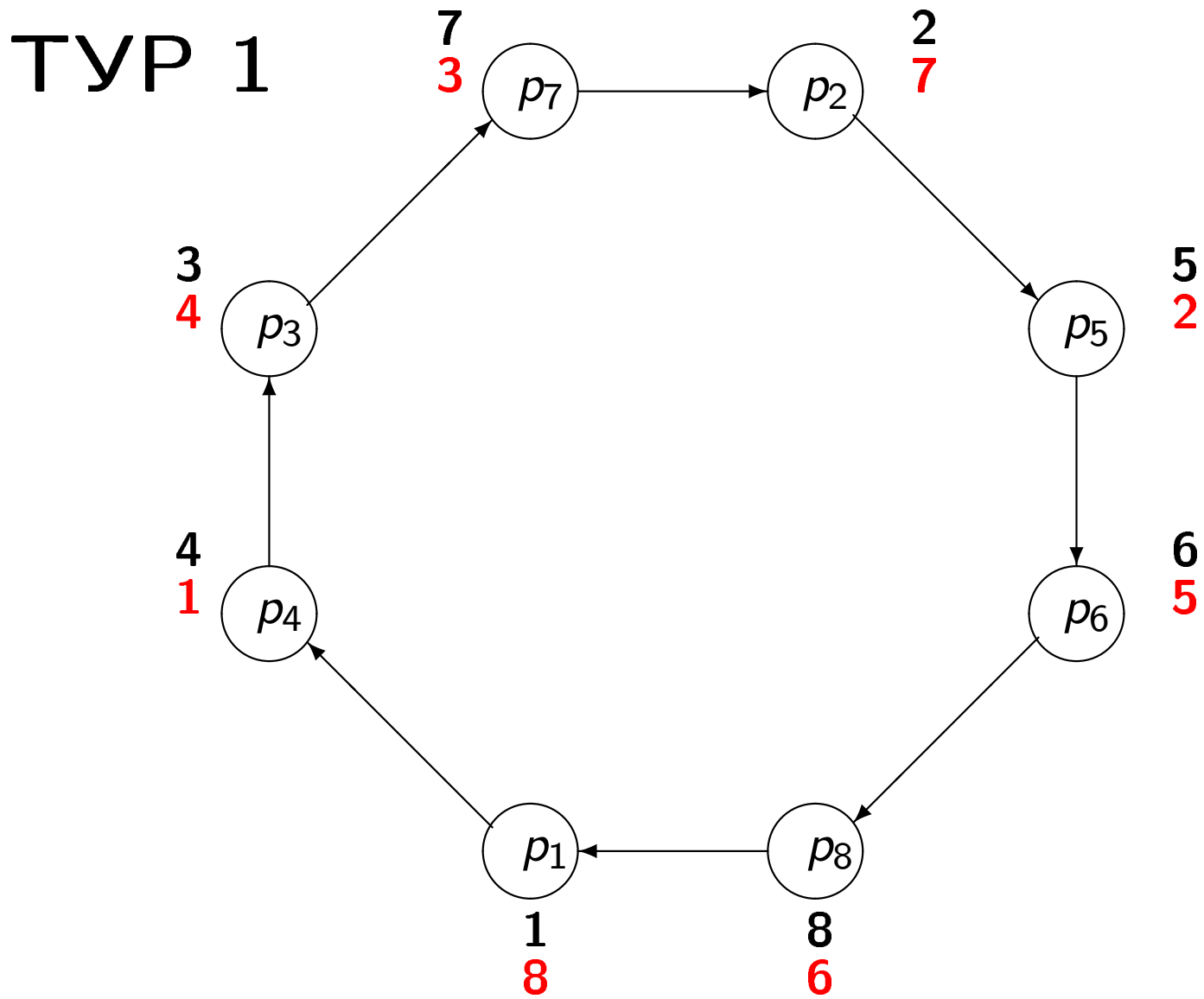
Алгоритм Петерсона/Долева–Клейва–Роде



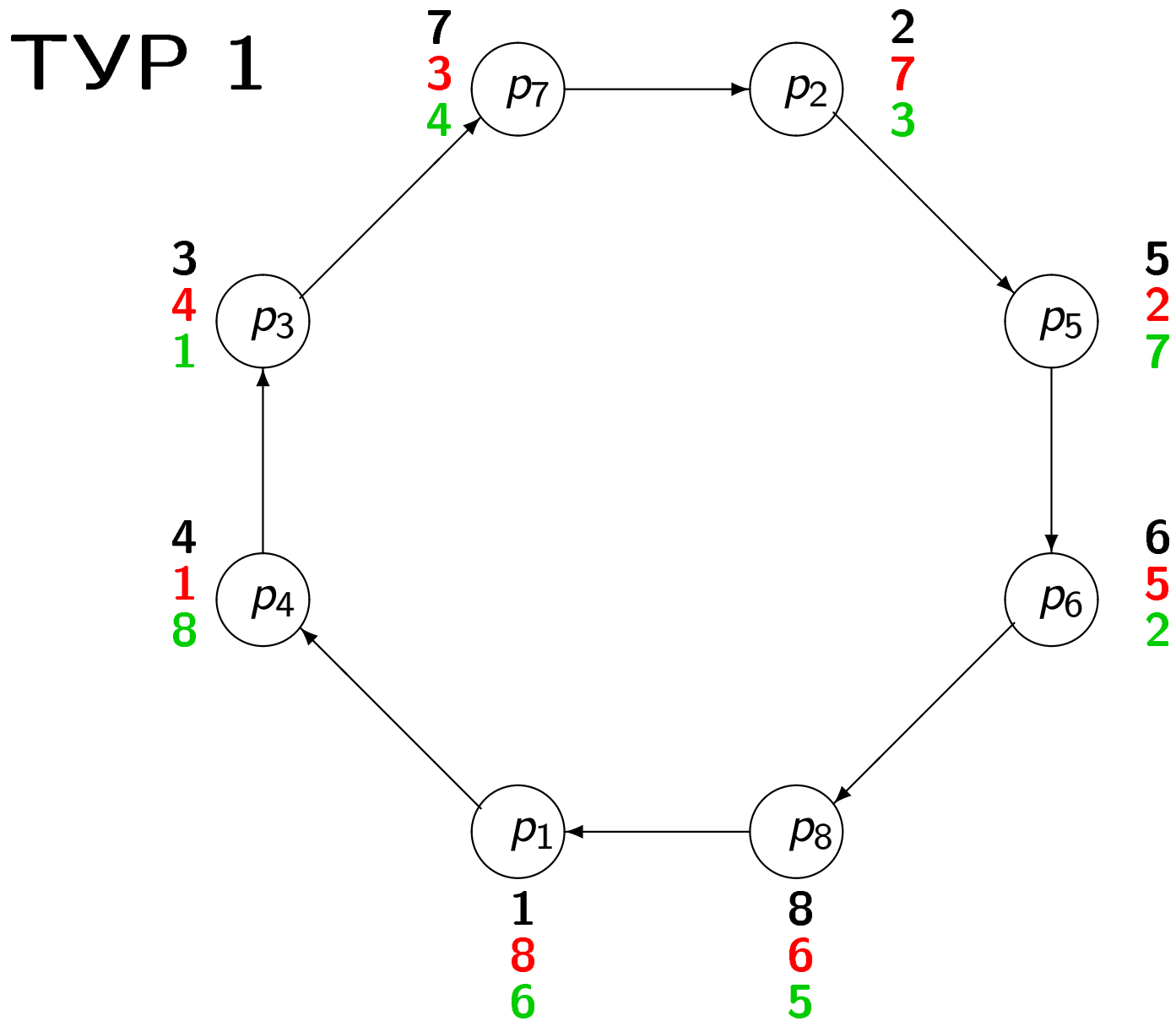
Алгоритм Петерсона/Долева–Клейва–Роде



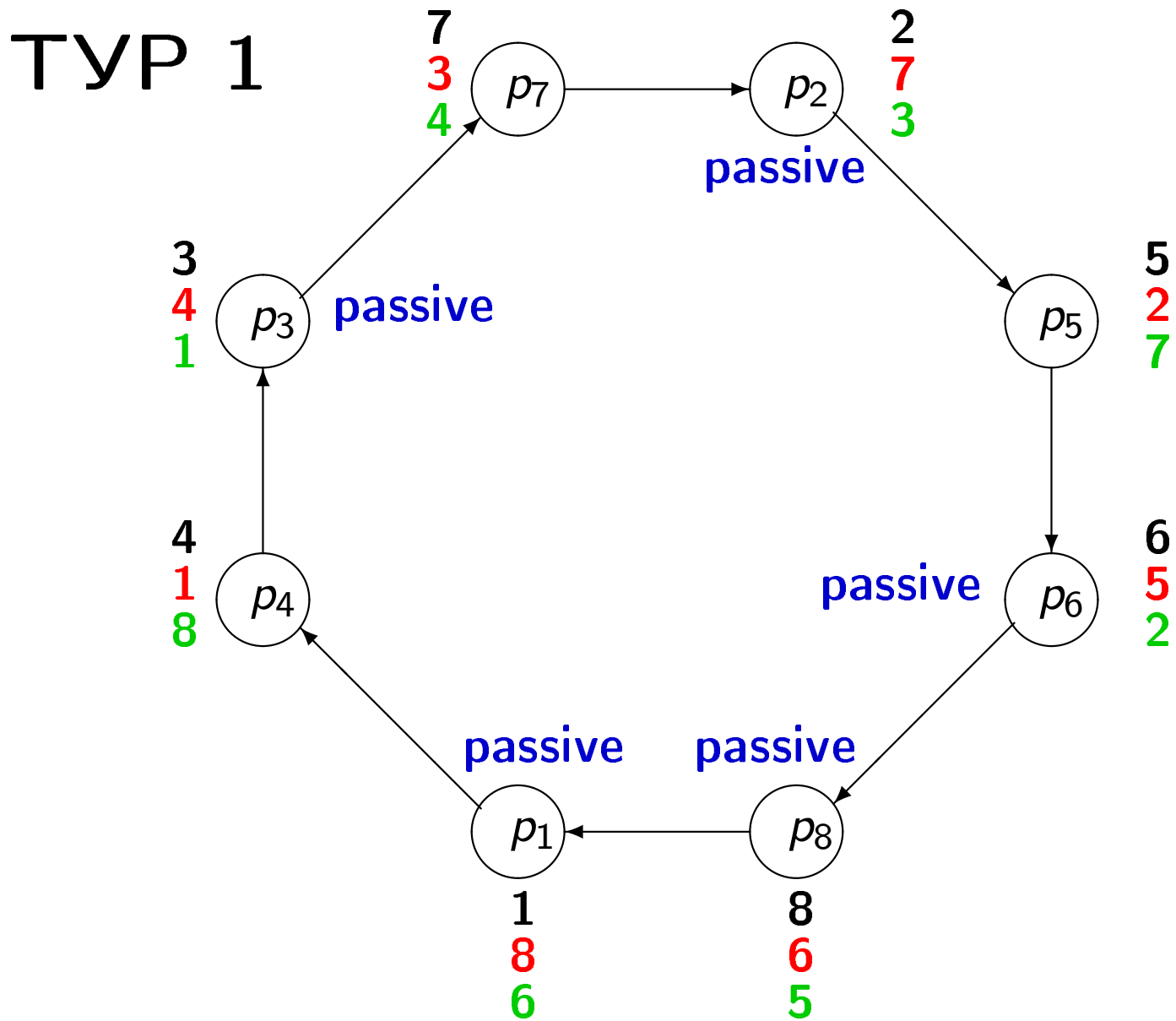
Алгоритм Петерсона/Долева–Клейва–Роде



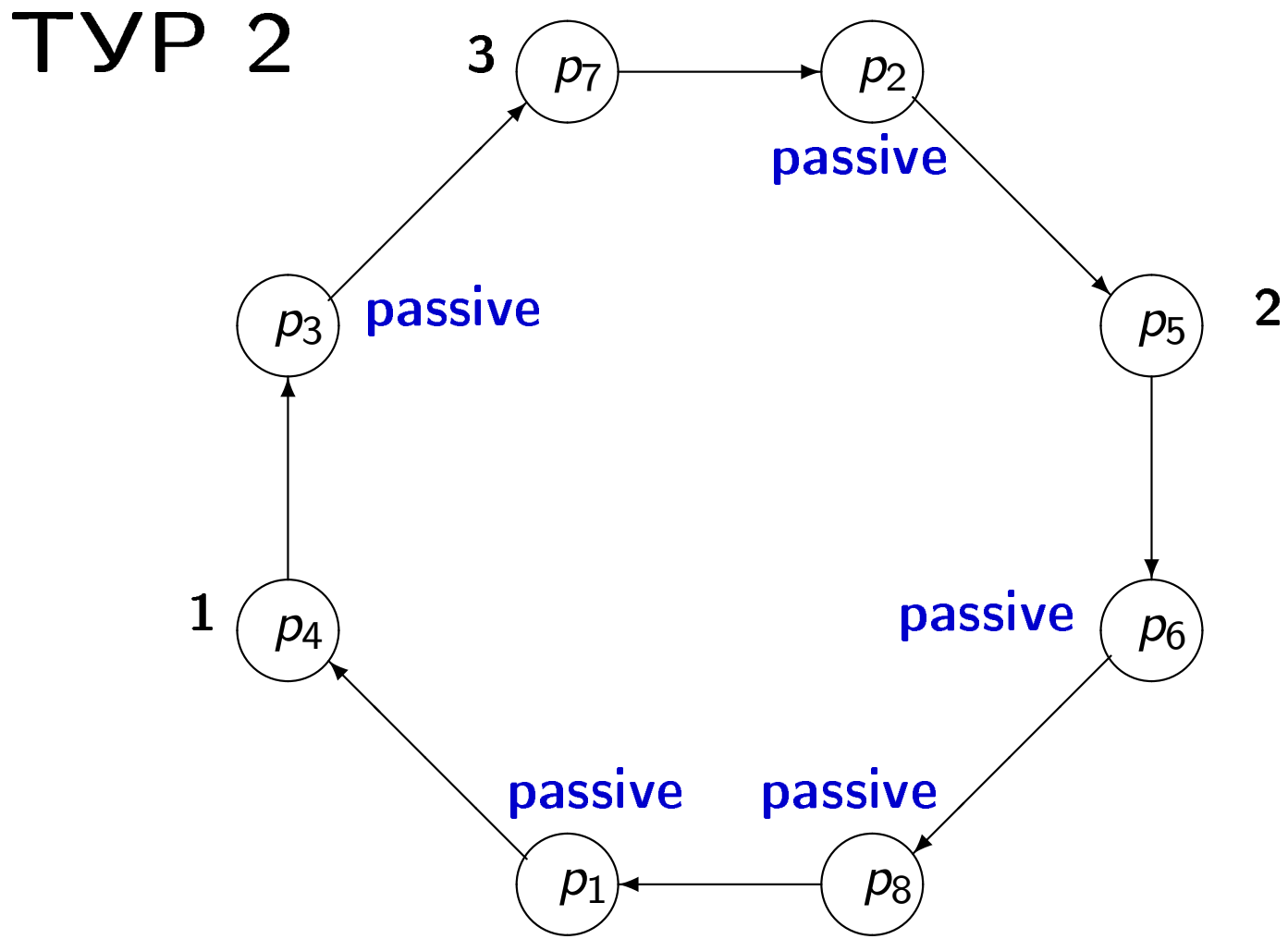
Алгоритм Петерсона/Долева–Клейва–Роде



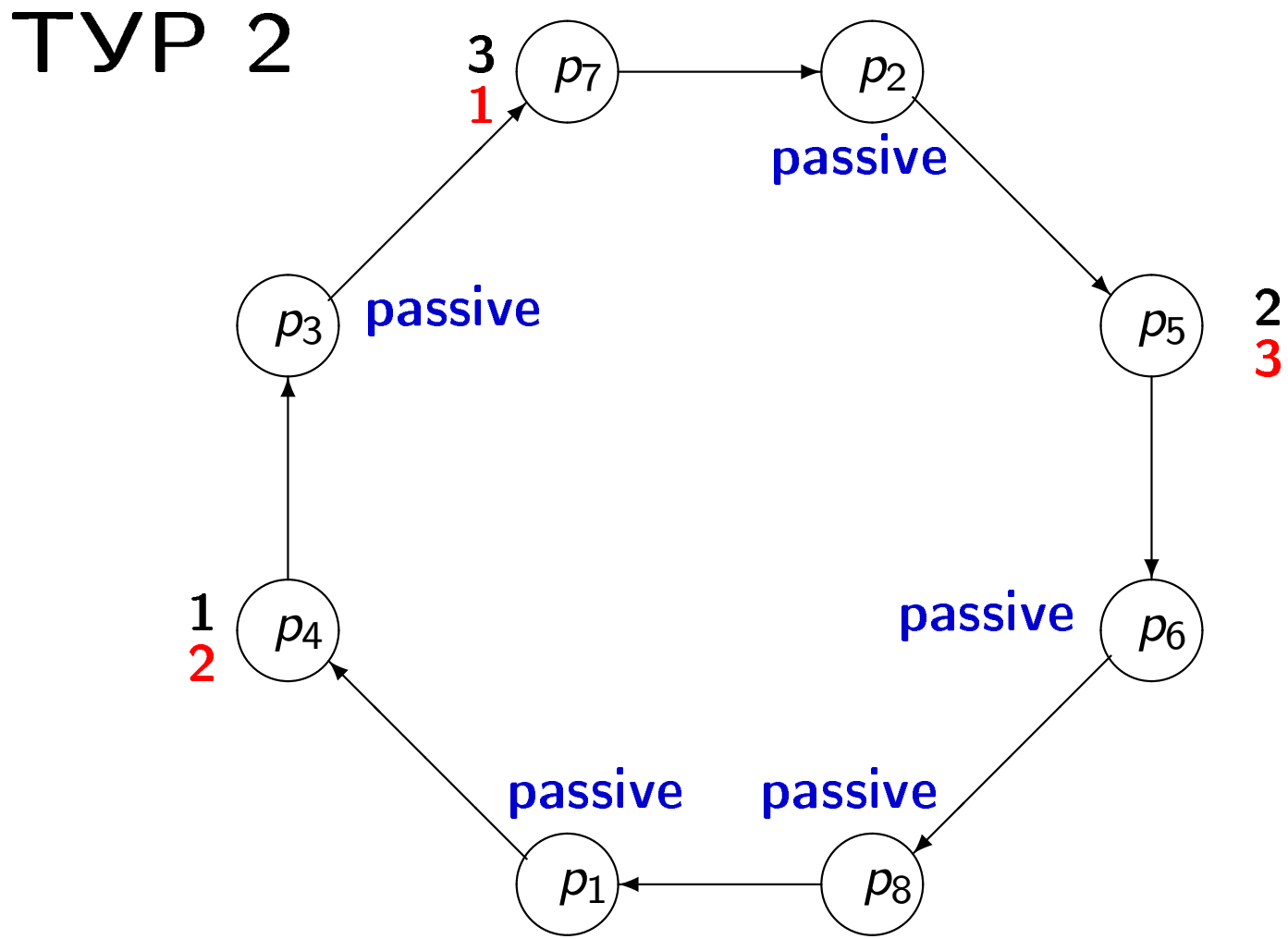
Алгоритм Петерсона/Долева–Клейва–Роде



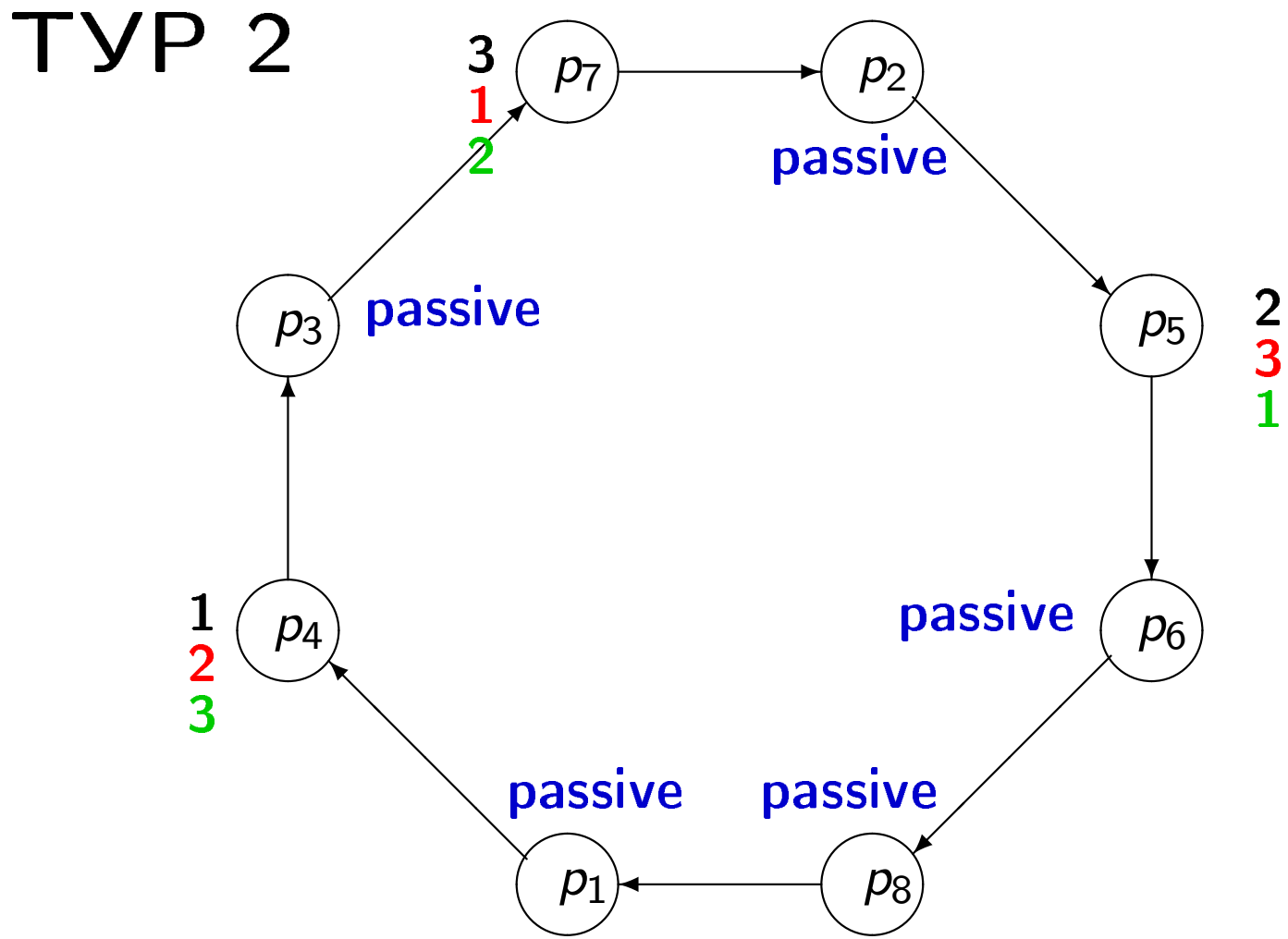
Алгоритм Петерсона/Долева–Клейва–Роде



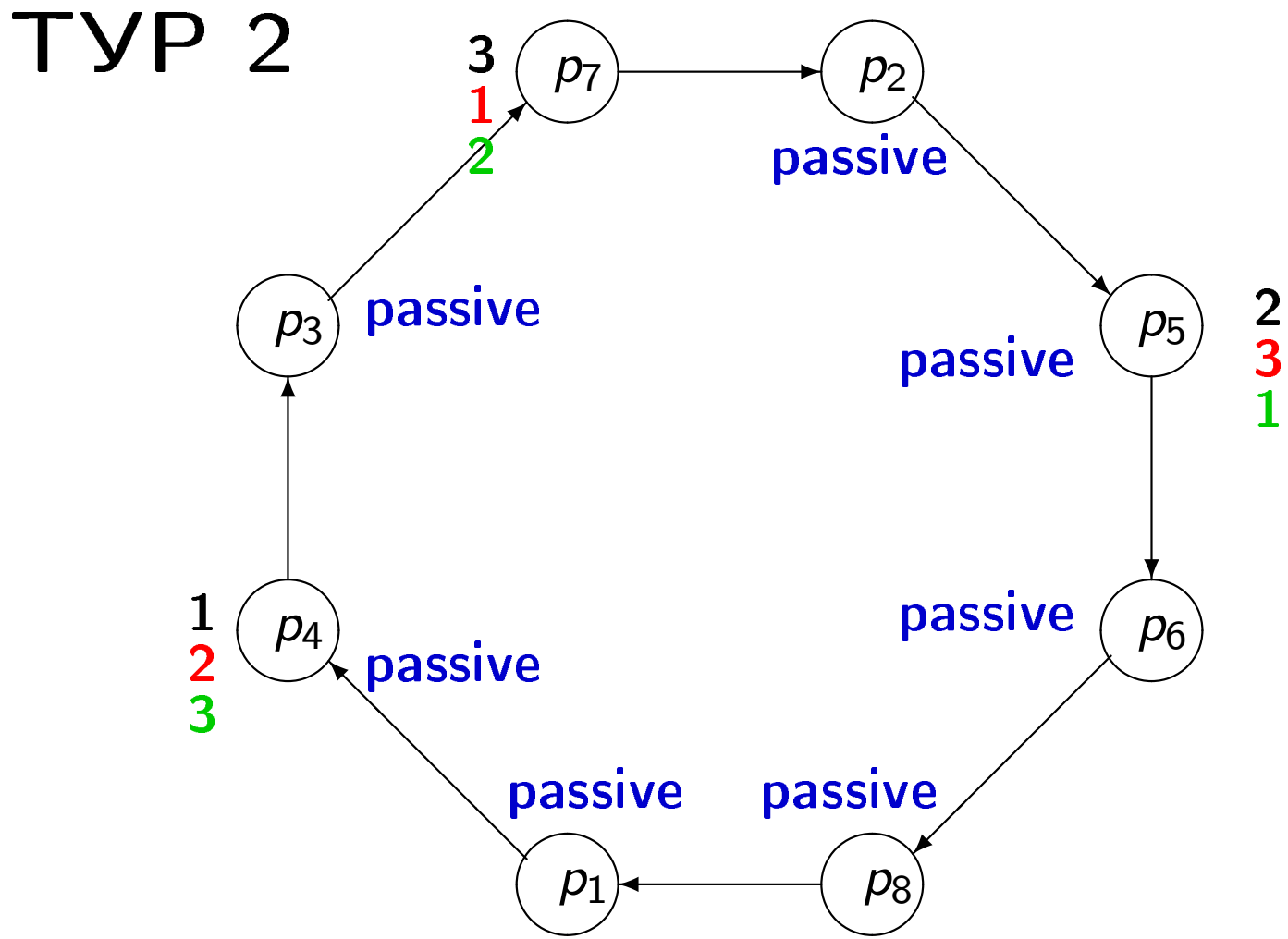
Алгоритм Петерсона/Долева–Клейва–Роде



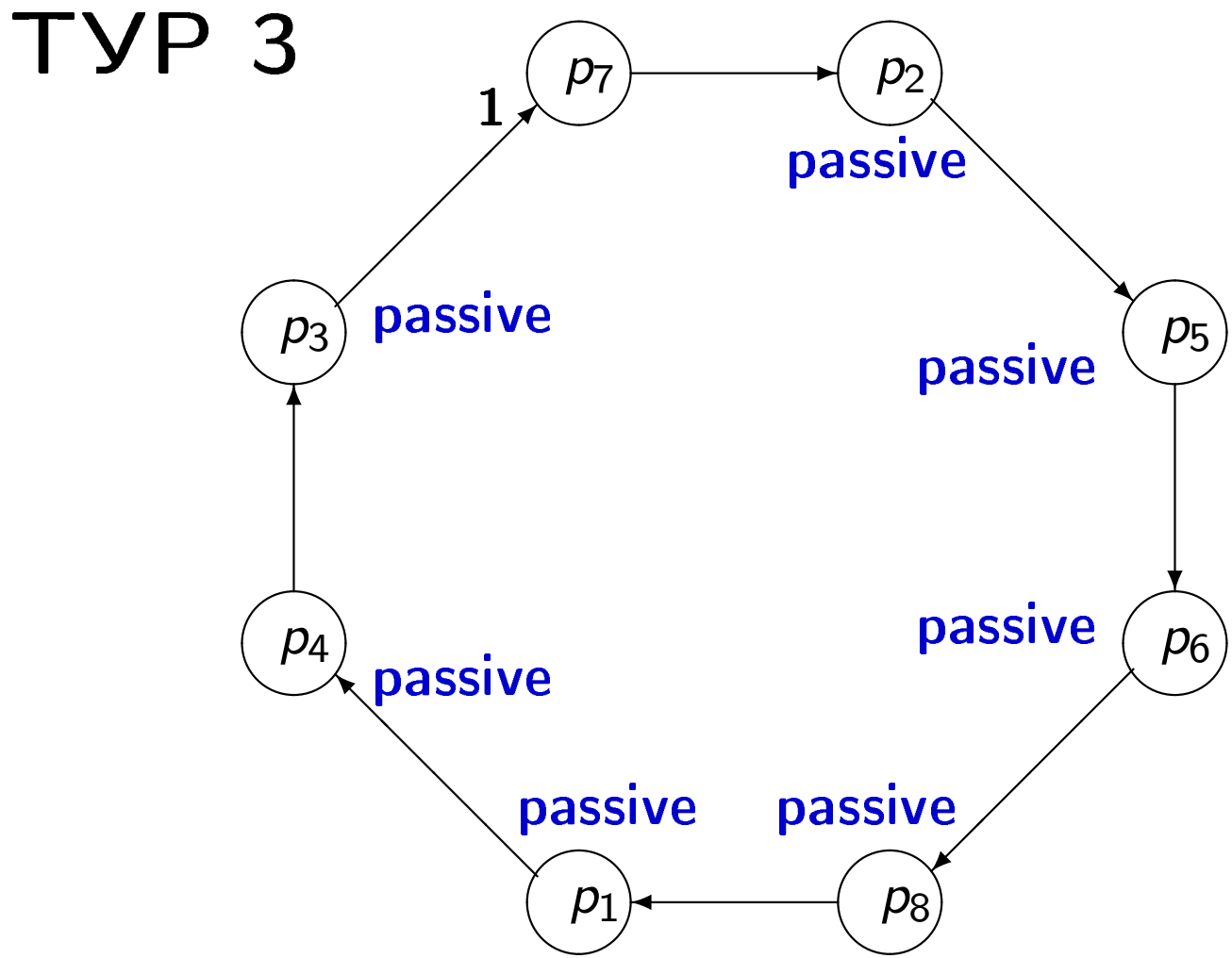
Алгоритм Петерсона/Долева–Клейва–Роде



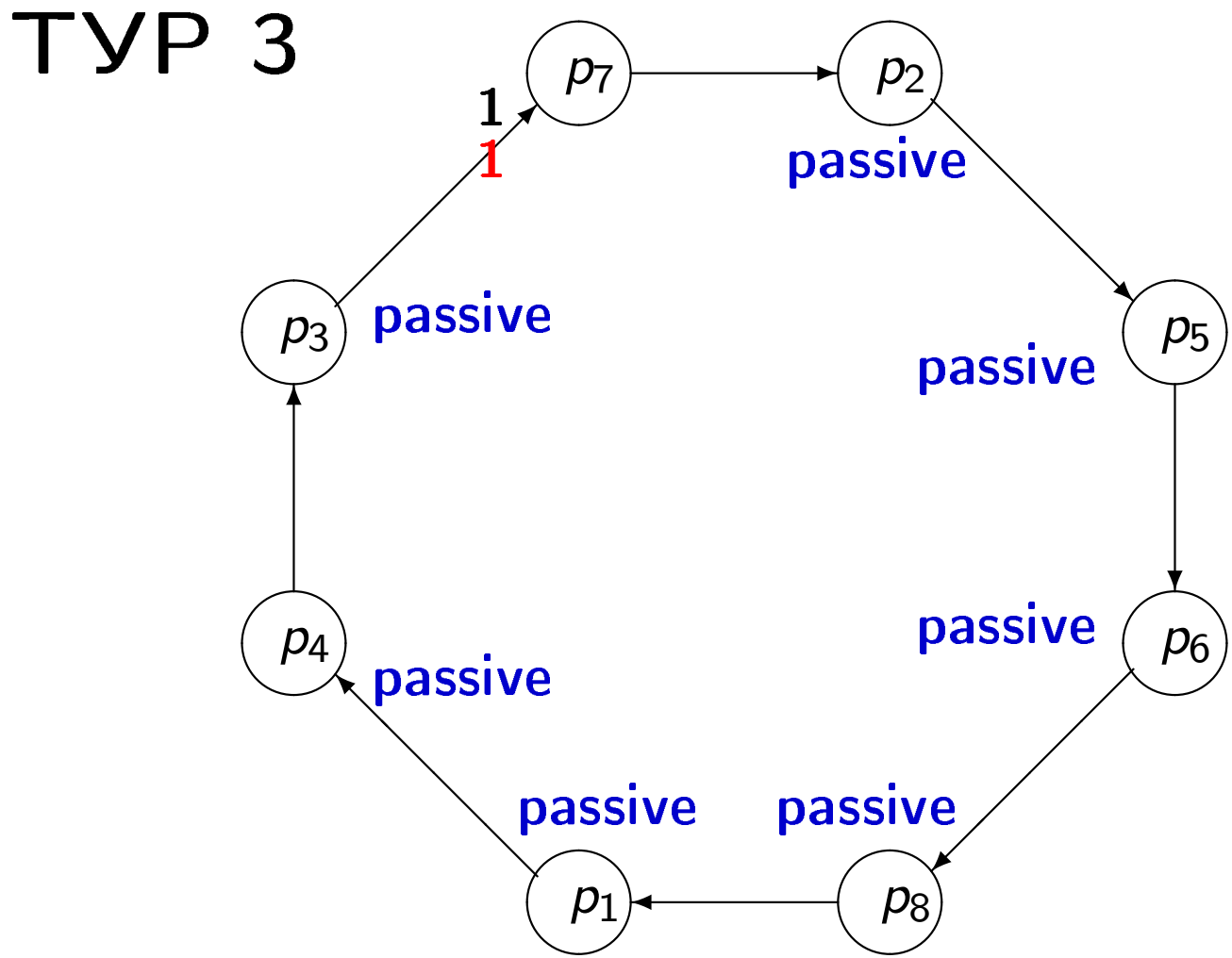
Алгоритм Петерсона/Долева–Клейва–Роде



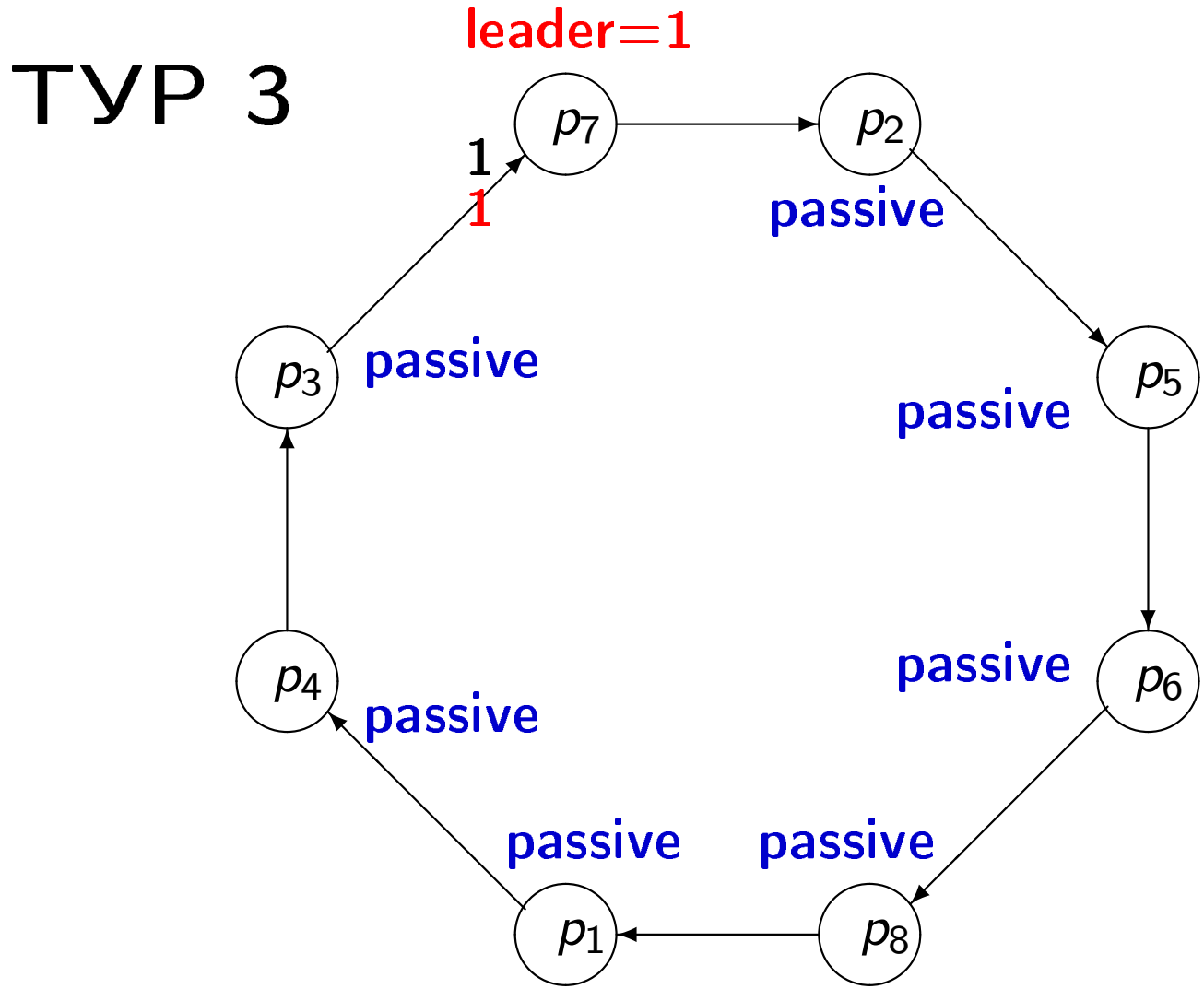
Алгоритм Петерсона/Долева–Клейва–Роде



Алгоритм Петерсона/Долева–Клейва–Роде



Алгоритм Петерсона/Долева–Клейва–Роде



Алгоритм Петерсона/Долева–Клейва–Роде

```
var ci_p    : P    init p ;        (* Текущий признак процесса p *)
    acn_p   : P    init udef ;     (* Признак активного соседа против ход
    win_p   : P    init udef ;     (* Признак победителя *)
    state_p : (active, passive, leader, lost) init active ;
begin if p is initiator then state_p := active else state_p := passive ;
      while win_p = udef do
        begin if state_p = active then
                begin send ⟨one, ci_p⟩; receive ⟨one, q⟩ ; acn_p := q;
                      if acn_p = ci_p then (* acn_p минимальный *)
                        begin send ⟨small, acn_p⟩ ; win_p := acn_p ;
                              receive ⟨small, q⟩
                        end
                      else (* acn_p — текущий признак соседа *)
                        begin send ⟨two, acn_p⟩; receive ⟨two, q⟩ ;
                              if acn_p < ci_p and acn_p < q
                                then ci_p := acn_p else state_p := passive
                              end
                end
```

Алгоритм Петерсона/Долева–Клейва–Роде

else (* $state_p = passive$ *)
    begin receive $\langle$**one**, $q\rangle$; send $\langle$**one**, $q\rangle$;
        receive m ; send m ;
        (* m — это либо $\langle$**two**, $q\rangle$, либо $\langle$**small**, $q\rangle$*)
        if m — это сообщение $\langle$**small**, $q\rangle$ **then** $win_p := q$
    end
end;
if $p = win_p$ **then** $state_p := leader$ **else** $state_p := lost$
end

Алгоритм Петерсона/Долева–Клейва–Роде

Процесс p является активным в некотором туре, если в начале тура он хранит активный отличительный признак ci_p . В противном случае p является пассивным и просто ретранслирует все сообщения, которые он получает.

Алгоритм Петерсона/Долева–Клейва–Роде

Процесс p является активным в некотором туре, если в начале тура он хранит активный отличительный признак ci_p . В противном случае p является пассивным и просто ретранслирует все сообщения, которые он получает.

Всякий активный процесс отправляет свой текущий отличительный признак следующему по порядку активному процессу и получает текущий отличительный признак от предшествующего активного процесса, используя сообщения типа $\langle \mathbf{one} \rangle$.

Полученный признак сохраняется в памяти (в качестве значения переменной acn_p).

Алгоритм Петерсона/Долева–Клейва–Роде

```
var ci_p    : P    init p ;        (* Текущий признак процесса p *)
    acn_p   : P    init udef ;     (* Признак активного соседа против ход
    win_p   : P    init udef ;     (* Признак победителя *)
    state_p : (active, passive, leader, lost) init active ;
begin if p is initiator then state_p := active else state_p := passive ;
      while win_p = udef do
        begin if state_p = active then
                begin send ⟨one, ci_p⟩; receive ⟨one, q⟩ ; acn_p := q ;
                      if acn_p = ci_p then (* acn_p минимальный *)
                        begin send ⟨small, acn_p⟩ ; win_p := acn_p ;
                              receive ⟨small, q⟩
                        end
                      else (* acn_p — текущий признак соседа *)
                        begin send ⟨two, acn_p⟩; receive ⟨two, q⟩ ;
                              if acn_p < ci_p and acn_p < q
                                then ci_p := acn_p else state_p := passive
                              end
                end
```

Алгоритм Петерсона/Долева–Клейва–Роде

Полученный признак сохраняется в памяти (в качестве значения переменной acn_p), и если данный признак переживет этот тур, то он станет текущим отличительным признаком процесса p в следующем туре.

Чтобы оценить, переживет ли отличительный признак acn_p данный тур, его нужно сравнить как с ci_p , так и с *активным* признаком, полученным в сообщении типа $\langle \mathbf{two} \rangle$.

Процесс p отправляет сообщение $\langle \mathbf{two}, acp_p \rangle$, чтобы предоставить такую возможность следующему по порядку активному процессу.

Алгоритм Петерсона/Долева–Клейва–Роде

```
var ci_p    : P   init p ;        (* Текущий признак процесса p *)
    acn_p   : P   init udef ;     (* Признак активного соседа против ход
    win_p   : P   init udef ;     (* Признак победителя *)
    state_p : (active, passive, leader, lost) init active ;
begin if p is initiator then state_p := active else state_p := passive ;
      while win_p = udef do
        begin if state_p = active then
                begin send ⟨one, ci_p⟩; receive ⟨one, q⟩ ; acn_p := q;
                      if acn_p = ci_p then (* acn_p минимальный *)
                        begin send ⟨small, acn_p⟩ ; win_p := acn_p ;
                              receive ⟨small, q⟩
                        end
                      else  (* acn_p — текущий признак соседа *)
                        begin  send ⟨two, acn_p⟩; receive ⟨two, q⟩ ;
                               if  acn_p < ci_p and acn_p < q
                                 then  ci_p := acn_p else state_p := passive
                               end
                end
```

Алгоритм Петерсона/Долева–Клейва–Роде

Исключительным является тот случай, когда выполняется равенство $acn_p = ci_p$; тогда данный отличительный признак остается единственным активным признаком, и сообщение $\langle \textbf{small}, acn_p \rangle$ оповещает об этом все процессы.

Алгоритм Петерсона/Долева–Клейва–Роде

```
var ci_p    : P    init p ;        (* Текущий признак процесса p *)
    acn_p   : P    init udef ;     (* Признак активного соседа против ход
    win_p   : P    init udef ;     (* Признак победителя *)
    state_p : (active, passive, leader, lost) init active ;
begin if p is initiator then state_p := active else state_p := passive ;
      while win_p = udef do
        begin if state_p = active then
                begin send ⟨one, ci_p⟩; receive ⟨one, q⟩ ; acn_p := q;
                      if  acn_p = ci_p then  (* acn_p минимальный *)
                        begin  send ⟨small, acn_p⟩ ; win_p := acn_p ;
                               receive ⟨small, q⟩
                        end
                      else (* acn_p — текущий признак соседа *)
                        begin send ⟨two, acn_p⟩; receive ⟨two, q⟩ ;
                              if acn_p < ci_p and acn_p < q
                                then ci_p := acn_p else state_p := passive
                              end
                end
```

Алгоритм Петерсона/Долева–Клейва–Роде

Теорема 8.5.

Алгоритм Петерсона/Долева–Клейва–Роде решает задачу о выборах для однонаправленных колец с использованием $O(N \log N)$ обменов сообщениями.

Алгоритм Петерсона/Долева–Клейва–Роде

Теорема 8.5.

Алгоритм Петерсона/Долева–Клейва–Роде решает задачу о выборах для однонаправленных колец с использованием $O(N \log N)$ обменов сообщениями.

Доказательство.

Будем говорить, что процесс участвует в i -ом туре, если основной цикл этого процесса выполняется i -й раз. Туры не синхронизированы глобально; возможна ситуация, при которой один из процессов опережает на несколько туров другой процесс, расположенный в иной части кольца. Но коль скоро каждый процесс отправляет и получает в каждом туре ровно два сообщения, и в каналах поддерживается очередность посланий, получатель всякого сообщения будет участвовать в том же туре, в котором находился его отправитель. В первом туре все инициаторы активны , и каждый активный процесс хранит индивидуальный «текущий отличительный признак».

Алгоритм Петерсона/Долева–Клейва–Роде

Утверждение 1.

Если в начале i -го тура имеется $k > 1$ активных процессов, и все «текущие отличительные признаки» ci , хранимые этими процессами, попарно различны, то в следующий тур перейдет не менее одного и не более $k/2$ процессов. По окончании i -го тура текущие отличительные признаки активных процессов будут вновь попарно различны, и среди них будет находиться самый младший признак.

Алгоритм Петерсона/Долева–Клейва–Роде

Доказательство утверждения 1.

После обмена сообщениями вида $\langle \mathbf{one}, q \rangle$ каждый активный процесс получит текущий отличительный признак своего ближайшего активного соседа, расположенного против хода часовой стрелки. Этот признак будет не совпадать с его собственным отличительным признаком.

Алгоритм Петерсона/Долева–Клейва–Роде

Доказательство утверждения 1.

После обмена сообщениями вида $\langle \mathbf{one}, q \rangle$ каждый активный процесс получит текущий отличительный признак своего ближайшего активного соседа, расположенного против хода часовой стрелки. Этот признак будет не совпадать с его собственным отличительным признаком.

Следовательно, каждый активный процесс продолжит этот тур, и произойдет обмен сообщениями вида $\langle \mathbf{two}, q \rangle$. Поэтому каждый активный процесс получит также текущий отличительный признак второго по близости активного соседа, расположенного против хода часовой стрелки.

Алгоритм Петерсона/Долева–Клейва–Роде

Доказательство утверждения 1.

После обмена сообщениями вида $\langle \mathbf{one}, q \rangle$ каждый активный процесс получит текущий отличительный признак своего ближайшего активного соседа, расположенного против хода часовой стрелки. Этот признак будет не совпадать с его собственным отличительным признаком.

Следовательно, каждый активный процесс продолжит этот тур, и произойдет обмен сообщениями вида $\langle \mathbf{two}, q \rangle$. Поэтому каждый активный процесс получит также текущий отличительный признак второго по близости активного соседа, расположенного против хода часовой стрелки.

Все активные процессы теперь хранят разные значения *acn* , а это означает, что процессы, выдержавшие этот тур, будут в конце его хранить разные отличительные признаки.

Алгоритм Петерсона/Долева–Клейва–Роде

Доказательство утверждения 1.

Этот тур выдержит, по крайней мере, тот отличительный признак, который был наименьшим в начале тура, и значит, хотя бы один процесс перейдет в следующий тур.

Алгоритм Петерсона/Долева–Клейва–Роде

Доказательство утверждения 1.

Этот тур выдержит, по крайней мере, тот отличительный признак, который был наименьшим в начале тура, и значит, хотя бы один процесс перейдет в следующий тур.

А поскольку отличительный признак, который расположен вслед за локальным минимумом, не будет локальным минимумом, число процессов, перешедших в следующий тур, не будет превосходить $k/2$. □

Из утверждения 1 следует, что наступит и такой тур (его номер не будет превосходить $\leq \lfloor \log N \rfloor + 1)$, в начале которого будет ровно один активный участник, а именно самый младший среди отличительных признаков всех инициаторов.

Алгоритм Петерсона/Долева–Клейва–Роде

Утверждение 2.

Если в начале тура есть только один активный процесс p с текущим отличительным признаком ci_p , то по окончании этого тура алгоритм завершит свою работу, и при этом для каждого процесса q будет выполнено равенство $win_q = ci_p$.

Алгоритм Петерсона/Долева–Клейва–Роде

Доказательство утверждения 2.

Сообщение $\langle \mathbf{one}, ci_p \rangle$, отправленное процессом p , будет ретранслировано всеми процессами и в конце концов будет получено самим процессом p . Процесс p убедиться в том, что выполняется равенство $acn_p = ci_p$, и отправит сообщение $\langle \mathbf{small}, acn_p \rangle$ по кольцу, вынудив тем самым каждый процесс q выйти из основного цикла, присвоив при этом переменной win_q значение acn_p . □

Алгоритм Петерсона/Долева–Клейва–Роде

Доказательство теоремы.

Данный алгоритм завершает свой выполнение в каждом из процессов, и при этом все процессы будут согласованно оповещены об отличительном признаке лидера (этот признак является значением переменной win_q) . Процесс с указанным признаком будет пребывать в состоянии **leader** , а все остальные процессы — в состоянии **lost**.
Потребуется самое большее $\lfloor \log N \rfloor + 1$ туров, в каждом из которых происходит обмен ровно $2N$ сообщениями; это и служит обоснованием оценки сложности $2N \log N + O(N)$ по числу обменов сообщениями. □

Алгоритм Петерсона/Долева–Клейва–Роде

Задачи.

1. Приведите начальную конфигурацию для алгоритма Петерсона/Долева–Клейва–Роде, при которой алгоритму действительно потребуется провести $\lfloor \log N \rfloor + 1$ туров.
2. Приведите также начальную конфигурацию, при которой этому алгоритму потребуется всего два тура, независимо от числа инициаторов.
3. Может ли алгоритм завершить работу за один тур?

Алгоритм Петерсона/Долева–Клейва–Роде

Теорема 8.6.

Если

- кольцо — однонаправленное,

Алгоритм Петерсона/Долева–Клейва–Роде

Теорема 8.6.

Если

- кольцо — однонаправленное,
- процессы не осведомлены о размерах кольца,

Алгоритм Петерсона/Долева–Клейва–Роде

Теорема 8.6.

Если

- кольцо — однонаправленное,
- процессы не осведомлены о размерах кольца,
- в каналах поддерживается очередность сообщений,

Алгоритм Петерсона/Долева–Клейва–Роде

Теорема 8.6.

Если

- кольцо — однонаправленное,
- процессы не осведомлены о размерах кольца,
- в каналах поддерживается очередность сообщений,
- все процессы являются инициаторами,

Алгоритм Петерсона/Долева–Клейва–Роде

Теорема 8.6.

Если

- кольцо — однонаправленное,
- процессы не осведомлены о размерах кольца,
- в каналах поддерживается очередность сообщений,
- все процессы являются инициаторами,

то сложность в среднем всякого алгоритма избрания лидера будет не меньше, чем $N \cdot \mathbf{H}_N$, где $\mathbf{H}_N = \sum_{k=1}^{N} 1/k$.

Алгоритм Петерсона/Долева–Клейва–Роде

Теорема 8.7.

Всякий алгоритм избрания лидера на основе сравнения для произвольных сетей имеет сложность (и в среднем, и в наихудшем случае) не меньшую, чем $\Omega(|E| + N \log N)$.

Алгоритм Петерсона/Долева–Клейва–Роде

Теорема 8.7.

Всякий алгоритм избрания лидера на основе сравнения для произвольных сетей имеет сложность (и в среднем, и в наихудшем случае) не меньшую, чем $\Omega(|E| + N \log N)$.

Доказательство.

Самостоятельно

Алгоритм Петерсона/Долева–Клейва–Роде

Теорема 8.7.

Всякий алгоритм избрания лидера на основе сравнения для произвольных сетей имеет сложность (и в среднем, и в наихудшем случае) не меньшую, чем $\Omega(|E| + N \log N)$.

Доказательство.

Самостоятельно

Следствие.

Всякий децентрализованный волновой алгоритм для произвольных сетей без предварительной осведомленности о соседях имеет сложность по числу обменов сообщениями, не меньшую чем $\Omega(|E| + N \log N)$.

Алгоритм Петерсона/Долева–Клейва–Роде

Теорема 8.7.

Всякий алгоритм избрания лидера на основе сравнения для произвольных сетей имеет сложность (и в среднем, и в наихудшем случае) не меньшую, чем $\Omega(|E| + N \log N)$.

Доказательство.

Самостоятельно

Следствие.

Всякий децентрализованный волновой алгоритм для произвольных сетей без предварительной осведомленности о соседях имеет сложность по числу обменов сообщениями, не меньшую чем $\Omega(|E| + N \log N)$.

Доказательство.

Самостоятельно

Эффект угасания

Алгоритм избрания лидера можно получить из произвольного волнового алгоритма при помощи конструкции, которая носит название угасание .

Каждый инициатор запускает отдельную волну. Чтобы отличаться от сообщений других волн, сообщения, движущиеся по волне, запущенной процессом p , должны быть помечены отличительным признаком процесса p .

Данный алгоритм гарантирует, что независимо от количества запущенных волн только одна волна приведет к решению, а именно, та волна, которую запустил инициатор с самым младшим отличительным признаком. Все остальные волны будут прерваны еще до того, как будет принято решение.

Эффект угасания

Алгоритм избрания лидера $Ex(A)$, соответствующий волновому алгоритму A , устроен так.

Эффект угасания

Алгоритм избрания лидера $Ex(A)$, соответствующий волновому алгоритму A , устроен так.

Всякий процесс в каждый момент времени активен по отношению не более чем к одной волне; эта волна называется его текущей активной волной , обозначается *caw* и имеет первоначальное значение *udef* . Инициаторы выборов поступают так, как будто каждый из них запускает волну и придает *caw* свой отличительный признак в качестве значения.

Эффект угасания

Как только сообщение некоторой волны, запущенной процессом q , достигает процесса p , то p проводит следующий анализ этого сообщения.

Эффект угасания

Как только сообщение некоторой волны, запущенной процессом q , достигает процесса p , то p проводит следующий анализ этого сообщения.

- Если $q > caw_p$, то сообщение просто игнорируется, что немедленно приводит к тому, что волна, запущенная процессом q , угасает.

Эффект угасания

Как только сообщение некоторой волны, запущенной процессом q , достигает процесса p , то p проводит следующий анализ этого сообщения.

- Если $q > caw_p$, то сообщение просто игнорируется, что немедленно приводит к тому, что волна, запущенная процессом q , угасает.
- Если $q = caw_p$, то указанное сообщение обрабатывается так, как это предписано волновым алгоритмом.

Эффект угасания

Как только сообщение некоторой волны, запущенной процессом q , достигает процесса p , то p проводит следующий анализ этого сообщения.

- Если $q > caw_p$, то сообщение просто игнорируется, что немедленно приводит к тому, что волна, запущенная процессом q , угасает.
- Если $q = caw_p$, то указанное сообщение обрабатывается так, как это предписано волновым алгоритмом.
- Если $q < caw_p$ или caw_p имеет значение $udef$, то p присоединяется к волне, запущенной процессом q , устанавливая в своих переменных первоначальные значения, а также выполняя присваивание $caw_p := q$.

Эффект угасания

Как только сообщение некоторой волны, запущенной процессом q , достигает процесса p , то p проводит следующий анализ этого сообщения.

- Если $q > caw_p$, то сообщение просто игнорируется, что немедленно приводит к тому, что волна, запущенная процессом q , угасает.
- Если $q = caw_p$, то указанное сообщение обрабатывается так, как это предписано волновым алгоритмом.
- Если $q < caw_p$ или caw_p имеет значение $udef$, то p присоединяется к волне, запущенной процессом q , устанавливая в своих переменных первоначальные значения, а также выполняя присваивание $caw_p := q$.

Когда волна, запущенная процессом q , приводит к осуществлению события решения (в большинстве алгоритмов решение всегда принимает сам процесс q), процесс q будет считаться избранным.

Эффект угасания

Задачи.

1. Разработайте алгоритм избрания лидера в произвольной сети на основе эффекта угасания, примененного к волновому алгоритму эха. Оцените сложность по числу сообщений построенного Вами алгоритма. Обоснуйте его корректность.
2. Разработайте алгоритм избрания лидера в кольцевой сети сети на основе эффекта угасания, примененного к волновому алгоритму в колцах. Сравните построенный Вами алгоритм с алгоритмом Ченя–Робертса. В чем состоит сходство и отличие этих двух алгоритмов?

КОНЕЦ ЛЕКЦИИ 8.