

Распределённые алгоритмы

mk.cs.msu.ru → Лекционные курсы → Распределённые алгоритмы

Блок 26

Древесный волновой алгоритм

Лектор:

Подымов Владислав Васильевич

E-mail:

valdus@yandex.ru

Это децентрализованный волновой алгоритм, предназначенный для топологии дерева с хотя бы двумя узлами

- ▶ или для любой топологии, в которой распределённо задано такое дерево, содержащее все узлы
- ▶ т.е. **остовное дерево**

Считается, что каждый узел p знает множество всех своих соседей ($Neigh_p$)

Инициаторами являются все листья

- ▶ (каждый узел может легко проверить, лист ли он: $|Neigh_p| \stackrel{?}{=} 1$)

Общая схема древесного алгоритма (**А**):

1. Каждый лист отправляет **фишку** (единственному) соседу
2. Внутренний узел,
 - 2.1 приняв фишки от всех соседей, кроме одного, отправляет фишку оставшемуся, и
 - 2.2 приняв фишку от оставшегося соседа, принимает решение

Код узла p (и инициатора, и последователя):

► Переменная $X_p : 2^{Neigh_p}$, начальное значение $Neigh_p$

1. Пока $|X_p| > 1$:

1.1 receive $_q(\underline{\mathbf{tok}})$ для любого $q \in X$

1.2 $X_p := X_p \setminus \{q\}$;

2. Пусть $X_p = \{q\}$

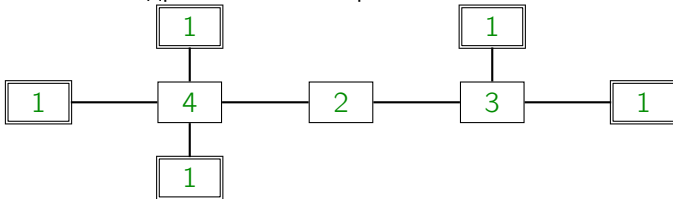
3. send $_q(\mathbf{tok})$

4. receive $_q(\underline{\mathbf{tok}})$

5. decide

Здесь считается, что последователь выполняет проверку (1) одновременно с приёмом (1.1)

Пример выполнения древесного алгоритма



Код узла p (и инициатора, и последователя):

► Переменная $X_p : 2^{Neigh_p}$, начальное значение $Neigh_p$

1. Пока $|X_p| > 1$:

1.1 receive $_q(\underline{\mathbf{tok}})$ для любого $q \in X$

1.2 $X_p := X_p \setminus \{q\}$;

2. Пусть $X_p = \{q\}$

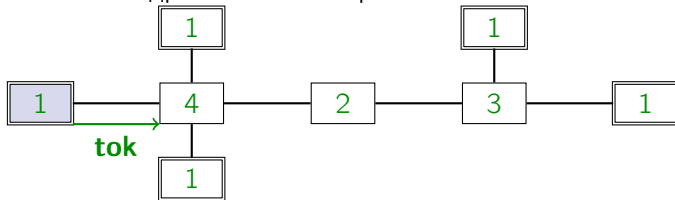
3. send $_q(\mathbf{tok})$

4. receive $_q(\underline{\mathbf{tok}})$

5. decide

Здесь считается, что последователь выполняет проверку (1) одновременно с приёмом (1.1)

Пример выполнения древесного алгоритма



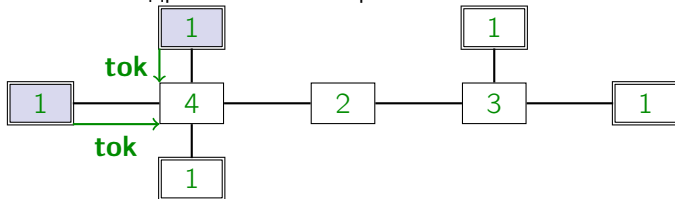
Код узла p (и инициатора, и последователя):

- ▶ Переменная $X_p : 2^{Neigh_p}$, начальное значение $Neigh_p$

1. Пока $|X_p| > 1$:
 - 1.1 $\text{receive}_q(\mathbf{tok})$ для любого $q \in X$
 - 1.2 $X_p := X_p \setminus \{q\}$;
2. Пусть $X_p = \{q\}$
3. $\text{send}_q(\mathbf{tok})$
4. $\text{receive}_q(\mathbf{tok})$
5. decide

Здесь считается, что последователь выполняет проверку (1) одновременно с приёмом (1.1)

Пример выполнения древесного алгоритма



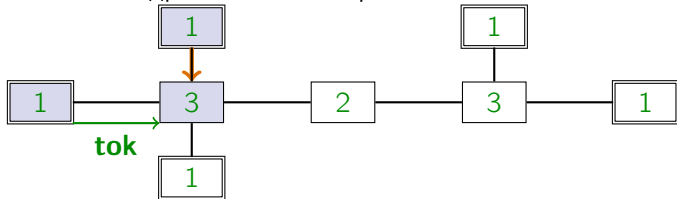
Код узла p (и инициатора, и последователя):

► Переменная $X_p : 2^{Neigh_p}$, начальное значение $Neigh_p$

1. Пока $|X_p| > 1$:
 - 1.1 receive $_q(\underline{\mathbf{tok}})$ для любого $q \in X$
 - 1.2 $X_p := X_p \setminus \{q\}$;
2. Пусть $X_p = \{q\}$
3. send $_q(\mathbf{tok})$
4. receive $_q(\underline{\mathbf{tok}})$
5. decide

Здесь считается, что последователь выполняет проверку (1) одновременно с приёмом (1.1)

Пример выполнения древесного алгоритма



Код узла p (и инициатора, и последователя):

► Переменная $X_p : 2^{Neigh_p}$, начальное значение $Neigh_p$

1. Пока $|X_p| > 1$:

1.1 receive $_q(\underline{\mathbf{tok}})$ для любого $q \in X$

1.2 $X_p := X_p \setminus \{q\}$;

2. Пусть $X_p = \{q\}$

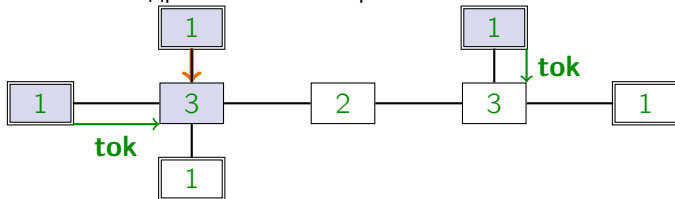
3. send $_q(\mathbf{tok})$

4. receive $_q(\underline{\mathbf{tok}})$

5. decide

Здесь считается, что последователь выполняет проверку (1) одновременно с приёмом (1.1)

Пример выполнения древесного алгоритма



Код узла p (и инициатора, и последователя):

► Переменная $X_p : 2^{Neigh_p}$, начальное значение $Neigh_p$

1. Пока $|X_p| > 1$:

1.1 receive $_q(\underline{\mathbf{tok}})$ для любого $q \in X$

1.2 $X_p := X_p \setminus \{q\}$;

2. Пусть $X_p = \{q\}$

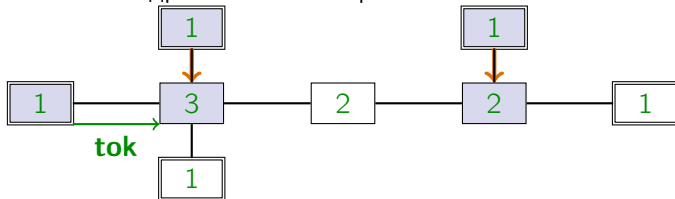
3. send $_q(\mathbf{tok})$

4. receive $_q(\underline{\mathbf{tok}})$

5. decide

Здесь считается, что последователь выполняет проверку (1) одновременно с приёмом (1.1)

Пример выполнения древесного алгоритма



Код узла p (и инициатора, и последователя):

► Переменная $X_p : 2^{Neigh_p}$, начальное значение $Neigh_p$

1. Пока $|X_p| > 1$:

1.1 receive $_q(\underline{\mathbf{tok}})$ для любого $q \in X$

1.2 $X_p := X_p \setminus \{q\}$;

2. Пусть $X_p = \{q\}$

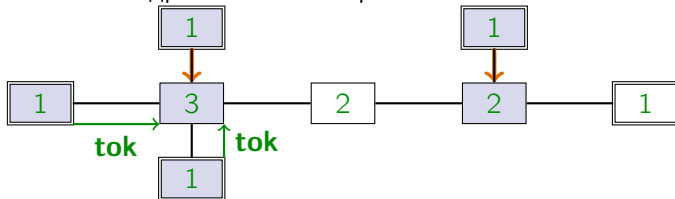
3. send $_q(\mathbf{tok})$

4. receive $_q(\underline{\mathbf{tok}})$

5. decide

Здесь считается, что последователь выполняет проверку (1) одновременно с приёмом (1.1)

Пример выполнения древесного алгоритма



Код узла p (и инициатора, и последователя):

► Переменная $X_p : 2^{Neigh_p}$, начальное значение $Neigh_p$

1. Пока $|X_p| > 1$:

1.1 receive $_q(\underline{\mathbf{tok}})$ для любого $q \in X$

1.2 $X_p := X_p \setminus \{q\}$;

2. Пусть $X_p = \{q\}$

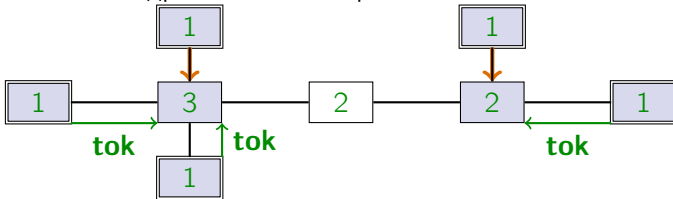
3. send $_q(\mathbf{tok})$

4. receive $_q(\underline{\mathbf{tok}})$

5. decide

Здесь считается, что последователь выполняет проверку (1) одновременно с приёмом (1.1)

Пример выполнения древесного алгоритма



Код узла p (и инициатора, и последователя):

► Переменная $X_p : 2^{Neigh_p}$, начальное значение $Neigh_p$

1. Пока $|X_p| > 1$:

1.1 receive $_q(\underline{\mathbf{tok}})$ для любого $q \in X$

1.2 $X_p := X_p \setminus \{q\}$;

2. Пусть $X_p = \{q\}$

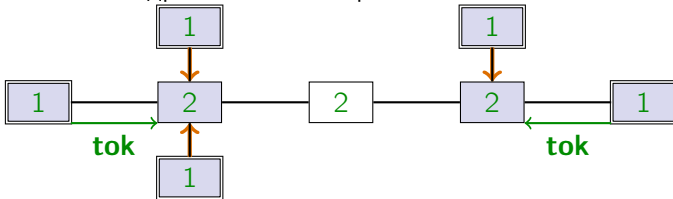
3. send $_q(\mathbf{tok})$

4. receive $_q(\underline{\mathbf{tok}})$

5. decide

Здесь считается, что последователь выполняет проверку (1) одновременно с приёмом (1.1)

Пример выполнения древесного алгоритма



Код узла p (и инициатора, и последователя):

► Переменная $X_p : 2^{Neigh_p}$, начальное значение $Neigh_p$

1. Пока $|X_p| > 1$:

1.1 receive $_q(\underline{\mathbf{tok}})$ для любого $q \in X$

1.2 $X_p := X_p \setminus \{q\}$;

2. Пусть $X_p = \{q\}$

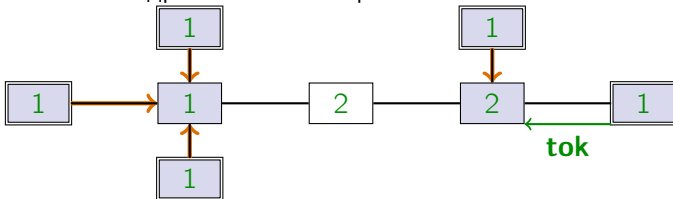
3. send $_q(\mathbf{tok})$

4. receive $_q(\underline{\mathbf{tok}})$

5. decide

Здесь считается, что последователь выполняет проверку (1) одновременно с приёмом (1.1)

Пример выполнения древесного алгоритма



Код узла p (и инициатора, и последователя):

► Переменная $X_p : 2^{Neigh_p}$, начальное значение $Neigh_p$

1. Пока $|X_p| > 1$:

1.1 receive $_q(\underline{\mathbf{tok}})$ для любого $q \in X$

1.2 $X_p := X_p \setminus \{q\}$;

2. Пусть $X_p = \{q\}$

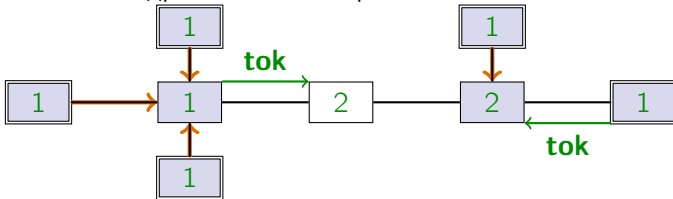
3. send $_q(\mathbf{tok})$

4. receive $_q(\underline{\mathbf{tok}})$

5. decide

Здесь считается, что последователь выполняет проверку (1) одновременно с приёмом (1.1)

Пример выполнения древесного алгоритма



Код узла p (и инициатора, и последователя):

► Переменная $X_p : 2^{Neigh_p}$, начальное значение $Neigh_p$

1. Пока $|X_p| > 1$:

1.1 receive $_q(\underline{\mathbf{tok}})$ для любого $q \in X$

1.2 $X_p := X_p \setminus \{q\}$;

2. Пусть $X_p = \{q\}$

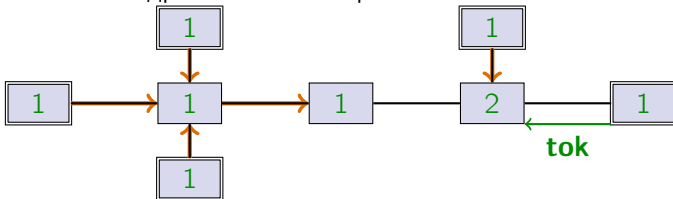
3. send $_q(\mathbf{tok})$

4. receive $_q(\underline{\mathbf{tok}})$

5. decide

Здесь считается, что последователь выполняет проверку (1) одновременно с приёмом (1.1)

Пример выполнения древесного алгоритма



Код узла p (и инициатора, и последователя):

► Переменная $X_p : 2^{Neigh_p}$, начальное значение $Neigh_p$

1. Пока $|X_p| > 1$:

1.1 receive $_q(\underline{\mathbf{tok}})$ для любого $q \in X$

1.2 $X_p := X_p \setminus \{q\}$;

2. Пусть $X_p = \{q\}$

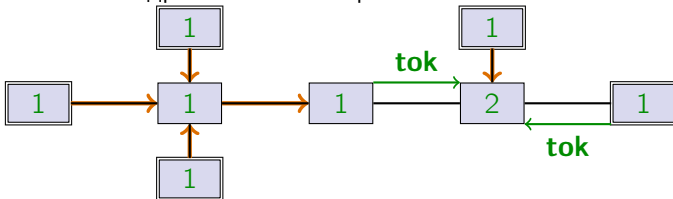
3. send $_q(\mathbf{tok})$

4. receive $_q(\underline{\mathbf{tok}})$

5. decide

Здесь считается, что последователь выполняет проверку (1) одновременно с приёмом (1.1)

Пример выполнения древесного алгоритма



Код узла p (и инициатора, и последователя):

► Переменная $X_p : 2^{Neigh_p}$, начальное значение $Neigh_p$

1. Пока $|X_p| > 1$:

1.1 receive $_q(\underline{\mathbf{tok}})$ для любого $q \in X$

1.2 $X_p := X_p \setminus \{q\}$;

2. Пусть $X_p = \{q\}$

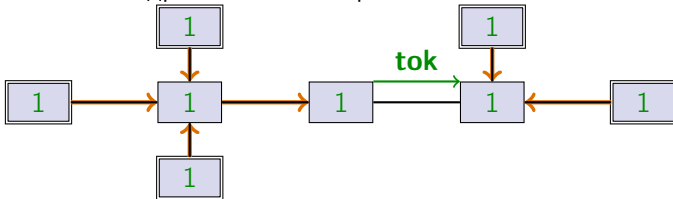
3. send $_q(\mathbf{tok})$

4. receive $_q(\underline{\mathbf{tok}})$

5. decide

Здесь считается, что последователь выполняет проверку (1) одновременно с приёмом (1.1)

Пример выполнения древесного алгоритма



Код узла p (и инициатора, и последователя):

► Переменная $X_p : 2^{Neigh_p}$, начальное значение $Neigh_p$

1. Пока $|X_p| > 1$:

1.1 receive $_q(\underline{\mathbf{tok}})$ для любого $q \in X$

1.2 $X_p := X_p \setminus \{q\}$;

2. Пусть $X_p = \{q\}$

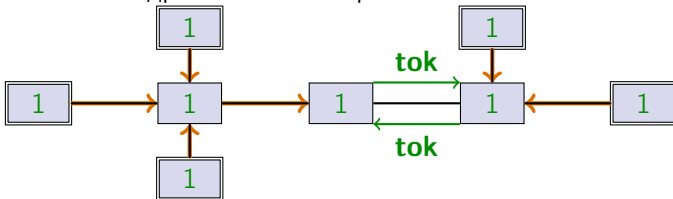
3. send $_q(\mathbf{tok})$

4. receive $_q(\underline{\mathbf{tok}})$

5. decide

Здесь считается, что последователь выполняет проверку (1) одновременно с приёмом (1.1)

Пример выполнения древесного алгоритма



Код узла p (и инициатора, и последователя):

► Переменная $X_p : 2^{Neigh_p}$, начальное значение $Neigh_p$

1. Пока $|X_p| > 1$:

1.1 receive $_q(\underline{\mathbf{tok}})$ для любого $q \in X$

1.2 $X_p := X_p \setminus \{q\}$;

2. Пусть $X_p = \{q\}$

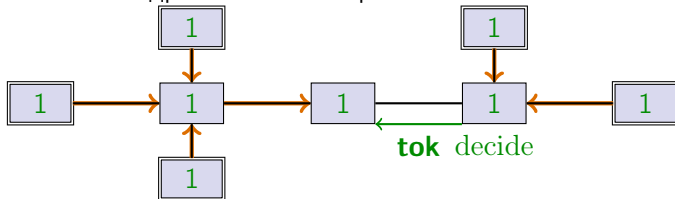
3. send $_q(\mathbf{tok})$

4. receive $_q(\underline{\mathbf{tok}})$

5. decide

Здесь считается, что последователь выполняет проверку (1) одновременно с приёмом (1.1)

Пример выполнения древесного алгоритма



Код узла p (и инициатора, и последователя):

► Переменная $X_p : 2^{Neigh_p}$, начальное значение $Neigh_p$

1. Пока $|X_p| > 1$:

1.1 receive $_q(\underline{\mathbf{tok}})$ для любого $q \in X$

1.2 $X_p := X_p \setminus \{q\}$;

2. Пусть $X_p = \{q\}$

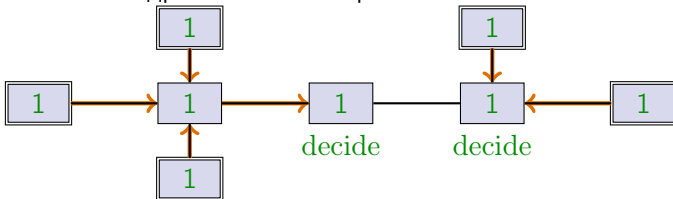
3. send $_q(\mathbf{tok})$

4. receive $_q(\underline{\mathbf{tok}})$

5. decide

Здесь считается, что последователь выполняет проверку (1) одновременно с приёмом (1.1)

Пример выполнения древесного алгоритма



Утверждение (уникальность отправителя). В любом вычислении алгоритма $\mathcal{A}$ каждый узел отправляет не более одной фишки

Доказательство. Согласно коду, фишка отправляется только в (3) и только один раз ▼

1. Пока $|X_p| > 1$:
 - 1.1 $\text{receive}_q(\mathbf{tok})$ для любого $q \in X$
 - 1.2 $X_p := X_p \setminus \{q\}$;
2. Пусть $X_p = \{q\}$
3. $\text{send}_q(\mathbf{tok})$
4. $\text{receive}_q(\mathbf{tok})$
5. decide

Далее будем полагать, что:

- ▶ Действие (1) - это проверка условия $|X_p| > 1$
- ▶ (1.1) и (1.2) - это одно действие (1.X)
- ▶ В действии (4) дополнительно выполняется присваивание $X_p := X_p \setminus \{q\}$;
 - ▶ Это присваивание влияет только на вид состояния узла p после выполнения (4) и не влияет на «глобальные» свойства алгоритма

Утверждение (смысл X_p). В любой достижимой конфигурации любой р.с. алгоритма $\mathcal{A}$ значение X_p — это множество всех соседей, от которых p не принял ни одной фишки

Доказательство.

Пусть $\mathcal{N}_p$ — множество всех соседей, от которых p не принял ни одной фишки

В начальной конфигурации $X_p = Neigh_p = \mathcal{N}_p$

Действия (1.X) и (4) удаляют заданный узел q одновременно

- ▶ из $\mathcal{N}_p$ (принимается фишка от q)
- ▶ и из X_p (это присваивание прописано явно)

Остальные действия не изменяют ни $\mathcal{N}_p$, ни X_p ▼

Утверждение (завершаемость). Все вычисления любой р.с. алгоритма $\mathcal{A}$ конечны

Доказательство.

(2)–(5) — это очевидно конечный набор действий

Значит, достаточно показать, что (1) и (1.X) для любого узла p выполняются конечное число раз

(1) выполняется на 1 раз больше, чем (1.X)

Если в вычислении отправляется суммарно $N < (|Neigh_p| - 1)$ фишек, адресованных p , то (1.X) выполняется N раз

Иначе значение $|X_p|$

- ▶ после каждого выполнения (1.X) строго уменьшается (согласно **уникальности отправителя** и **смыслу X_p**) и
- ▶ не может стать меньше 1 (согласно (1), (1.2) и выбору топологии), и остаётся только применить **теорему о проверке живости** для функции нормировки $|X_p|$ и целевого множества всех тупиковых конфигураций ▼

Пусть дерево топологии имеет вид $T = (V, E)$

Утверждение (суммарный размер X_p). В любой достижимой конфигурации любой р.с. алгоритма $\mathcal{A}$ верно равенство $\sum_{p \in V} |X_p| = 2|E| - K$, где K — суммарное число принятых фишек

Доказательство.

Согласно смыслу X_p и уникальности отправителя, если K_p — число фишек, принятых узлом p , то $|Neigh_p| = |X_p| + K_p$

Если просуммировать это равенство по всем $p \in V$ и применить известный факт о том, что сумма степеней вершин графа есть дважды число рёбер, то получится равенство $2|E| = \sum_{p \in V} |X_p| + \sum_{p \in V} K_p$

При этом $\sum_{p \in V} K_p = K$, откуда следует равенство из условия ▼

Утверждение (чистота каналов). В любой достижимой заключительной конфигурации любой р.с. алгоритма $\mathcal{A}$ в коммуникационной подсистеме не содержится ни одной фишки

Доказательство.

Согласно **уникальности отправителя**, каждый узел отправляет не более одной фишки в каждый канал

Согласно (1.1) и (4), если каждый из соседей узла p отправляет ему не более одной фишки, то p принимает все эти фишки ▼

Утверждение (принятие решения). В любом вычислении любой р.с. алгоритма A хотя бы раз принимается решение

Доказательство. Предположим от противного, что это не так: есть топология и вычисление для неё, такие что ни один узел в этом вычислении не принимает решение

Согласно **завершаемости**, рассматриваемое вычисление конечно

По устройству кода, в последней (тупиковой) конфигурации γ этого вычисления ни один узел не выполнил (4) — иначе он выполнил бы и (5)

Согласно **смыслу** X_p , тогда в γ для каждого узла p верно $|X_p| \geq 1$

По **суммарному размеру** X_p , в γ верно равенство $\sum_{p \in V} |X_p| = 2|E| - K$

Согласно **чистоте каналов**, K — это суммарное число не только принятых, но и отправленных фишек

Согласно **уникальности отправителя**, тогда $(|V| - K)$ узлов в γ не отправили ни одной фишки

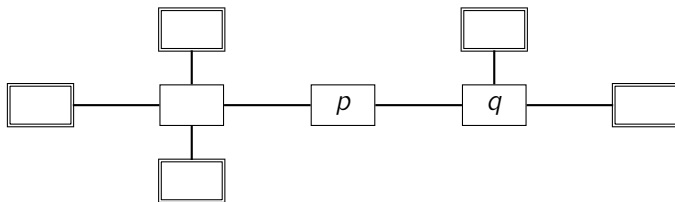
По устройству (1) и (3), для этих узлов q в γ верно $|X_q| \geq 2$

Значит, $2|E| - K = \sum_{p \in V} |X_p| \geq |V| + (|V| - K) = 2|V| - K$

Следовательно, $|E| \geq |V|$, что *противоречит* определению дерева ▼

Зададим дерево $T_{pq} = (V_{pq}, E_{pq})$ как компоненту связности узла p после удаления ребра (p, q) из дерева топологии

Например:



T_{pq} — это поддерево, состоящее из пяти левых вершин и соединяющих их рёбер

T_{qp} — это поддерево, состоящее из трёх правых вершин и соединяющих их рёбер

Утверждение (причинность в T_{pq}). Для любого события α отправки фишки узлом p узлу q в любом вычислении любой р.с. алгоритма $\mathcal{A}$ и для любого узла r дерева T_{pq} в последовательности событий этого вычисления содержится событие β узла r , такое что $\beta \preceq \alpha$

Доказательство (индукцией).

База: если p — лист, то $V_{pq} = \{p\}$ и можно взять $\beta = \alpha$

Индуктивный переход:

Если p не лист, то по устройству кода и смыслу X_p он отправляет фишку $p \rightarrow q$ (3) только после приёма фишек от каждого своего соседа r , кроме q

По предположению индукции, для отправки (3) фишки соседом r , связанной с этим приёмом, в каждом узле из T_{rp} происходит нестрогая причина этой отправки

При этом $V_{pq} = \left(\bigcup_{r \in \text{Neigh}_p \setminus \{q\}} V_{rp} \right) \cup \{p\}$

Значит, по свойствам отношения $\preceq$, для действия (3) узла p в каждом узле из V_{pq} происходит нестрогая причина этого действия ▼

Утверждение (полнота покрытия). В любом вычислении любой р.с. алгоритма $\mathcal{A}$ любое событие `decide` является нестрогим следствием хотя бы одного события каждого узла

Доказательство.

По устройству кода, действие `decide` (5), выполненное узлом q , является следствием приёма сообщений (1.X), (4) от каждого его соседа p

По определению $\prec$, причинами этих приёмов являются отправки соседями p сообщений узлу q

По причинности в T_{pq} и свойствам $\preceq$, в каждом узле дерева T_{pq} для каждого соседа p узла q содержится причина действия (5) узла q

При этом $(\bigcup_{p \in Neigh_q} V_{pq}) \cup \{q\} = V$ ▼

Теорема. Алгоритм $\mathcal{A}$ является волновым алгоритмом

Доказательство. Ранее были доказаны свойства завершаемости, принятия решения и полноты покрытия для $\mathcal{A}$ ▼

Д.з. 1. Докажите, что в древесном волновом алгоритме решение всегда принимают ровно две вершины