

Распределённые алгоритмы

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

Блок 37

Избрание лидера и построение остовного дерева:
алгоритм Галлагера-Хамблета-Спиры (GHS)

Лектор:

Подымов Владислав Васильевич

E-mail:

valdus@yandex.ru

Выборы и построение остовного дерева

Задача построения остовного дерева в р.с. можно поставить так: требуется в каждом узле p выделить подмножество \mathcal{N}_p множества соседей ($Neigh_p$) так, чтобы для некоторого остовного дерева T для каждого узла p множество \mathcal{N}_p было множеством всех соседей p в T

Алгоритмы построения остовного дерева тесно связаны с **алгоритмами избрания лидера**

Пусть C_E и C_T — коммуникационные сложности **задачи избрания лидера** и **построения остовного дерева** соответственно для произвольной связной топологии $\Gamma = (V, E)$

Выборы и построение остовного дерева

Избрать лидера можно так:

- ▶ построить остовное дерево сети и
- ▶ избрать лидера в дереве со сложностью $O(|V|)$

Значит, в частности, $C_E \leq C_T + O(|V|)$

Построить остовное дерево можно так:

- ▶ избрать лидера и
- ▶ запустить из лидера алгоритм эха со сложностью $2|E|$

Значит, в частности, $C_T \leq C_E + 2|E|$

Известна нижняя оценка сложности избрания лидера в произвольной связной топологии (и в любом классе топологий, включающих все кольца): $\Omega(|V| \log |V| + |E|)$

Следовательно, чтобы научиться избирать лидера с наилучшей коммуникационной сложностью ($\Theta(|V| \log |V| + |E|)$), необходимо и достаточно научиться вычислять остовное дерево с этой сложностью

GHS: вступление, допущения

Алгоритм Галлагера-Хамблета-Спиры (GHS) — это описанный дальше распределённый алгоритм построения остовного дерева, предназначенный для произвольной связной топологии и имеющий коммуникационную сложность $O(|V| \log |V| + |E|)$ (в худшем случае)

В GHS используются следующие допущения и соглашения:

1. $\Gamma = (V, E)$ — граф топологии
2. Каждому каналу e из E приписан вес $\omega(e)$ — число (\mathbb{N}_0 , \mathbb{Z} или \mathbb{R})
Разным каналам приписаны разные веса
3. Каждый узел p знает множество всех своих соседей ($Neigh_p$) и веса инцидентных ему каналов ($\omega_p[q] = \omega(p - q)$)
4. Все узлы являются **инициаторами**
То есть единственное различие узлов — это знания (3),
и все узлы начинают выполнять действия,
не дожидаясь приёма сообщения

Минимальные остовные деревья

Весом графа назовём сумму весов рёбер этого графа

Минимальное остовное дерево (MST) — это **остовное дерево**, имеющее наименьший вес среди всех остовных деревьев

Утверждение 1. Если все веса рёбер графа попарно различны, то существует единственное MST этого графа

Доказательство. Предположим от противного, что существуют два различных MST: $T_1 = (V, E_1)$ и $T_2 = (V, E_2)$

Рассмотрим ребро e наименьшего веса, содержащееся в одном из этих деревьев и не содержащееся в другом

Без ограничения общности положим, что $e \in E_1$ и $e \notin E_2$

Тогда в графе $T'_2 = (V, E_2 \cup \{e\})$ содержится цикл C

Так как в T_1 нет циклов, то в C содержится ребро e' , не входящее в T_1

По выбору ребра e , верно $w(e) < w(e')$

Значит, $(V, (E_2 \setminus \{e'\}) \cup \{e\})$ — дерево, имеющее меньший вес, чем T_2
(*противоречие*) ▼

Минимальные остовные деревья

Основываясь на этом утверждении, дальше будем писать просто «MST», имея в виду единственное MST графа Γ

Фрагментами будем называть поддеревья MST

Ребро e относительно фрагмента F будем называть

- ▶ **внутренним**, если обе его вершины принадлежат F ,
- ▶ **внешним**, если обе не принадлежат, и
- ▶ **граничным**, если одна принадлежит и одна не принадлежит

Минимальным граничным ребром фрагмента будем называть граничное ребро, имеющее наименьший вес среди всех граничных рёбер

Утверждение 2 (Д.з. 1). Если $F = (V_F, E_F)$ — фрагмент и $e = (v, w)$ — **минимальное граничное ребро** фрагмента F , то $(V_F \cup \{v, w\}, E_F \cup \{e\})$ — **фрагмент**

GHS: общее описание

На каждом этапе выполнения алгоритма каждый узел принадлежит некоторому фрагменту, вычисленному алгоритмом (текущему фрагменту этого узла)

В начале выполнения узел p принадлежит фрагменту $(\{p\}, \emptyset)$

Если фрагменты $F_1 = (V_1, E_1)$ и $F_2 = (V_2, E_2)$ не пересекаются и e — граничное ребро для них обоих и минимальное граничное ребро для одного из них, то при выполнении этого алгоритма F_1 и F_2 могут быть объединены по ребру e в дерево $(V_1 \cup V_2, E_1 \cup E_2 \cup \{e\})$, которое по утверждению 2 также является фрагментом

GHS: общее описание

Каждому фрагменту присваивается особое число — **ранг**

Фрагмент большего ранга (по сравнению с заданным) будем называть **старшим**, меньшего — **младшим**, равного — **ровесником**

Фрагмент вида $(\{p\}, \emptyset)$ имеет ранг 0

Ранг объединения фрагментов F_1 и F_2 с рангами r_1 и r_2 равен

- ▶ рангу старшего фрагмента, если $r_1 \neq r_2$
- ▶ $r_1 + 1$, если $r_1 = r_2$

Каждому фрагменту с хотя бы одним ребром присваивается **имя**

Имя объединения ровесников — это вес соединяющего их ребра

Объединению фрагментов разных рангов присваивается имя старшего фрагмента

GHS: общее описание

Ребро, вес которого является именем фрагмента, будем называть **стержнем**, и узлы, инцидентные стержню, будем называть **стержневыми**

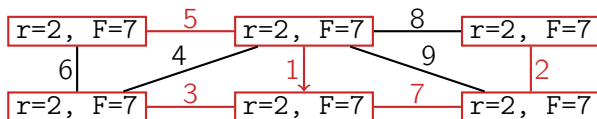
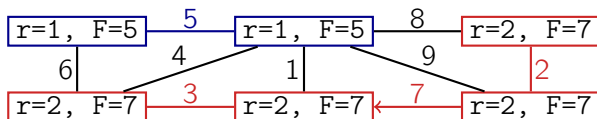
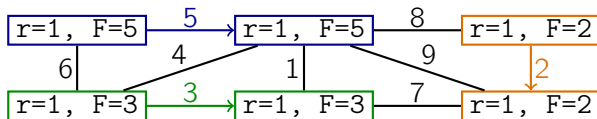
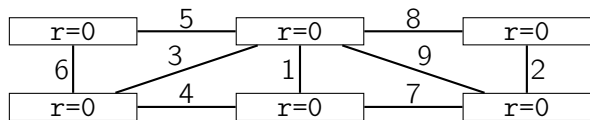
После объединения фрагментов ранг и имя рассылаются всем узлам фрагмента в направлении от стержня

Так как веса всех рёбер попарно различны, то имена различных фрагментов различны и проверка граничности ребра равносильна проверке неравенства имён узлов этого ребра

Алгоритм GHS строит MST последовательным объединением фрагментов, начиная с фрагментов ранга 0 и оканчивая фрагментом, представляющим собой всё MST

GHS: общее описание

Пример построения MST из фрагментов (r — ранг, F — имя):



GHS: простейшие свойства

Утверждение 3. В любом фрагменте любого ранга $r \in \mathbb{N}_0$ содержится не менее 2^r узлов

Доказательство (индукцией по способам объединения фрагментов).

В каждом фрагменте ранга 0 содержится ровно $2^0 = 1$ узел

Если объединяются фрагменты разных рангов, старший F имеет ранг r и содержит не менее 2^r узлов, то ранг объединения равен r , и в объединении содержатся по крайней мере все узлы F , и их не менее 2^r

Если объединяются фрагменты одного ранга r и каждый содержит не менее 2^r узлов, то объединение имеет ранг $(r + 1)$ и содержит не менее $(2^r + 2^r) = 2^{r+1}$ узлов ▼

GHS: простейшие свойства

Утверждение 4. Для описанного способа построения MST значение ранга в каждом узле изменяется не более $\log_2 |V|$ раз

Доказательство.

По утверждению 3, значение ранга фрагмента не может быть больше $\log_2 |V|$

При изменении ранга в узле он увеличивается хотя бы на 1

Значит, изменение может произойти не более $\log_2 |V|$ раз ▼

Следствие. Для описанного способа построения MST суммарное число изменений рангов в узлах сети не превосходит $|V| \log_2 |V|$

GHS: правила присоединения фрагмента

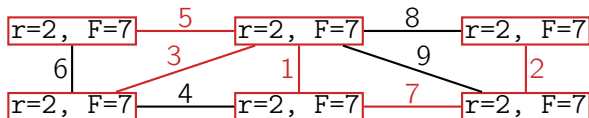
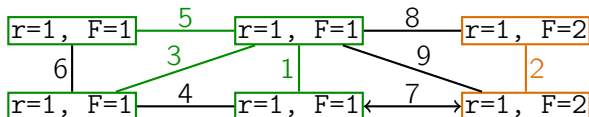
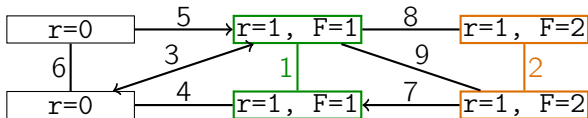
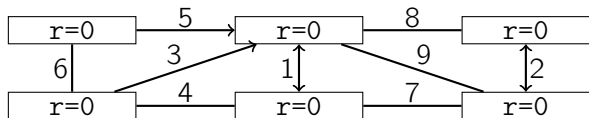
Объединение фрагментов в GHS носит «полуодносторонний» характер: либо младший фрагмент присоединяется к старшему, либо ровесники присоединяются друг к другу

Более точно, **присоединение** фрагмента F к фрагменту G по минимальному граничному ребру e_F для F в GHS будет следовать трём правилам:

1. Если G — старший, то узлы фрагмента F присоединяются к фрагменту G : изменяют свои имя и ранг на имя и ранг G
2. Если F и G — ровесники и e_F — минимальное граничное ребро для G , то узлы F и G присоединяются друг к другу, увеличивая ранг и приобретая имя стержня e_F
3. В остальных случаях F ожидает, когда станет применимым одно из правил (1), (2)

GHS: правила присоединения фрагмента

Пример объединения фрагментов в GHS по правилам A, B



GHS: переменные узла p

- ▶ $name_p : \Omega^\perp = \perp$ — имя текущего фрагмента (Ω — тип весов)
- ▶ $rank_p : \mathbb{N}_0 = 0$ — ранг текущего фрагмента
- ▶ $parent_p : Neigh_p^\perp = \perp$ — сосед в направлении стержня фрагмента (родитель)
- ▶ $cfrag_p[q] : \{\mathfrak{t}, \mathfrak{f}\}^\perp = \perp$ для каждого $q \in Neigh_p$ — тип ребра $p - q$
 - ▶ \mathfrak{t} означает, что ребро входит в MST
 - ▶ \mathfrak{f} означает, что ребро не входит в MST
 - ▶ \perp означает, что принадлежность ребра MST неясна
- ▶ $state_p : \{search, found\} = found$ — режим работы узла
 - ▶ $search$ означает поиск минимального граничного ребра
 - ▶ $found$ означает, что минимальное граничное ребро найдено
- ▶ $\omega best_p^\perp : \Omega \cup \{\perp\} = \perp$ — вес наилучшего известного узлу граничного ребра
- ▶ $ctest_p : Neigh_p^\perp = \perp, cbest_p : Neigh_p^\perp = \perp$ — вероятные направления к минимальному граничному ребру
- ▶ $cN_p : \mathbb{N}_0 = 0$ — счётчик исследованных рёбер

GHS: виды сообщений

- ▶ (**connect**, *rank*) — отправляется в минимальное граничное ребро для обозначения готовности фрагмента присоединиться
- ▶ (**initiate**, *rank*, *name*, *state*) — принимается по минимальному граничному ребру для присоединения и пересылается узлам фрагмента
- ▶ (**test**, *rank*, *name*) — отправляется в инцидентное ребро типа \perp наименьшего веса в рамках поиска минимального граничного ребра
- ▶ **reject** — отправляется в ответ на **test** во внутреннее ребро
- ▶ **accept** — отправляется в ответ на **test** в граничное ребро
- ▶ (**report**, *w_{best}*) — отправляется по направлению к стержню с наименьшим обнаруженным весом граничного ребра
- ▶ **changeroot** — принимается от соседа, расположенного ближе к стержню (родителя)

GHS: инициализация узла p

Первое однократно выполняющееся действие каждого узла p :

1. $q = \underset{q \in Neigh_p}{\text{Argmin}} \omega_p[q]$
 2. $cfrag_p[q] := \mathfrak{t}$;
 3. $\text{send}_q(\mathbf{connect}, 0)$
-

Узел в тривиальном фрагменте ранга 0 очевидным образом находит минимальное граничное ребро и объявляет о готовности присоединиться по этому ребру

После этого каждый узел p в бесконечном цикле

- ▶ ожидает выполнения хотя бы одного из предусловий, описанных далее,
- ▶ произвольно выбирает одно из выполненных предусловий и
- ▶ выполняет процедуру, отвечающую выбранному предусловию

GHS: приём connect узлом p

Предусловие:

в канале хотя бы от одного соседа q есть сообщение (**connect**, $rank$), где (а) $rank \leq rank_p$ и (б) если $rank = rank_p$, то $cfrag_p[q] \neq \perp$ (т.е. можно применить правило (1) или (2) объединения фрагментов)

1. receive $_q$ (**connect**, $rank$) для сообщения и q из условия
2. Если $rank < rank_p$ (правило 1):
 - 2.1 $cfrag_p[q] := \text{t}$;
 - 2.2 send $_q$ (**initiate**, $rank_p$, $name_p$, $state_p$)
3. Если $rank = rank_p$ (правило 2):
send $_q$ (**initiate**, $rank_p + 1$, $\omega_p[q]$, $search$)

Узлу q отправляется приказ присоединиться, если он

- ▶ младший или
- ▶ ровесник и при этом узел p уверен, что (p, q) — минимальное граничное ребро для его текущего фрагмента

Вопрос. Что происходит с сообщением типа **connect** в вычислении в целом, если сейчас для него условие не выполнено?

GHS: приём **initiate** узлом p

Предусловие:

в канале хотя бы от одного соседа q есть сообщение типа **initiate**

1. receive $_q$ (**initiate**, $rank$, $name$, $state$) для произвольного $q \in Neigh_p$
2. $(rank_p, name_p, state_p, parent_p) := (rank, name, state, q)$;
3. $(wbest_p, cbest_p) := (\infty, \perp)$;
4. Для всех $r \in Neigh_p \setminus \{q\}$, таких что $cfrag_p[r] = \uparrow$:
send $_r$ (**initiate**, $rank$, $name$, $state$)
5. Если $state_p = search$:
 - 5.1 $cN_p := 0$;
 - 5.2 $TEST_p$

После получения приказа о присоединении узел

- ▶ обновляет информацию о фрагменте (ранг, имя, статус поиска минимального граничного ребра, родитель),

GHS: приём **initiate** узлом p

Предусловие:

в канале хотя бы от одного соседа q есть сообщение типа **initiate**

1. receive $_q$ (**initiate**, $rank$, $name$, $state$) для произвольного $q \in Neigh_p$
2. $(rank_p, name_p, state_p, parent_p) := (rank, name, state, q)$;
3. $(wbest_p, cbest_p) := (\infty, \perp)$;
4. Для всех $r \in Neigh_p \setminus \{q\}$, таких что $cfrag_p[r] = \dagger$:
send $_r$ (**initiate**, $rank$, $name$, $state$)
5. Если $state_p = search$:
 - 5.1 $cN_p := 0$;
 - 5.2 $TEST_p$

После получения приказа о присоединении узел

- ▶ «сбрасывает» информацию о наилучшем известном граничном ребре,

GHS: приём **initiate** узлом p

Предусловие:

в канале хотя бы от одного соседа q есть сообщение типа **initiate**

1. receive $_q$ (**initiate**, $rank$, $name$, $state$) для произвольного $q \in Neigh_p$
2. $(rank_p, name_p, state_p, parent_p) := (rank, name, state, q)$;
3. $(wbest_p, cbest_p) := (\infty, \perp)$;
4. Для всех $r \in Neigh_p \setminus \{q\}$, таких что $cfrag_p[r] = \dagger$:
send $_r$ (**initiate**, $rank$, $name$, $state$)
5. Если $state_p = search$:
 - 5.1 $cN_p := 0$;
 - 5.2 $TEST_p$

После получения приказа о присоединении узел

- ▶ пересылает остальным соседям в «старом» фрагменте приказ о присоединении и

GHS: приём **initiate** узлом p

Предусловие:

в канале хотя бы от одного соседа q есть сообщение типа **initiate**

1. receive $_q$ (**initiate**, $rank$, $name$, $state$) для произвольного $q \in Neigh_p$
2. $(rank_p, name_p, state_p, parent_p) := (rank, name, state, q)$;
3. $(wbest_p, cbest_p) := (\infty, \perp)$;
4. Для всех $r \in Neigh_p \setminus \{q\}$, таких что $cfrag_p[r] = \dagger$:
send $_r$ (**initiate**, $rank$, $name$, $state$)
5. Если $state_p = search$:
 - 5.1 $cN_p := 0$;
 - 5.2 $TEST_p$

После получения приказа о присоединении узел

- ▶ если в приказе говорится продолжить поиск минимального граничного ребра, то присоединяется к поиску при помощи (описанной далее) процедуры $TEST_p$

GHS: приём **initiate** узлом p

Предусловие:

в канале хотя бы от одного соседа q есть сообщение типа **initiate**

1. receive $_q$ (**initiate**, $rank$, $name$, $state$) для произвольного $q \in Neigh_p$
2. $(rank_p, name_p, state_p, parent_p) := (rank, name, state, q)$;
3. $(wbest_p, cbest_p) := (\infty, \perp)$;
4. Для всех $r \in Neigh_p \setminus \{q\}$, таких что $cfrag_p[r] = \dagger$:
send $_r$ (**initiate**, $rank$, $name$, $state$)
5. Если $state_p = search$:
 - 5.1 $cN_p := 0$;
 - 5.2 $TEST_p$

Вопрос.

Зачем узел сбрасывает информацию о наилучшем граничном ребре?
Останется ли алгоритм корректным, если этого не делать?

GHS: приём *initiate* узлом p

Процедура $TEST_p$:

1. Если $\exists q \in Neigh_p : cfrag_p[q] = \perp$, то:
 - 1.1 Выбрать указанный выше q с наименьшим значением $\omega_p[q]$
 - 1.2 $c_{test}_p := q$;
 - 1.3 $send_q(\mathbf{test}, rank_p, name_p)$
2. Иначе:
 - 2.1 $c_{test}_p := \perp$;
 - 2.2 $REPORT_p$

Узел p в рамках поиска минимального граничного ребра

- ▶ выбирает инцидентное ребро типа \perp (неопробованное) с наименьшим весом и отправляет в него сообщение **test** для проверки того, является ли оно граничным,
- ▶ а если таких рёбер нет, то считает поиск завершённым и докладывает о результатах в сторону стержня

GHS: приём *initiate* узлом p

Процедура $REPORT_p$:

1. Если $cN_p = |\{q \mid q \in Neigh_p \setminus \{parent_p\}, cfrag_p = \text{t}\}|$ и $c_{test}_p \neq \perp$:
 - 1.1 $state_p := found$;
 - 1.2 $send_{parent_p}(\mathbf{report}, \omega_{best}_p)$
-

Доклад о результатах поиска наилучшего граничного ребра устроен так

Если узел получил от всех детей мнения о наилучших граничных рёбрах и завершил свой локальный поиск наилучшего граничного ребра, то он отправляет вес наилучшего известного ему граничного ребра родителю (в сторону стержня)

GHS: приём test узлом p

Предусловие: в канале хотя бы от одного соседа q есть сообщение (**test**, $rank$, $name$), где $rank \leq rank_p$

1. receive $_q$ (**test**, $rank$, $name$) для сообщения и q из условия
2. Если $name \neq name_p$ (это граничное ребро): send $_q$ (**accept**)
3. Иначе (это внутреннее ребро):
 - 3.1 Если $cfrag_p[q] = \perp$: $cfrag_p[q] := f$;
 - 3.2 Если $q \neq ctest_p$: send $_q$ (**reject**)
Иначе: $TEST_p$

Если p получил от узла q нестаршего фрагмента сообщение о проверке граничности ребра, то:

- ▶ Если фрагменты p и q разные, то p отвечает, что ребро граничное
- ▶ Иначе p меняет тип ребра на «внутреннее» и в зависимости от того, проверяет ли сейчас это ребро, либо продолжает перебирать инцидентные рёбра (запускает $TEST_p$), либо сообщает q о том, что ребро внутреннее

GHS: приём test узлом p

Предусловие: в канале хотя бы от одного соседа q есть сообщение (**test**, $rank$, $name$), где $rank \leq rank_p$

1. receive $_q$ (**test**, $rank$, $name$) для сообщения и q из условия
 2. Если $name \neq name_p$ (это граничное ребро): send $_q$ (**accept**)
 3. Иначе (это внутреннее ребро):
 - 3.1 Если $cfrag_p[q] = \perp$: $cfrag_p[q] := f$;
 - 3.2 Если $q \neq ctest_p$: send $_q$ (**reject**)
Иначе: $TEST_p$
-

Вопрос. Почему блокируется приём сообщения **test** от старшего фрагмента, и что происходит с этим сообщением в вычислении в целом?

Вопрос. Почему в (3.2) считается правильным иногда не отправлять ожидаемый ответ **reject** («ребро не граничное») и вместо этого выполнять $TEST_p$?

GHS: приём accept узлом p

Предусловие:

в канале хотя бы от одного соседа q есть сообщение **accept**

1. receive $_q$ (**accept**) для произвольного $q \in Neigh_p$
2. $ctest_p := \perp$;
3. Если $\omega_p[q] < \omega_{best_p}$:
 - 3.1 $\omega_{best_p} := \omega_p[q]$;
 - 3.2 $c_{best_p} := q$;
4. REPORT $_p$

Если на сообщение **test**, отправленное в TEST $_p$, получен ответ, что проверяемое ребро — граничное, то p признаёт это ребро наилучшим инцидентным, завершает поиск и докладывает родителю

Вопрос. Зачем «сбрасывается» значение $test_ch_p$?

Останется ли алгоритм корректным, если удалить этот «сброс»?

GHS: приём **reject** узлом p

Предусловие: в канале хотя бы от одного соседа q есть сообщение **reject**

1. receive $_q$ (**reject**) для произвольного $q \in Neigh_p$
 2. Если $cfrag_p[q] = \perp$: $cfrag_p[q] := \mathbb{f}$;
 3. $TEST_p$
-

Если на сообщение **test** получен ответ, что проверяемое ребро является внутренним, то тип ребра соответственно обновляется и поиск наилучшего инцидентного граничного ребра продолжается

GHS: приём report узлом p

Предусловие: $state_p = found$ и в канале хотя бы от одного соседа q , кроме $parent_p$, есть сообщение типа **report**

1. receive $_q(\mathbf{report}, \omega)$ для произвольного $q \in Neigh_p$
2. Если $q \neq parent_p$:
 - 2.1 Если $\omega < \omega_{best_p}$: $(\omega_{best_p}, c_{best_p}) := (\omega, q)$;
 - 2.2 $cN_p := cN_p + 1$;
 - 2.3 $REPORT_p$
3. Иначе:
 - 3.1 Если $\omega > \omega_{best_p}$: $CHANGEROOT_p$
 - 3.2 Иначе: если $\omega = \omega_{best_p} = \infty$, то завершить выполнение узла

Если доклад получен от ребёнка, то p обновляет мнение о наилучшем граничном ребре и докладывает родителю

GHS: приём report узлом p

Предусловие: $state_p = found$ и в канале хотя бы от одного соседа q , кроме $parent_p$, есть сообщение типа **report**

1. receive $_q$ (**report**, ω) для произвольного $q \in Neigh_p$
2. Если $q \neq parent_p$:
 - 2.1 Если $\omega < \omega_{best_p}$: $(\omega_{best_p}, c_{best_p}) := (\omega, q)$;
 - 2.2 $cN_p := cN_p + 1$;
 - 2.3 $REPORT_p$
3. Иначе:
 - 3.1 Если $\omega > \omega_{best_p}$: $CHANGEROOT_p$
 - 3.2 Иначе: если $\omega = \omega_{best_p} = \infty$, то завершить выполнение узла

Если же доклад получен от родителя, то это доклад от одного стержневого узла другому (они и только они являются взаимными родителями), и стержневой узел, нашедший ребро лучшего веса, запускает процедуру $CHANGEROOT_p$ отправки сообщения **connect** в ребро с этим весом

GHS: приём report узлом p

Предусловие: $state_p = found$ и в канале хотя бы от одного соседа q , кроме $parent_p$, есть сообщение типа **report**

1. receive $_q(\mathbf{report}, \omega)$ для произвольного $q \in Neigh_p$
2. Если $q \neq parent_p$:
 - 2.1 Если $\omega < \omega_{best_p}$: $(\omega_{best_p}, c_{best_p}) := (\omega, q)$;
 - 2.2 $cN_p := cN_p + 1$;
 - 2.3 $REPORT_p$
3. Иначе:
 - 3.1 Если $\omega > \omega_{best_p}$: $CHANGEROOT_p$
 - 3.2 Иначе: если $\omega = \omega_{best_p} = \infty$, то завершить выполнение узла

Если оба стержневых узла не нашли ни одного граничного ребра, то процедура построения MST в узле завершается

GHS: приём report узлом p

Предусловие: $state_p = found$ и в канале хотя бы от одного соседа q , кроме $parent_p$, есть сообщение типа **report**

1. receive $_q$ (**report**, ω) для произвольного $q \in Neigh_p$
 2. Если $q \neq parent_p$:
 - 2.1 Если $\omega < \omega_{best_p}$: $(\omega_{best_p}, c_{best_p}) := (\omega, q)$;
 - 2.2 $cN_p := cN_p + 1$;
 - 2.3 $REPORT_p$
 3. Иначе:
 - 3.1 Если $\omega > \omega_{best_p}$: $CHANGEROOT_p$
 - 3.2 Иначе: если $\omega = \omega_{best_p} = \infty$, то завершить выполнение узла
-

Вопрос. Почему сообщение типа **report**

может быть получено только от узла текущего фрагмента?

GHS: приём report узлом p

Процедура $CHANGEROOT_p$:

1. Если $cfrag_p[cbest_p] = \perp$: $send_{cbest_p}(\mathbf{changeroot})$
 2. Иначе:
 - 2.1 $send_{cbest_p}(\mathbf{connect}, rank_p)$
 - 2.2 $cfrag_p[cbest_p] := \perp$;
-

Сообщение **changeroot** передаётся внутри фрагмента в сторону узла, которому инцидентно найденное минимальное граничное ребро

В это ребро отправляется сообщение **connect** о намерении объединиться с фрагментом на другом конце ребра

GHS: приём **changeroot** в узле p

Предусловие:

в канале хотя бы от одного соседа q есть сообщение **changeroot**

1. $\text{receive}_q(\mathbf{changeroot})$ для произвольного $q \in \text{Neigh}_p$
 2. CHANGEROOT_p
-

Д.з. 2. Ответьте хотя бы на три поставленных ранее вопроса

GHS: свойства

Теорема. Любое вычисление GHS конечно, и в заключительной конфигурации условие $cfrag_p[q] = \text{t}$ верно тогда и только тогда, когда ребро $p - q$ входит в MST

Доказательство (основная идея).

Из описания и подробных пояснений должно быть ясно, что сообщение **connect** отправляется только в минимальное граничное ребро фрагмента

По **утверждению 2**, если показать, что каждый фрагмент, имеющий хотя бы одно граничное ребро, отправляет сообщение **connect**, то алгоритмом действительно за конечное число действий строится MST в том смысле, как написано в условии теоремы

GHS: свойства

Доказательство (основная идея).

Предположим от противного, что в вычислении возникает фрагмент, отличный от MST, ни один узел которого никогда не отправляет **connect**

Согласно описанию алгоритма, это означает, что в каждом фрагменте на очередном этапе выполнения действий этого фрагмента происходит **блокировка**: бесконечное безрезультатное ожидание того, что предусловие очередного действия станет истинным

Согласно схеме пересылки сообщений внутри фрагмента и между фрагментами, блокировка фрагмента F_1 (с рангом $rank_1$ и именем $name_1$) возможна только в тех случаях, когда сообщение из F_1 отправляется узлу другого фрагмента F_2 (с рангом $rank_2$ и именем $name_2$) и никогда не принимается в F_2 из-за невыполненного предусловия, и это возможно только для двух сообщений:

1. (**connect**, $rank_1$)
2. (**test**, $rank_1$, $name_1$)

GHS: свойства

Доказательство (основная идея).

Для каждого из этих видов сообщений невозможность принять его в F_2 означает, что верно одно из следующих условий:

1. $rank_1 > rank_2$
2. $rank_1 = rank_2$ и $\omega_{e_{F_1}} > \omega_{e_{F_2}}$
3. $rank_1 = rank_2$, $\omega_{e_{F_1}} = \omega_{e_{F_2}}$, и в F_2 выполняется поиск минимального граничного ребра

Д.з. 3. Показать, что в GHS недостижимы конфигурации, в которых для каждого фрагмента выполнено хотя бы одного из трёх условий выше

Это означает, что если существует текущий фрагмент, отличный от MST, то существует и незаблокированный фрагмент (*противоречие*) ▼

Д.з. 4. Оценить, сколько сообщений каждого типа отправляется при выполнении алгоритма, и на основании этой оценки показать, что коммуникационная сложность GHS не превосходит $2|V| \log_2 |V| + 5|E|$