Ответы на экзаменационные задачи по курсу

“Распределённые алгоритмы и системы” (2016)

*Составители:*

*Степанов Евгений*

*avasite*

**Оглавление**

[Задача 1. (л1 стр 166) Покажите, что в «наивном» протоколе передачи данных с 4 сообщениями (Лекция 1) возможно дублирование или потеря сообщений ввиду того, что NCP A вынуждена считаться с возможностью выхода из строя NCP B. 5](#_Toc454003781)

[Задача 2. (л1 стр 177) Постройте простой протокол с двумя обменами сообщениями, который никогда не допускает потери сообщений (хотя может дублировать сообщения). Докажите корректность этого протокола (т.е., что построенный протокол никогда не теряет ни одного сообщения). 5](#_Toc454003782)

[Задача 3. (л2 стр 88) Докажите теорему. 5](#_Toc454003783)

[Задача 4. (л2 стр 101) Докажите, что отношение причинно-следственной зависимости между событиями выполнения ≺ является отношением частичного порядка? При каких условиях это отношение будет являться отношением полного (линейного) порядка? 6](#_Toc454003784)

[Задача 5 ???. (л2 стр 117) Подберите подходящее определение для причинно-следственного порядка переходов системы с синхронным обменом сообщениями. Предложите определение часов для систем такого рода и постройте распределенные алгоритмы для вычисления показаний этих часов. 7](#_Toc454003785)

[Задача 6. (л2 стр 123) Докажите, что существуют такие распределенные системы, которые не способны вычислять функцию глобальных часов Θg. 7](#_Toc454003786)

[Задача 7. (л2 стр 128) Покажите, что функция Лэмпорта L действительно является часами. 8](#_Toc454003787)

[Задача 8. (л2 стр 137) 8](#_Toc454003788)

[Задача 9. (л3 стр 41) Верно ли, что утверждение, которое является истинным в каждой конфигурации любого выполнения, обязательно является инвариантом? 8](#_Toc454003789)

[Задача 10. (л3 стр 41) Приведите пример такой системы переходов S и такого утверждения P, что P всегда истинно в системе S, но при этом не является инвариантом S. 9](#_Toc454003790)

[Задача 11. (л3 стр 51) Предположим, что P1 и P2 это инварианты системы S. Докажите, что (P1 or P2) и (P1 and P2) также являются инвариантами. 9](#_Toc454003791)

[Задача 12. (л3) 9](#_Toc454003792)

[Задача 13. (л3 стр 92, 126) Покажите, что симметричный протокол раздвижного окна не удовлетворяет требованию неизбежной доставки сообщения, если из двух допущений справедливости F1 и F2 выполняется только допущение F2. 10](#_Toc454003793)

[Задача 14. (л3 стр 92, 126) Будет ли симметричный протокол раздвижного окна удовлетворять требованию неизбежной доставки сообщения, если будет выполняться только допущение F1? 10](#_Toc454003794)

[Задача 15. (л3 стр 94, 126) Докажите, что если в симметричном протоколе раздвижного окна lp + lq = 1 и начальные значения переменных ap и aq равны -lq и -lp, то равенства ap + lq = sp и aq + lp = sq всегда выполняются. 10](#_Toc454003795)

[Задача 16. (л3 стр 130, 94) 10](#_Toc454003796)

[Задача 17. (л3 стр 130, 94) 10](#_Toc454003797)

[Задача 18. 11](#_Toc454003798)

[Задача 19. (л3 стр 144) Покажите, что в случае L = 1 в протоколе раздвижного окна достаточно использовать только одну из двух переменных ap или sp (и только одну из переменных aq или sq). 12](#_Toc454003799)

[Задача 20. Почему никакой протокол не может предоставить гарантии того, что слово будет доставлено по назначению за ограниченный срок времени? 12](#_Toc454003800)

[Задача 21. (л4 стр 108) 12](#_Toc454003801)

[Задача 22. (л4 стр 119) 13](#_Toc454003802)

[Задача 23. (л4) В протоколе с таймерами отправитель может занести в отчет слово как возможно утраченное, в то время как это слово было благополучно доставлено получателю. Опишите выполнение этого протокола, в ходе которого происходит подобный эффект. 14](#_Toc454003803)

[Задача 24. (л4) Предположим, что в связи с выходом из строя часового механизма, получатель не может закрыть сеанс связи вовремя. Опишите вычисление протокола с таймерами, в ходе которого слово будет утрачено, но отправитель не сможет отметить это в отчете. 15](#_Toc454003804)

[Задача 25. (л4) Опишите такое вычисление протокола с таймерами, в ходе которого получатель открывает сеанс связи после получения пакета с порядковым номером большим нуля. 15](#_Toc454003805)

[Задача 26. (л4) 15](#_Toc454003806)

[Задача 27. (л5 стр 53) Допустим, что таблицы маршрутизации так обновляются после каждого изменения топологической структуры сети, что они остаются ациклическими по ходу обновления. Может ли это служить гарантией того, что пакеты всегда доставляются по адресу даже в том случае, когда сеть претерпевает бесконечно большое количество топологических изменений? 16](#_Toc454003807)

[Задача 28. (л5 стр 53) Докажите, что ни один алгоритм маршрутизации не способен обеспечить доставку пакетов по адресу, если сеть испытывает непрерывные изменения топологии. 16](#_Toc454003808)

[Задача 29. (л5 стр 105) Зачем в алгоритме маршрутизации Туэга нужно передавать в каждом сообщении имя текущей опорной вершины w? 16](#_Toc454003809)

[Задача 30. (л5 стр 105) Можно ли исключить из алгоритма Туэга отправление сообщений ⟨nys, w⟩? Будет ли модифицированный таким образом алгоритм корректным? 16](#_Toc454003810)

[Задача 31. (л5 стр 122, 123) Докажите, что приведенная ниже формула задает инвариант алгоритма Чанди-Мисры вычисления путей, ведущих в вершину v0. 17](#_Toc454003811)

[Задача 32. (л5 стр 125) В описании алгоритма Чанди-Мисры не указывается, до каких пор должно проводиться вычисление маршрутов в каждом процессе. Докажите, в любой конфигурации любого выполнения алгоритма Чанди-Мисры промежуточные таблицы маршрутизации являются ациклическими. Что нужно добавить к алгоритму Чанди-Мисры, для того чтобы каждый процесс мог узнать, что построение таблиц маршрутизации в сети завершено. 17](#_Toc454003812)

[Задача 33. (л6 стр 14, 17, 73) Выясните, какие значения будут иметь все переменные в заключительной конфигурации алгоритма Netchange в том случае, когда этот алгоритм применяется к сети, имеющей следующую топологическую структуру ...: 18](#_Toc454003813)

[Задача 34. (л7 стр 149) Покажите, что в каждом вычислении древесного алгоритма в точности два процесса принимают решение. 18](#_Toc454003814)

[Задача 35. ??? (л7 стр 38) Привести пример сети, в которой произвольный волновой алгоритм произведет ровно N обменов сообщениями, где N число процессов в сети. 19](#_Toc454003815)

[Задача 36. ??? (л7 стр 43) Привести пример сети, в которой произвольный волновой алгоритм произведет ровно K обменов сообщениями, где K число каналов в сети. 19](#_Toc454003816)

[Задача 37. (л7) Придумать кольцевой алгоритм с двумя инициаторами и доказать его корректность. 19](#_Toc454003817)

[Задача 38. (л7) Придумать кольцевой алгоритм с двумя инициаторами и доказать его корректность. 20](#_Toc454003818)

[Задача 39. (л8 стр 25) В доказательстве фазового алгоритма доказывается, что отношение fpq(i) ≼ gpq(i) сохраняется даже если канал pq не является очередью. Показать, что данное отношение сохраняется даже если канал pq является каналом с потерей сообщений. Показать, что данное отношение не сохраняется если в канале pq сообщения могут дублироваться. 20](#_Toc454003819)

[Задача 40. (л8 стр 11) Показать, что модифицированный фазовый алгоритм, который вырабатывает решение после принятия D - 1 сообщений от каждого соседа на входе, не является волновым алгоритмом. 20](#_Toc454003820)

[Задача 41. (л8) Предположим, что Вы намереваетесь использовать волновые алгоритмы в сетях с дублированием сообщений в каналах. Какие изменения необходимо внести в алгоритм Эха. Какие изменения необходимо внести в алгоритм Финна. 20](#_Toc454003821)

[Задача 42. (л8 стр 129) Привести вычисление алгоритма Тарри, в результате которого получается дерево, не являющееся деревом поиска в глубину. 21](#_Toc454003822)

[Задача 43. (л7 стр 121) Адаптируйте алгоритм эха для вычисления суммы входных данных всех процессов. 21](#_Toc454003823)

[Задача 44. (л8 стр 44) Доказать теорему: Используя фазовый алгоритм избрание лидера может быть произведено за O(D|E|) обменов сообщениями и используя O(D) единиц времени. 21](#_Toc454003824)

[Задача 45. (л9 стр 13) Докажите, что алгоритм избрания лидера путем сравнения (отыскания экстремумов) является волновым алгоритмом, если событие избрания процесса лидером рассматривать как событие решения. 22](#_Toc454003825)

[Задача 46. (л9 стр 38) Доказать теорему: Алгоритм Ченя-Робертса решает задачу избрания лидера, используя менее (N2) сообщений и O(N) единиц времени. 22](#_Toc454003826)

[Задача 47. (л9 стр 46) 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? 23](#_Toc454003827)

[Задача 48. (л9) Привести начальную конфигурацию сети, в которой алгоритм Патерсона-Долева-Клейва-Роде требует (⌊logN⌋ +1) раундов. Также привести начальную конфигурацию, в которой алгоритму достаточно двух раундов вне зависимости от числа инициаторов. Возможно ли алгоритму завершиться за один раунд? 23](#_Toc454003828)

[Задача 49. Сравнить алгоритм угасания волны для колец с алгоритмом Ченя-Робертса. В чем различия и какой они имеют эффект? 24](#_Toc454003829)

[Задача 50. (л10 стр 58) Установить для сообщений всех семи типов алгоритм Галладжера-Хамблета-Спиры, может ли каждое из них быть послано процессу, находящемуся в состоянии сна. 24](#_Toc454003830)

[Задача 51. ??? (л10 стр 107) Алгоритм Этья (Attiya) отличается от алгоритма Кораха-Каттена-Морана только тем, что фишка вместо того, чтобы впасть в состояние погони, замирает и посылает сообщение-убийцу, которое догоняет преследуемую фишку, убивает ее и возвращается назад. После возвращения убийцы выполнение обхода продолжается. Реализовать алгоритм Attiya и установить его сложность. 24](#_Toc454003831)

[Задача 52. (л11 стр 23) Временная сложность алгоритма завершения вычисления называется число единиц времени между завершением базового алгоритма и завершением контрольного алгоритма. Какова временная сложность алгоритма Дейкстры-Шолтена? 24](#_Toc454003832)

[Задача 53. ??? Если алгоритм Шави-Франчеза применяется к произвольной сети, процессы которой имеют уникальные идентификаторы, и при этом используется волновой алгоритм Галледжера-Хамблета-Спиры, то временная сложность такого контрольного алгоритма будет составлять (NlogN). Можно ли улучшить временную сложность до O(N) ценой обмена O(N) дополнительными сообщениями? 25](#_Toc454003833)

[Задача 54. ??? (л11 стр 53) Показать существование такого базисного вычисления, в котором происходит обмен m сообщениями, при контроле которого алгоритм Сафры использует m(N - 1) сообщений. 25](#_Toc454003834)

[Задача 55. (л11 стр 58) В алгоритме Раны предполагается, что процессы наделены отличительными признаками. Предположим теперь, что все процессы анонимны, но обладают возможностью отправлять сообщения своим последователям по кольцу, и при этом число процессов заранее известно. Внесите в алгоритм Раны необходимые изменения, позволяющие ему работать в рамках таких допущений. 25](#_Toc454003835)

[Задача 56. (л11 стр 58) Обоснуйте корректность алгоритма Раны на основе инвариантов этого алгоритма. 26](#_Toc454003836)

[Задача 57. (л11 стр 58) Внесите изменения в алгоритм Раны так, чтобы для передачи сообщений можно было использовать произвольный волновой алгоритм, а не только кольцевой алгоритм. 26](#_Toc454003837)

[Задача 58. (л12) Предположим, что снятие снимка процессом p является дополнительным внутренним событием ap. Показать, что S\* значимый срез ⇔ ∀p, q ap not≼ aq & aq not≼ ap: 26](#_Toc454003838)

[Задача 59. (л2) Будем рассматривать регистрацию моментального локального состояния процесса p как еще одно внутреннее событие ap. Докажите, что S\* является значимым ⇔ ∀p, q : ap ∥ aq 27](#_Toc454003839)

[Задача 60. ??? Профессор Захаров утверждает: 27](#_Toc454003840)

# Задача 1. (л1 стр 166) Покажите, что в «наивном» протоколе передачи данных с 4 сообщениями (Лекция 1) возможно дублирование или потеря сообщений ввиду того, что NCP A вынуждена считаться с возможностью выхода из строя NCP B.

Протокол передачи с 4 сообщениями:

1. NCP А send (data, т, х)

2. NCP В receive (data, т, х), send (open, т, х, у)

3. NCP А receive (open, т, х, у), send (agree, т, х, у)

4. NCP В receive (agree, т, х, у), deliver т, send (аск, х, у), close

5. NCP А receive (аск, х, у), notify, close

В связи с тем, что приходится учитывать возможность выхода из строя NCP В, исправление ошибок неизбежно будет проводиться так, что дублирование сообщений возникнет даже тогда, когда ни одна из NCP в действительности не выйдет из строя. NCP В отправляет сообщение об ошибке (посоп, х, у), когда получено сообщение (agree, т, х, у), но никакой сеанс связи не открыт. Предположим, что NCP А не получает сообщения (аск, х, у) даже после нескольких повторных отправлений сообщения (agree, т, х, у); в ответ приходят только сообщения (посоп, х, у). Поскольку не исключена возможность выхода из строя NCP В перед тем, как она получила сообщение (agree, т, х, у), NCP А вынуждена открыть новый сеанс связи (отправив сообщение (data, т, х)), чтобы предотвратить потерю информации т\ Но ведь NCP В могла с равным успехом уже доставить информацию т, и просто было потеряно сообщение (аск, х, у). В этом случае произойдет дублирование информации.

# Задача 2. (л1 стр 177) Постройте простой протокол с двумя обменами сообщениями, который никогда не допускает потери сообщений (хотя может дублировать сообщения). Докажите корректность этого протокола (т.е., что построенный протокол никогда не теряет ни одного сообщения).

Опишем протокол с двумя обменами сообщениями, при этом введем идентификационный номер сообщения id, о котором знает NPC А:

1. NCP A send (data, т, id)

2. NCP В receive (data, т, id), deliver т, send (ack, id), close

3. NCP A receive (ack, id), notify, close

Док-во корректности данного протокола:

Предположим обратное, пусть сообщение т с номером id1 было потеряно. Это означает, что процесс А получил уведомление о доставке данного сообщения (notify), но процесс B это сообщение не получал. Процедура notify может быть вызвана у NPC A только на третьем шаге протокола после получения сообщения ack с номером id1. Такое сообщение может быть отправлено только NPC B в случае, когда было получено сообщение с типом data и таким же идентификационным номером (id1) и оно было доставлено процессу B. Так как сообщение с типом data для любой другой полезной нагрузки (не сообщение т) будет содержать другой идентификационный номер, то это означает, что сообщение т было доставлено процессу B. Получаем противоречие, значит потеря сообщения в данном протоколе невозможна.

# Задача 3. (л2 стр 88) Докажите теорему.

Теорема.

Пусть j — конфигурация распределенной системы (с асинхронным обменом сообщениями), и пусть ep и eq события, которые происходят в разных процессах p и q, и при этом оба события допустимы в конфигурации j. Тогда событие ep допустимо в конфигурации eq(j), а событие eq в конфигурации ep(j), и при этом ep(eq(j)) = eq(ep(j)).

Доказательство. Чтобы избежать поочередного разбора разных случаев в зависимости от того, с событиями какого типа (внутренними, отправления сообщения или приема сообщения) мы имеем дело, введем единообразное обозначение (с, х, у, d) для каждого события. Здесь символы с и d обозначают состояния процесса до осуществления события и после его осуществления, х — это совокупность сообщений, принятых по ходу осуществления события, a y — это совокупность сообщений, отправленных при осуществлении этого события. Таким образом, внутреннее событие (с, d) будет обозначаться записью (с, 0, 0, d) (0 — это пустое мн-во), событие отправления сообщения (с, т, d) — записью (с, 0, {т}, d), а событие приема сообщения (с, т, d) — записью (с, {т}, 0, d). В рамках введенной системы обозначений событие е = (с, х, у, d) процесса р допустимо в конфигурации у = (сp1, ..., сp, ..., cpn, М), если ср = с и х ∊ М. В этом случае

е(у) = (сp1, ..., d, ..., (M\x)∪y).

Теперь предположим, что события ep = (bp, хр, ур, dp), eq = (bq, xq, yq, dq) допустимы в конфигурации

y = (..., ср, ..., cq, ..., М),

т. е. ср = bр, cq = bq, хр ∊ М и xq ∊ М. Важную роль здесь играет то обстоятельство, что множества хр и xq не пересекаются, так как адресатом сообщения хр (если таковое имеется) является процесс р, тогда как адресатом сообщения xq (если таковое имеется) является процесс q.

Рассмотрим конфигурацию ур = ep(у) и заметим, что

ур = (..., dp, ..., cq, ..., (М \ хр) ∪ ур ).

Так как xq ∊ M и xq∩xp = 0, справедливо включение xq ∊ (M\xp∪yp), и поэтому событие eq является допустимым в конфигурации ур. Рассмотрим конфигурацию ypq = eq(yp) и заметим, что

ypq = (…, dp, ..., dq, ..., ((М \ хр ∪ ур) \ xq) ∪ yq )

Учитывая, что рассматриваемые процессы равноправны, мы можем, применяя те же рассуждения, показать, что событие ep является допустимым в конфигурации уq = eq(y). Рассмотрим конфигурацию yqp = ep(yq) и заметим, что

yqp = (…, dp, ..., dq, ..., ((М \ xq ∪ yq) \ хр ∪ ур))

Поскольку М — это мультимножество сообщений и при этом xp ∊ M и xq ∊ М, справедливо равенство

((М \ хр ∪ ур) \ xq ∪ yq) = ((М \ xq ∪ yq) \ хр ∪ ур),

и поэтому ypq = yqp.

# Задача 4. (л2 стр 101) Докажите, что отношение причинно-следственной зависимости между событиями выполнения ≺ является отношением частичного порядка? При каких условиях это отношение будет являться отношением полного (линейного) порядка?

(На самом деле) Рассмотрим отношение a ≺= b, которое определяется формулой (a ≺ b V a = b) .

1) Это отношение рефлексивно, так как a=a.

2) Это отношение антисимметрично, так как если a и b — разные события одного процесса, то лишь одно из них происходит раньше другого (предположим, что это событие a, тогда верно a ≺= b, но неверно b ≺= a). Если a и b — события разных процессов, то должны существовать события s1, s2, r1, r2 удовлетворяющие следующим условиям:

(1) a ≺= s1 ≺= r1 ≺= b

(2) b ≺= s2 ≺= r2 ≺= a

s1 — событие отправления сообщения в процессе с событием а.

r1 — событие приема сообщения в процессе с событием b, транзитивно или напрямую вызванное событием s1.

s2 — событие отправления сообщения в процессе с событием b.

r2 — событие приема сообщения в процессе с событием a, транзитивно или напрямую вызванное событием s2.

При этом из r2 ≺= a (2) и a ≺= s1(1) следует, что r2 ≺= s1(3).

А из r1 ≺= b (1) и b ≺= s2 (2) следует, что r1 ≺= s2 (4).

s1 ≺= r1 (1) + (3) + (4) => r2 ≺= s2, что противоречит условию (2). Значит а и b не могут быть событиями из разных процессов.

Значит a ≺= b и b ≺= a выполняются только, когда a = b.

3)Это отношение транзитивно по определению.

Данное отношение будет отношением полного порядка при выполнении следующего условия: для любых двух соседних переходов конфигураций в выполнении E данные переходы могут быть вызваны событиями в разных процессах только в том случае, если первый переход — это отправка сообщения, а второй переход — это прием этого же сообщения. (эквивалентное условие: любые два соседних события удовлетворяют 1 или 2 пункту определения отношения ≺, т.е. являются сравнимыми). При выполнении условия события образуют цепь, где все элементы будут сравнимы по транзитивности. Если это не так, то данные события являются не сравнимыми (нету передачи сообщений между ними для транзитивности) и не будет отношения полного порядка.

# Задача 5 ???. (л2 стр 117) Подберите подходящее определение для причинно-следственного порядка переходов системы с синхронным обменом сообщениями. Предложите определение часов для систем такого рода и постройте распределенные алгоритмы для вычисления показаний этих часов.

(Взято из статьи)

S1 If a<(i)b, then a « b. (<(i) — предшествует в одном процессе, « - сильный причинно-следственный порядок)

S2 If (s, r) принадлежит Г, для любого a из C : a « s ~ a « r and s « a ~ r « a.

S3 If a « b and b « c, then a « c.

“*не до конца решена задача про логические часы в системах с синхронной передачей сообщения, там скорее всего должны быть часы с определением через сильную причинно-следственную связь, которая вводится в начале ответа*”

# Задача 6. (л2 стр 123) Докажите, что существуют такие распределенные системы, которые не способны вычислять функцию глобальных часов Θg.

Такой функцией может воспользоваться сторонний наблюдатель системы, который может наблюдать последовательность происходящих событий. Однако эту последовательность нельзя наблюдать, пребывая внутри системы; иначе говоря, распределенный алгоритм не способен вычислять функцию Θg. Это следует из теоремы 2.19 (Пусть j — конфигурация распределенной системы (с асинхронным обменом сообщениями), и пусть ep и eq события, которые происходят в разных процессах p и q, и при этом оба события допустимы в конфигурации j. Тогда событие ep допустимо в конфигурации eq(j), а событие eq в конфигурации ep(j), и при этом ep(eq(j)) = eq(ep(j)).). Предположим, что какой-то распределенный алгоритм записывает значение Θg(ei) = i для одного из тех событий еi, которые фигурируют в предпосылке теоремы. Рассмотрим теперь эквивалентное выполнение, в котором изменен порядок осуществления двух событий; хотя настоящее значение функции Θg изменилось, процесс будет записывать то же самое значение i. Иными словами, функция Θg определена на множестве выполнений, а не вычислений.

# Задача 7. (л2 стр 128) Покажите, что функция Лэмпорта L действительно является часами.

Доказательство от противного. Пусть функция Лэмпорта не является часами, тогда a ≺ b => Θ(a) >= Θ(b). Значит для события b длина максимальной последовательности событий (е1, …, b) равна Θ(b). Для события а длина максимальной последовательности событий (е1, …, a) равна Θ(a). Так как a ≺ b, то существует цепочка событий, приводящих от а к b, тогда рассмотрим максимальную цепочку для события а и добавим к ней построенную цепочку до события b. Получили новую цепочку (e1, …, a, …, b). Длина этой цепочки минимум на 1 больше чем, Θ(a). Значит длина одной из цепочек для b > Θ(a) + 1. Однако из предположения следует, что длина максимальной цепочки для b меньше или равна Θ(a). Получаем противоречие, значит функция Лэмпорта является часами.

# Задача 8. (л2 стр 137)

Можно ли построить такую функцию часов Θ, которая

1. могла быть вычислена распределенным алгоритмом;

2. для любого вычисления и для любых двух событий a и b в этом вычислении обладала свойством a ≺ b <=> Θ(a) < Θ(b):

Векторные часы Маттерна

Векторные часы в системепроцессов — массив или вектор излогических часов, одни часы на процесс. Локальный экземпляр вектора с наименьшими возможными значениями часов для каждого процесса строится следующим образом:

* изначально все значения часов равны 0;
* в случае внутреннего события счётчик текущего процесса увеличивается на 1;
* перед отправкой сообщения внутренний счётчик, соответствующий текущему процессу, увеличивается на 1, и вектор целиком прикрепляется к сообщению;
* при получении сообщения счётчик текущего процесса увеличивается на 1, далее значения в текущем векторе выставляются в максимум от текущего и полученного.

Сравнение векторов: (a1, …, an) <= (b1, …, bn) <=> для любого i: ai<=bi

=> для событий a и b одного процесса Θ(a) < Θ(b) за счет увеличения собственного счетчика, (остальные счетчики могут лишь возрасти ). для события отправления и получения Θ(a) < Θ(b), так как у получателя берется максимум и увеличивается счетчик на 1. Транзитивность сохраняется.

<= Значением функции часов является вектор Θ(b) = (b1, ..., bn), в котором bi — это число событий е, произошедших в процессе pi и удовлетворяющих соотношению е ≺ b. По определению сравнения получаем, что a ≺ b

# Задача 9. (л3 стр 41) Верно ли, что утверждение, которое является истинным в каждой конфигурации любого выполнения, обязательно является инвариантом?

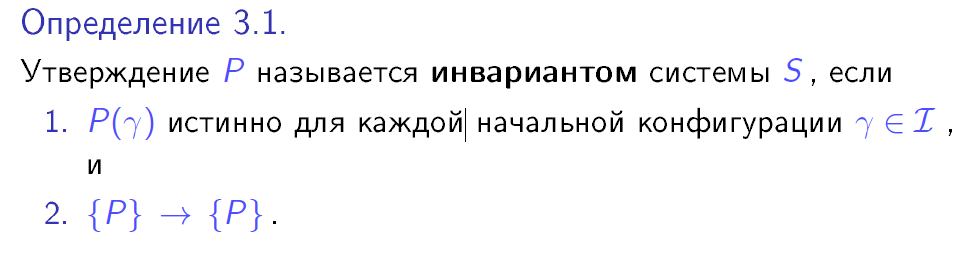
Неверно, из-за недостижимых состояний. Пример однопроцессорной программы:

k=0;

if (k == 1)

k = 2;

Утверждение k<2 истинно при любом выполнении, но при переходе k==1 утверждение перестает быть верным, следовательно не является инвариантом.



# Задача 10. (л3 стр 41) Приведите пример такой системы переходов S и такого утверждения P, что P всегда истинно в системе S, но при этом не является инвариантом S.

То же, что и в предыдущей задаче.

# Задача 11. (л3 стр 51) Предположим, что P1 и P2 это инварианты системы S. Докажите, что (P1 or P2) и (P1 and P2) также являются инвариантами.

1) Так как в начальных конфигурациях истинны P1 и P2 по определению инварианта, то будут истинны и (P1 or P2) и (P1 and P2)

2) Так как { P1} → { P1} и { P2} → { P2}, то и для вариантов с and и or аналогичные утверждения будут выполнены (сравниваются конфигурации, в которых истинны те или иные утверждения)

Из (1) и (2) следует. Что (P1 or P2) и (P1 and P2) также являются инвариантами.

# Задача 12. (л3)

Пусть S параллельная композиция систем S1 и S2.

Докажите, что всякий инвариант P систем S1 и S2 также является инвариантом системы S.

Докажите, что всякая Q-производная P систем S1 и S2 также является Q-производной системы S.

Приведите пример утверждения P, которое является всегда истинным в обеих системах S1 и S2, но не является таковым в системе S.

1) Так как P является инвариантом систем S1 и S2, то Р истинно в начальной конфигурации параллельной композиции, потому что начальные конфигурации всех трех систем совпадают. Так как система переходов является объединением систем переходов S1 и S2, то { P} → { P} будет верно и для параллельной композиции.

2) Опр.: Q-производная P систем S1 и S2:

для любой начальной конфигурации Q=>P

{Q and P} → {Q => P}

Доказательство аналогично первому пункту

3) Три конфигурации, одна внутренняя переменная (со значением 2):

S1: i++; S2: i--;

i--; i++;

P: i > 0;

Однако при параллельной композиции можно два раза вычесть единицу и получить ноль.

# Задача 13. (л3 стр 92, 126) Покажите, что симметричный протокол раздвижного окна не удовлетворяет требованию неизбежной доставки сообщения, если из двух допущений справедливости F1 и F2 выполняется только допущение F2.

Допущение F1 гарантирует, что всякий пакет будет отправляться снова и снова, если не будет получено подтверждение его получения. Если F1 не выполняется, то происходит конечное количество отправки одного сообщения, которые все будут потеряны в канале, после чего может происходить бесконечная отправка другого сообщения. Тогда неизбежная доставка первого сообщения не будет обеспечена.

# Задача 14. (л3 стр 92, 126) Будет ли симметричный протокол раздвижного окна удовлетворять требованию неизбежной доставки сообщения, если будет выполняться только допущение F1?

Не будет. Канал будет бесконечно терять сообщение т несмотря на то, что отправитель его будет бесконечно слать.

# Задача 15. (л3 стр 94, 126) Докажите, что если в симметричном протоколе раздвижного окна lp + lq = 1 и начальные значения переменных ap и aq равны -lq и -lp, то равенства ap + lq = sp и aq + lp = sq всегда выполняются.

По индукции, для начальных значений очевидно (Внимание, в условии начальные значения перепутаны, см. учебник Тэля). В шаге индукции интересует только действие получения. Во-первых, замечаем, что на отправку может отправиться только одно возможное сообщение с i = sq , во-вторых, ap и sq отличаются не больше чем, на 1 и ap <=sq (это следует из той леммы с большим количеством неравенств, крайние значения у нас с разницей в 1, поэтому и средние значения отличаются не больше, чем на 1). В-третьих, ap = max(ap, sq), поэтому значение ap будет увеличиваться на 1 (мы не можем прислать сообщение с номером меньшим, так как мы можем отправлять только одно сообщение и будем отправлять его, пока не придет подтверждение (т.е. оно будет получено). Так как значение sp тоже будет увеличиваться на 1 (мы последовательно шлем сообщения из-за ap ), то ap + lq = sp сохранится и после действия получения сообщения.

# Задача 16. (л3 стр 130, 94)

Докажите следующее утверждение.

Лемма 3.3.

Отправление пакета ⟨pack;w; i⟩ процессом p в протоколе раздвижного окна допустимо только тогда, когда i < ap + L.

Действие Sp, сопровождается логическим условием i < sp + lp, выполнимость этого условия, как следует из леммы 3.1 (куча неравенств), влечет за собой выполнимость неравенства i < ар + L.

# Задача 17. (л3 стр 130, 94)

Докажите следующее утверждение.

Лемма 3.4.

Если outp[i] ̸= udef, то выполняется неравенство i < sp + L.

Согласно свойству (2р) инварианта P справедливо соотношение ар > i — lq. Выполнимость соотношения i < ар + lq, как следует из леммы 3.1 (куча неравенств), влечет выполнимость неравенства i < sp + L.

# Задача 18.

Докажите следующее утверждение.

Теорема 3.7.

Утверждение P′, определяемое следующее формулой

P′ := P ^ ⟨pack;w; i⟩ следует за ⟨pack;w′; i′⟩ в Qp ) => i > i′ - L (4p)

^ ⟨pack;w; i⟩ следует за ⟨pack;w′; i′⟩ в Qq ) => i > i′ - L (4q)

^ ⟨pack;w; i⟩ принадлежит Qp ) => i >= ap - lp (5p)

^ ⟨pack;w; i⟩ принадлежит Qq ) => i >= aq - lq (5q) (в условии была ошибка)

является инвариантом протокола раздвижного окна при условии, что в каналах связи поддерживается очередность передаваемых сообщений.

Доказательство. Так как ранее нам удалось показать, что утверждение Р является инвариантом, можно ограничиться обоснованием того, что свойства (4р), (4q), (5р) и (5q) выполняются в начальных конфигурациях и сохраняются при каждом переходе. Заметим, что в каждой начальной конфигурации очереди сообщений пусты, и поэтому свойства (4р), (4q), (5р) и (5q), очевидно, выполняются. Остается показать, что они сохраняются при переходах системы.

Sp: Чтобы убедиться, что действие Sp сохраняет выполнимость условий (4р) и (5р), заметим, что Sp не добавляет новых пакетов в очередь Qp и не изменяет значения переменной ар.

Чтобы убедиться, что действие Sp сохраняет выполнимость условия (5q), заметим, что если Sр добавляет пакет ⟨pack;w; i⟩ в очередь Qq, то i >= ар. Согласно лемме 3.3 (куча неравенств) отсюда следует, что i >= aq — lq.

Чтобы убедиться, что действие Sp сохраняет выполнимость условия (4q), заметим, что если пакет ⟨pack;w′; i′⟩ уже находится в очереди Qq, то согласно свойству (lq) имеет место неравенство i′ < sp + lp. Поэтому если Sp добавляет в очередь пакет ⟨pack;w; i⟩ с порядковым номером i >= ар, то согласно лемме 3.3 будет выполняться неравенство i′ < ар + L <= i + L.

Rp: Чтобы убедиться, что действие Rp сохраняет выполнимость условий (4р) и (4q), заметим, что Rp не добавляет никаких пакетов ни в очередь Qp, ни в очередь Qq.

Чтобы убедиться, что действие Rp сохраняет выполнимость условия (5р), заметим, что при увеличении значения переменной ар (после приема пакета ⟨pack;w′; i′⟩) до величины i′ — lq + 1 для всех оставшихся пакетов ⟨pack;w; i⟩ из очереди Qp будет выполняться неравенство i > i' — L (в силу свойства (4р)). Следовательно, неравенство i >= ар — lp сохраняет свою выполнимость и после увеличения значения переменной ар.

Чтобы убедиться, что действие Rp, сохраняет выполнимость условия (5q), заметим, что Rp не изменяет состава очереди Qq и значения переменной aq.

Lp: Действие Lp не добавляет никаких пакетов в очереди Qp и Qq и не изменяет значений переменных ар и aq. Поэтому оно сохраняет выполнимость условий (4р), (4q), (5р) и (5q).

Как следует из симметричности рассматриваемого протокола, действия Sq, Rq и Lq также сохраняют справедливость утверждения Р'.

# Задача 19. (л3 стр 144) Покажите, что в случае L = 1 в протоколе раздвижного окна достаточно использовать только одну из двух переменных ap или sp (и только одну из переменных aq или sq).

Из задачи 15 ap + lq = sp и aq + lp = sq.Просто подставляем вместо одной переменной выражение через другую и все.

# Задача 20. Почему никакой протокол не может предоставить гарантии того, что слово будет доставлено по назначению за ограниченный срок времени?

Никакой протокол не может предоставить гарантии того, что слово будет доставлено по назначению за ограниченный срок времени, поскольку не исключена возможность того, что за это время все пакеты будут потеряны.

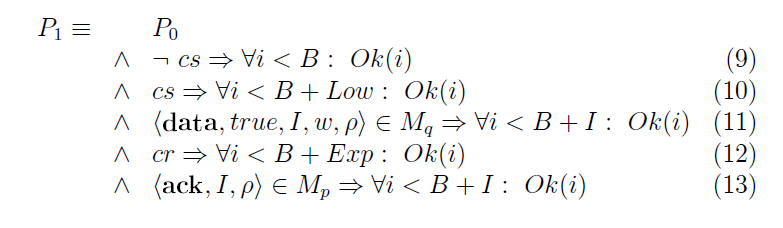
# Задача 21. (л4 стр 108)

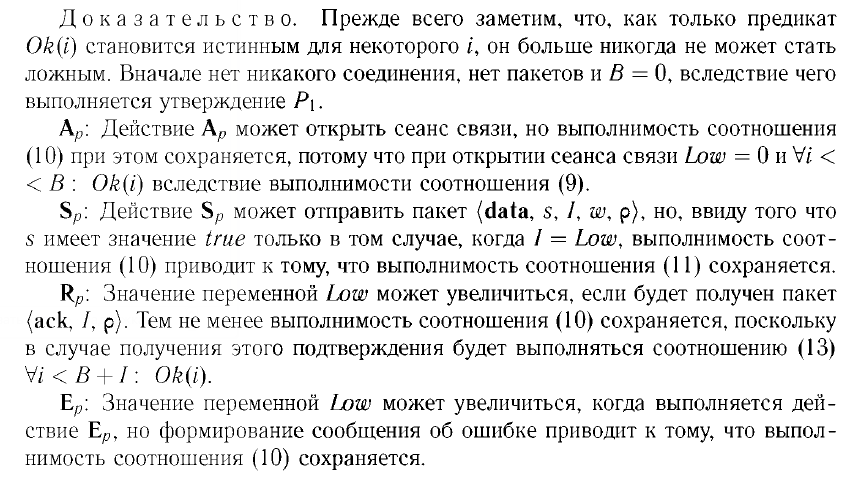
Докажите следующее утверждение.

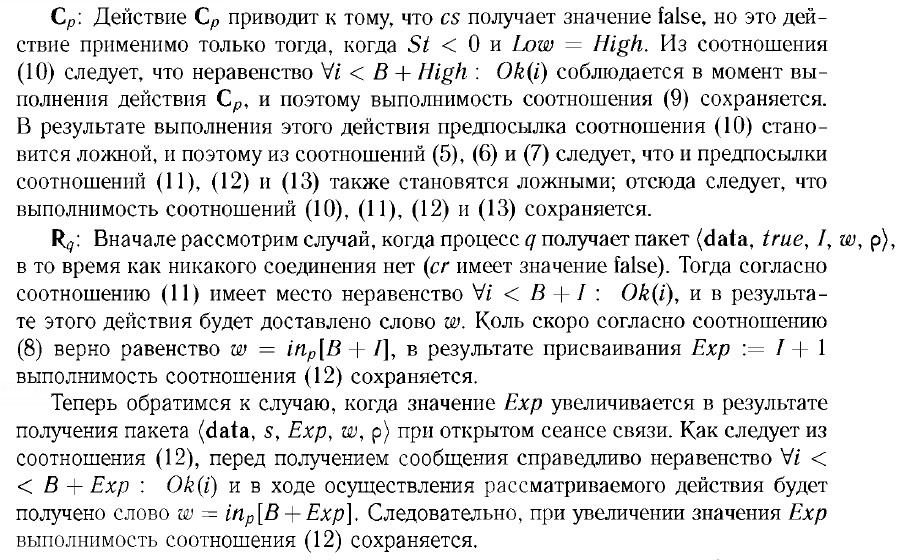
Теорема 4.2.

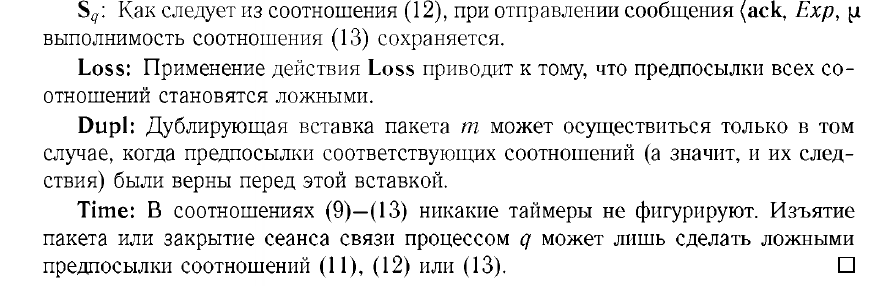
Следующее утверждение является инвариантом протокола с таймерами.

Инвариант необходим для доказательства корректности протокола с таймерами (а именно части про то, что любое сообщение либо доставлено либо занесена запись о том, что оно не доставлено)









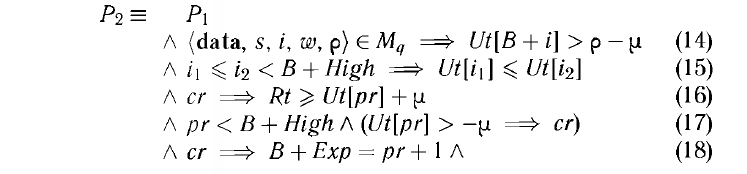
# Задача 22. (л4 стр 119)

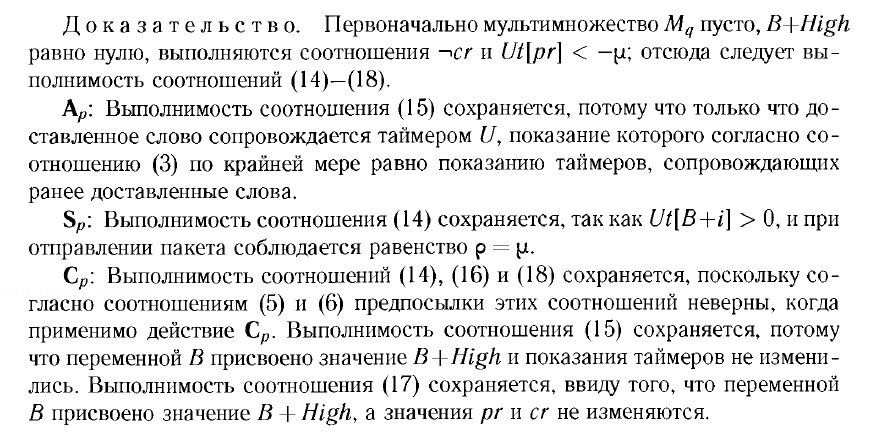
Докажите следующее утверждение.

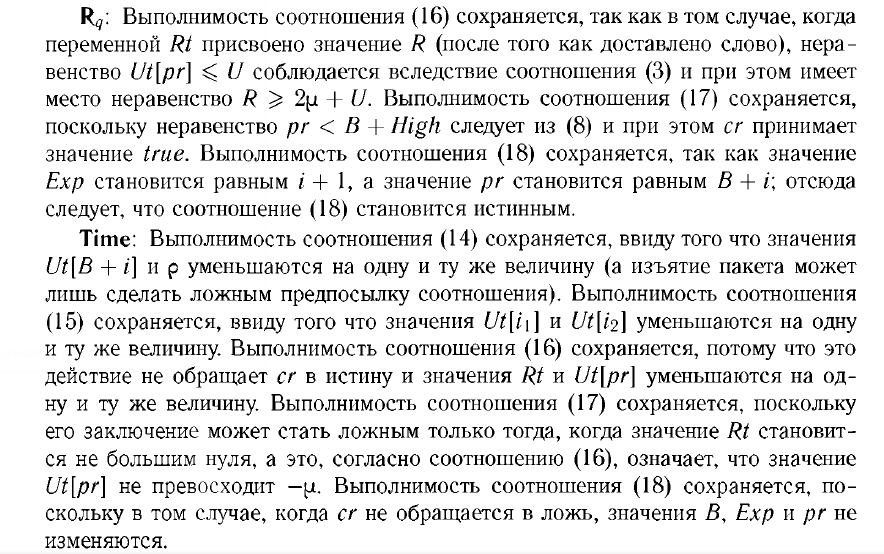
Теорема 4.4. Следующее утверждение является инвариантом протокола с тай-

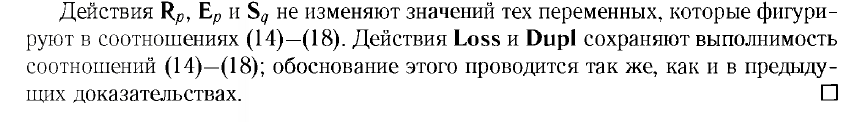
мерами.

Инвариант необходим для доказательства корректности протокола с таймерами (а именно части про то, что соблюдается порядок передачи слов)









# Задача 23. (л4) В протоколе с таймерами отправитель может занести в отчет слово как возможно утраченное, в то время как это слово было благополучно доставлено получателю. Опишите выполнение этого протокола, в ходе которого происходит подобный эффект.

Слово было доставлено получателю, отправляется подтверждение.

Подтверждение теряется в канале.

Истекает таймер отправленного сообщения на отправителе, и он заносит в отчет данное слово как возможно утраченное, хотя оно было доставлено, просто потерялось подтверждение.

# Задача 24. (л4) Предположим, что в связи с выходом из строя часового механизма, получатель не может закрыть сеанс связи вовремя. Опишите вычисление протокола с таймерами, в ходе которого слово будет утрачено, но отправитель не сможет отметить это в отчете.

1) Отправляется пакет с номером 0. (больше от процесса отправителя пакетов нет)

2) Получатель принимает сообщение с номером 0 и отправляет подтверждение с номером 1.

3) Подтверждения теряются, таймер у отправителя истекает, в то время как получатель работает с неисправным таймером.

4) Отправитель закрывает сеанс с сообщением об ошибке.

5) Получатель продолжает слать подтверждение с номером 1.

6) Таймер срабатывает и получатель закрывает сеанс.

7) Отправитель открывает новый сеанс связи и отправляет пакет с номером ноль.

8) Отправитель получает подтверждение с номером 1 с шага 5 и считает, что сообщение доставлено. Он отправляет сообщение с номером 1 и признаком начала последовательности.

9) Сообщение с номером 0 теряется.

10) Получатель открывает сеанс на сообщение с номером 1 с шага 8.

В итоге получаем, что сообщение с номером 0 с шага 7 было потеряно, при этом отправитель ошибочно считает его доставленным.

# Задача 25. (л4) Опишите такое вычисление протокола с таймерами, в ходе которого получатель открывает сеанс связи после получения пакета с порядковым номером большим нуля.

1) Все пакеты с номером N теряются.

2) Получатель закрывает сеанс связи, но у отправителя есть еще сообщения, поступившие с процесса.

3) У отправителя истекает таймер, он засчитывает пакет N, как вероятно потерянный, и прибавляет 1 к LOW. Следующий пакет N+1 будет содержать признак начала последовательности.

4) Получатель получает сообщение с номером N + 1 и откроет сеанс связи с Exp = N + 2, так как в пакете есть признак начала последовательности.

# Задача 26. (л4)

Проектировщик сети хотел бы воспользоваться протоколом с таймерами, но при этом желает, чтобы о возможно утраченных словах запись в отчете осуществлялась ранее. Для этого он модифицирует действие Ep следующим образом:

Ep: (\* Сформировать отчет об ошибке для возможно утраченного слова \*)

{ Ut[B + Low] < 0 }

begin error[B + Low] := true ; Low := Low + 1 end

Будет ли модифицированный таким образом протокол удовлетворять требованиям отсутствия потерь и упорядочения или для этого необходимо внести также другие изменения?

Каковы, по Вашему мнению, преимущества и недостатки указанной модификации?

Требование отсутствия потерь будет выполняться, так как примененные изменения не изменяют значения предиката P0 и P1, использовавшиеся для доказательства отсутствия потерь.

Требование упорядочения тоже сохранится, так как мы можем предполагать, что Ut продолжают уменьшаться даже после выполнения действия Ep. Предикат P2 тогда будет истинен, поэтому требование будет выполнено.

Преимущества данного протокола в такой модификации протокола в более быстрой реакции на отключение получателя (плюс возможно его более быстрое восстановление). Недостатками являются большее количество ошибочных отчетов о возможных потерях.

# Задача 27. (л5 стр 53) Допустим, что таблицы маршрутизации так обновляются после каждого изменения топологической структуры сети, что они остаются ациклическими по ходу обновления. Может ли это служить гарантией того, что пакеты всегда доставляются по адресу даже в том случае, когда сеть претерпевает бесконечно большое количество топологических изменений?

Нет, не может. Предположим, что топологическое изменение происходит на каждое отправление пакета. Вначале маршрут проходит из u1 в u2, после прибытия в вершину u2, маршрут изменяется на ациклический путь из u2 в u1 и далее. После прибытия в вершину u1топология снова изменилась на первоначальную и история повторяется с самого начала. Таким образом, пакет будет постоянно находиться в цикле и никогда не будет доставлен.

# Задача 28. (л5 стр 53) Докажите, что ни один алгоритм маршрутизации не способен обеспечить доставку пакетов по адресу, если сеть испытывает непрерывные изменения топологии.

1-ый вариант Достаточно постоянно отключать канал к получателю, во время последней пересылки пакета.

2-ой вариант Зациклить пакет, изолируя кусочек сети, переменно включая и выключая выходы из этого кусочка.

# Задача 29. (л5 стр 105) Зачем в алгоритме маршрутизации Туэга нужно передавать в каждом сообщении имя текущей опорной вершины w?

Аккуратное программирование позволяет избавиться от параметра w во всех сообщениях, если в каналах связи соблюдается очередность сообщений. Если же очередность следования сообщений нарушается, может случиться так, что узел получит сообщение с параметром w', в то время как он ожидает поступления сообщения с параметром w, причем опорная вершина w' следует по порядку за опорной вершиной w. В этом случае параметр w позволяет разобраться с тем, какому этапу обработки опорных вершин соответствует полученное сообщение; далее, воспользовавшись локальной буферизацией сообщений (в рамках канала или процесса), можно устанавливать правильный порядок обработки сообщений.

# Задача 30. (л5 стр 105) Можно ли исключить из алгоритма Туэга отправление сообщений ⟨nys, w⟩? Будет ли модифицированный таким образом алгоритм корректным?

на первый взгляд нет, т.к. если nys посылаться не будет, то процессу не ясно, ys не был послан ему потому что не должен быть послан, или потому что его сосед ещё не успел послать ys.

однако, т.к. мы должны ретранслировать данные <dtab,w,D>, всем соседям, которые скажут ys, то теоретически можно сохранить это сообщение (т.е. каждое) (которое кстати большое), на будущее и послать, если кто из соседей вдруг снова скажет <ys, w>. Однако это потребует много памяти.

Всё равно останется проблема завершения - не ясно, когда узлу можно будет осознать, что другие узлы тоже закончили, и можно удалить сохранённые сообщения, для этого придётся вводить алгоритм обнаружения завершения вычисления.

# Задача 31. (л5 стр 122, 123) Докажите, что приведенная ниже формула задает инвариант алгоритма Чанди-Мисры вычисления путей, ведущих в вершину v0.

∀u, w : ⟨mydist; v0; d⟩ ∈ Mwu => d(w; v0) <= d

^ ∀u : d(u; v0) <= Du[v0]

Из этого инварианта будет вытекать корректность алгоритма Чанди-Мисры (в предположении слабой справедливости, т.е. что посланное в канал сообщение рано или поздно - будет прочинато)

# Задача 32. (л5 стр 125) В описании алгоритма Чанди-Мисры не указывается, до каких пор должно проводиться вычисление маршрутов в каждом процессе. Докажите, в любой конфигурации любого выполнения алгоритма Чанди-Мисры промежуточные таблицы маршрутизации являются ациклическими. Что нужно добавить к алгоритму Чанди-Мисры, для того чтобы каждый процесс мог узнать, что построение таблиц маршрутизации в сети завершено.

Докажем ацикличность по индукции, - изначально таблицы маршрутизации ацикличны, значит, если однажды появится в таблицах цикл длинны <= 0, то это значит, чтобы было получено какое-то сообщение <mydist, v0, d> от w процессом v, и процесс v изменил свой Dv[v0] на d+dist(v,w) и Nbv[v0] на w.

Так как это ребро привело к появлению цикла, то он должен проходить через это ребро => это цикл (v, w, v1, v2, …, vk, v) (следует помнить, что каждый переход в цикле - это шаг, который сделает каждая из вершин в соответствии с той конфигурацией, которая сейчас в ней находится), выберем из цикла 2 цепочки (v, w) и (w, v1, v2, …, vk, v), существование второй из них означает, что когда-то были посланы сообщения в обратном порядке от v к w, и это привело к тому, что Dw[v0] = dist(w,v1) + dist(v1, v2) + … + dist (vk, v) + Dv[v0] ==> Dw[v0] >= Dv[v0]

но т.к. к нам пришло сообщение привнёсшее изменение в конфигурацию, а значит прошла проверка, что (d=Dw[v0]) + dist(v, w) < Dv[v0],

т.е. у нас противоречие, т.ч.к.

Можно проверить завершение алгоритма с использованием одного из алгоритмов завершения, либо доделать этот алгоритм.

По сути нам нужно сообщить конкретному процессу, когда сквозь наш граф будет таки построено дерево Nbv т.е. когда даже листья этого дерева построятся, для этого достаточно, чтобы каждый процесс, получив сообщение mydist, от соседа, ответил на это сообщение fin, либо сразу, если mydist не привел к перестройке конфигурации в этой вершине, либо если mydist изменил конфигурацию, и они были посланы другим соседям, то нужно послать fin-подтверждение на сообщение mydist, после того, как все соседи, которым был перепослан mydist тоже ответят fin.

После того, как узел получит fin от всех соседей - для него подсчёт конфигурации закончен.

Таким образом, листья Nbv получат от всех соседей fin, т.к. у соседей свои маршруты, и это пошлёт fin родителю, родитель уже сам получил fin от всех других соседей, которые не являются его дочерними и дождётся fin от своих дочерних, в конце концов процесс сойдётся у вершины v0.

По сути это и есть алгоритм обнаружения завершения.

# Задача 33. (л6 стр 14, 17, 73) Выясните, какие значения будут иметь все переменные в заключительной конфигурации алгоритма Netchange в том случае, когда этот алгоритм применяется к сети, имеющей следующую топологическую структуру ...:

A - B C

| | |

D - E - F

После того как была достигнута заключительная конфигурация, в сети возник новый канал связи между узлами A и F. Какое сообщение узел F отправит узлу A при обработке уведомления ⟨repair;A⟩? Какое послание узел A отправит узлу F в ответ на это сообщение?

нужно описать следующие структуры данных для каждого узла: Neighu, Du, Nbu[v], Du[w, v]

Neighu - очевидно

Du содержит кратчайшие расстояния до других узлов

Nbu[v] - следующая вершина в кратчайшем расстоянии - всё очевидно, кроме NbA[E, F, C] = B или D, причём могут быть разные (т.к. каждый раз срабатывает max - и у нас нету гарантии, что max - стабилен в выдаче результатов), аналогично в обратную сторону NbE[A] может равняться или B или D

ndis - содержит величину кратчайшего пути, заполняется очевидно

узел F отправит узлу A сообщение о каждом известном F узле v из V <mydist, v, Du[v]>, т.е. по сути расскажет, как можно достичь все другие узлы из него.

точно также поступит и узел А

Получая эти сообщения от другого узла, каждый узел обновит ndis и вызовет update (v), т.к. для узла A ndis[(F=w), (C=v)] =1 и путь к С стоит теперь 2 и лежит через F, то A уведомит об этом всех своих соседей, что через неё за 2 хопа можно добраться до С, уведомит - т.е. пошлёт всем соседям сообщение mydist, в частности пошлёт его и обратно в F (так уж следует из псевдокода алгоритма (хотя вообще говоря это избыточность))

# Задача 34. (л7 стр 149) Покажите, что в каждом вычислении древесного алгоритма в точности два процесса принимают решение.

Суть древесного алгоритма:

Каждая вершина, заметив, что она получила токен от всех соседей, кроме одного, пошлёт токен последнему.

Таким образом всё начнётся с листьев, и закончится тем, что сойдётся в какой-то внутренней вершине дерева.

Почему 2 процесса:

В процессе p произошло decide (т.к. это волновой алгоритм, то такой p существует), decide произошёл, когда p получил tok от u, и раз он принял decide, то до этого он находился в состоянии, когда он не получил ток только от одного из соседей, а значит он послал tok последнему соседу u.

Так как u послал tok p, то он послал ток последнему соседу, не приславшему токен, а значит получив посланный tok от p, он получил tok от всех своих соседей, а значит на u после этого произойдёт decide.

Вообще говоря - я не согласен с тем, что оба процесса примут решение, потому что это лишь в предположении, что процессы не проспят и последние 2-е успеют одновременно послать друг другу сообщение.

Так что это верно при синхронной передаче данных, но не при асинхронной.

# Задача 35. ??? (л7 стр 38) Привести пример сети, в которой произвольный волновой алгоритм произведет ровно N обменов сообщениями, где N число процессов в сети.

В лекции есть доказательство, почему всегда будет не меньше N сообщений, в случае если инициатор всего лишь 1.

Но не ясно, почему обменов не может быть больше, чем N, ведь алгоритм всегда можно “раздуть”.

Вероятно нужно придумать не общий вид сети, а какую-то утрированную сеть, например граф с 2-мя вершинами и одним ребром.

# Задача 36. ??? (л7 стр 43) Привести пример сети, в которой произвольный волновой алгоритм произведет ровно K обменов сообщениями, где K число каналов в сети.

В лекции есть доказательство, почему всегда будет не меньше |E| сообщений, если в сети процессы не могут друг друга идентифицировать, но не ясно, почему количество обменов может быть ограничено сверху, т.к. алгоритм всегда можно “раздуть”.

Возможно следует выбрать сеть, в которой K = 0, т.к. сеть из одной вершины и 0 рёбрами.

# Задача 37. (л7) Придумать кольцевой алгоритм с двумя инициаторами и доказать его корректность.

Допустим граф - цикл, или есть заложенный в каждый узел гамильтонов цикл в нашей сети.

Каждый инициатор запускает по одному его персональному токену в обоих направлениях цикла, если один инициатор дождётся токенов другого инициатора, то он принимает decision.

инициатор:

begin

send tok to Nextp;

send tok to Prevp;

receive tok from Nextp;

receive tok from Prevp;

decide;

end

не инициатор

begin

receive tok from одно направление цикла;

send tok to другое направление цикла;

end

Для доказательства корректности, нужно показать, что алгоритм когда-нибудь остановится (очевидно, ведь у нас нет ни одного цикла)

Что случится хотя бы один decision (доказывается от обратного - если нету decide, значит не было receive, значит не было send, … значит не было send в одном из инициаторов - противоречие)

Берём decide, если был decide, значит сработал receive, значит сработал send в другом процессе , … доводим логику до того, что в каждом процессе сработал send хоть раз.

# Задача 38. (л7) Придумать кольцевой алгоритм с двумя инициаторами и доказать его корректность.

deja-vu

# Задача 39. (л8 стр 25) В доказательстве фазового алгоритма доказывается, что отношение fpq(i) ≼ gpq(i) сохраняется даже если канал pq не является очередью. Показать, что данное отношение сохраняется даже если канал pq является каналом с потерей сообщений. Показать, что данное отношение не сохраняется если в канале pq сообщения могут дублироваться.

Если сообщения могут теряться, то доказательство не изменится, т.к. для каждого *l*-го принятого сообщения всё равно существует m*l* отправленное сообщение.

А вот в случае, если сообщения могут дублироваться - это не верно, т.к. очевидно, процесс p мог послать сообщение процессу q, это сообщение продублировалось, q приняло оба, т.е. приняло gpq(2), и только после этого p послало второе сообщение fpq(2).

Этап, который утрачивает силу в доказательстве со слайдов это “поэтому все m*l* различны”

# Задача 40. (л8 стр 11) Показать, что модифицированный фазовый алгоритм, который вырабатывает решение после принятия D - 1 сообщений от каждого соседа на входе, не является волновым алгоритмом.

Диаметр сети - это длинна максимального простого пути в графе.

Возьмём граф:

A <-> B <-> C

Диаметр графа D = 2

Допустим инициатор - B и допустим B послал по всем выходным каналам сообщение, но по каналу B->A сообщение застряло в канале и висит там, а B -> C сообщение прошло, тогда RecC = 1 = D-1

а значит будет принято decision, однако в A ещё не произошло ни одного события, т.е. это не волновой алгоритм.

# Задача 41. (л8) Предположим, что Вы намереваетесь использовать волновые алгоритмы в сетях с дублированием сообщений в каналах. Какие изменения необходимо внести в алгоритм Эха. Какие изменения необходимо внести в алгоритм Финна.

*Это печальное решение для эха:*

В задаче 39 показано, почему алгоритм Эха падает, если в канале может быть дублирование сообщения, для борьбы с этим, нужно уметь идентифицировать сообщения, т.е. присваивать им некоторые номера, т.е. пусть процесс при посылке tok, будет приписывать порядковый номер tok, который он посылает (в силу того, что алгоритм эха никогда не посылает больше D сообщений по одному каналу, то максимальный порядковый номер будет = D), причём если мы хотим, чтобы канал мог быть не очередью, то каждому процессу придётся завести массив <каждый входной канал>x<было ли получено i-е сообщение>, в котором отмечать true и false (битовое хранение данных можно реализовать вычислительно в 8 раз оптимальнее, чем байтовое).

Лучше так: приказать каждому процессу запонить, от кого он уже однажды получал токен (а не просто плюсовать переменную, как это сделано на алгоритме в слайдах)

Чтобы алгоритм Финна работал при возможности дублирования сообщений не нужно делать Ничего.

Он и так будет работать, т.к. получение дублированного сообщения повторно любым из процессов, не приведёт ни к каким значимым действиям внутри этого процесса (множества Inc и Ninc не изменятся, новых сообщений не породится, decide не произойдёт)

# Задача 42. (л8 стр 129) Привести вычисление алгоритма Тарри, в результате которого получается дерево, не являющееся деревом поиска в глубину.

В лекции 8 на стр 129-134 как раз дан пример дерева, не являющегося деревом поиска в глубину, однако в этом примере, дерево **не** поиска в глубину - никак не может получится из вычисления алгоритмом Тарри.

В целом, если рассмотреть полный граф (в котором все вершины соединены) - то чтобы ни построил алгоритм Тарри, оно всегда не будет являться деревом поиска в глубину.

Но Захаров скорее всего захочет примера, когда при разных комбинациях работы Тарри - алгоритм может выдать и то и другое.

Для этого следует рассмотреть тот самый граф из лекций, но с дополнительным ребром, соединяющим вершину в “нижнем предлевом уголу” с “нижней средней вершиной”.

# Задача 43. (л7 стр 121) Адаптируйте алгоритм эха для вычисления суммы входных данных всех процессов.

Алгоритм эха работает так:

инициатор один

получив токен первый раз процесс посылает по токену всем соседям, кроме родителя, от которого он получил первый токен, когда он получит по токену от всех детишек, то пошлёт токен и родителю.

Где-то писалось, что алгоритм эха можно развить до построения остовного дерева (*только никак не вспомню где*) (каждый процесс итак знает своего родителя), необходимо при посылке последнего токена родителю, посылать <mes, true если это сообщение шлётся родителю, sum>, каждый процесс складывает sum от всех полученных токенов, в которых фигурирует true, добавляет в суму своё личное значение и отправляет это наверх с меткой true.

Строго-говоря нужно доказать корректность, для этого нужно показать, что каждая сумма будет 1) хотя бы один раз (показать, что каждый процесс хотя бы раз пошлёт свою sum родителю и сказать, что эту sum родитель не продолбает, т.к. метка true) и 2) не более чем один раз учтена (доказывать от обратного).

# Задача 44. (л8 стр 44) Доказать теорему: Используя фазовый алгоритм избрание лидера может быть произведено за O(D|E|) обменов сообщениями и используя O(D) единиц времени.

O(D|E|) следует из того, что алгоритм посылает по каждому каналу не более, чем D штук сообщений (благодаря if … && Sentp < D begin … Sentp += 1 ... end)

Ну а рёбер всего |E| штук (это в случае односторонних каналов, если каналы неодносторонние, то они используются в обе стороны, и потому 2\*|E| штук)

***Скорее всего это неверно:***

При подсчёте единиц времени следует помнить, что передача сообщения всегда длится 1 ед. (потому что иначе можно собрать сеть из последовательно соединённых в цепочку (из односторонних каналов) процессов, и там можно распределить задержки так, чтобы сначала пришлось ждать общения первого крайнего процесса с другим краем, потом с предпоследним с того края, …, и тогда сложность будет похожа на O(D|E|))

исходя из того, что диаметр сети = D, следует, что длинна максимального простого цикла (двусторонний канал считаем за 2 односторонних) = O(D)

Необходимо показать, что пока первый посланный токен процессом-инициатором шёл до самого дальнего процесса в течение времени D, токены до всех других процессов уже успели добраться (потому что диаметр сети D - т.е. длиннее путей нету), а значит каждый из этих процессов уже послал токены которые начали расходиться по графу, в частности идти по пути в обратную сторону - в направлении обратно к инициатору, и когда токен из самого длинного пути туда-сюда вернётся, другие пути туда-сюда уже будут обойдены токенами и тоже вернутся, а значит у инициатора Sentp станет = D, а значит случится decide всего лишь за время туда-сюда, т.е. за O(D)

# Задача 45. (л9 стр 13) Докажите, что алгоритм избрания лидера путем сравнения (отыскания экстремумов) является волновым алгоритмом, если событие избрания процесса лидером рассматривать как событие решения.

Для доказательства того, что алгоритм является волновым, необходимо:

1. Показать, что каждое вычисление конечно
2. Каждое вычисление содержит хотя бы один *decide*
3. Каждому *decide* предшествует хотя бы одно вычисление в каждом из процессов.

Алгоритм избрания лидера, это алгоритм который

1. в каждом процессе одинаковый алгоритм
2. децентрализованный, инициаторы - любые
3. алгоритм достигает заключительной конфигурации (отсюда следует верность 1-го пункта волнового алгоритма) и в каждой существует ровно один процесс, который либо в состоянии lost либо leader (такой только один и ровно один, т.е. верно 3-е) (~~изначально все находятся в состоянии sleep, а значит в каждом процессе выполнилось хотя бы одно событие (правда не факт, что оно выполнилось до события decide, т.е. присовеия лидеру значения leader)~~)

Однако, т.к. согласно условию алгоритм избрания лидера работает на условии отыскания экстремумов, а потому это значит, что каждый процесс должен был передать своё значение до decide, т.е. верно 2-е

# Задача 46. (л9 стр 38) Доказать теорему: Алгоритм Ченя-Робертса решает задачу избрания лидера, используя менее (N2) сообщений и O(N) единиц времени.

***Док-во полностью описано в лекциях. Итак:***

Обозначим p0 инициатора с наименьшим именем.

Всякий другой процесс может быть либо не-инициатором, либо инициатором с отличительным признаком, превышающим p0 , и поэтому все процессы передадут далее маркер ⟨tok,p0⟩ выпущенный p0. Значит, p0 получит свой маркер обратно и будет избран лидером.

Не-инициаторы не будут избраны, но все они перейдут в состояние *lost* самое позднее к моменту передачи маркера, который выпустил p0 .

Инициатор p , для которого p > p0 , не станет лидером, т.к. p0 не передаст далее маркер ⟨tok, p⟩, 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 ⟨tok,p0⟩. Это и служит обоснованием того, что наш алгоритм решает задачу о выборах.

В алгоритме задействовано не более N различных маркеров, и каждый маркер передается не более N раз; этим и обосновывается оценка O(N2) сложности по числу обменов сообщениями.

Чтобы убедиться в том, что может иногда может понадобиться передать Ω(N2) 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: 1/2N(N+1).

# Задача 47. (л9 стр 46) 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 Ченя-Робертса – O(N\*logN) – в среднем и O(N2) – в худшем, если все процессы инициаторы, то в среднем ≈ 0.69\*N\*log N

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.

Минимум по количеству сообщений будет в том случае если на картинке в л9 стр 43 (там просто расположены процессы по возрастанию **по часовой стрелке**) сообщения будут ходить в кольце **против часовой стрелки** (тогда сообщения каждого из не минимальных процессов пройдёт лишь 1 шаг (т.е. это N-1 передача сообщения)), а сообщение лидера пройдёт полный круг - это N передач, т.е. всего будет 2N-1 посылка сообщений

# Задача 48. (л9) Привести начальную конфигурацию сети, в которой алгоритм Патерсона-Долева-Клейва-Роде требует (⌊logN⌋ +1) раундов. Также привести начальную конфигурацию, в которой алгоритму достаточно двух раундов вне зависимости от числа инициаторов. Возможно ли алгоритму завершиться за один раунд?

заметка - логарифм имеется ввиду двоичный

1. пример из л9 стр 57 не подходит, т.к. там 3 тура, а нам надо log8+1 = 4

подойдёт кольцо 1 4 2 5 3 6 -> после первого тура 1 2 3 -> после второго тура 1 -> 3-й тур установит лидера 1

(есть немного теории на л9 стр 84)

1. подойдёт кольцо 1 2 3 4 5 6 7 … n -> после первого тура останется только 1 -> второй тур установит процесс 1 лидером
2. Только если кольцо состоит из одного элемента, в любой другой конфигурации у нас найдётся в первом туре m-1-n, где m и n (возможно m==n) - больше 1, и понадобится второй тур, чтобы установить, что 1 - лидер

# Задача 49. Сравнить алгоритм угасания волны для колец с алгоритмом Ченя-Робертса. В чем различия и какой они имеют эффект?

кольцевой волновой описан в л7 стр 65, Ченя-Робертса в л9 стр 36

кольцевой волновой угасания волны - означает, что инициатор пошлёт токен, и если по пути встретится другой процесс, у которого номер меньше, то он этот токен прервёт, а сам пошлёт новый токен - новую волну, которая должна будет обойти весь круг

алгоритм Ченя-Робертса идеологически точно такой же, но есть нюанс

согласно определению Ченя-Роберста - каждый процесс, увидев токен от меньшего процесса сразу присваивает себе lost,

в то время как для алгоритма угасания волны идёт вычисление лидера, после чего идёт этап рассылки всем звеньям информации о том, что лидер был успешно выбран.

# Задача 50. (л10 стр 58) Установить для сообщений всех семи типов алгоритм Галладжера-Хамблета-Спиры, может ли каждое из них быть послано процессу, находящемуся в состоянии сна.

connect - да может

initiate - нет, т.к. всегда шлётся в ответ на connect (стр 69), т.е. тот узел уже проснулся, или собратьям по фрагменту (а значит они уже успели прочухаться и объединиться с кем-то) (стр 75)

test - да - стр 81 - test может быть послан другому узлу, чтобы узнать можно ли в будущем слиться, по оптимальному ребру для этого фрагмента, цель - запрос узал другого фрагмент о вожможности дальнейшего слияния - другой фрагмент может быть спящим узлом

reject - нет, т.к. шлётся на стр 85 в ответ на test, посланный другой вершиной, т.к. выяснилось, что ребро - внутрифрагментное

accept - нет, т.к. шлётся в ответ на test на стр 87

report - нет, т.к. на стр 91 шлётся в отца, но если есть отец, то это значит, что уже проходило много сообщений внутри фрагмента

changeroot - нет

# Задача 51. ??? (л10 стр 107) Алгоритм Этья (Attiya) отличается от алгоритма Кораха-Каттена-Морана только тем, что фишка вместо того, чтобы впасть в состояние погони, замирает и посылает сообщение-убийцу, которое догоняет преследуемую фишку, убивает ее и возвращается назад. После возвращения убийцы выполнение обхода продолжается. Реализовать алгоритм Attiya и установить его сложность.

Всё, что есть в книге Теля:

Чтобы охотник мог вернуться в узел, требуется двунаправленность сети. Если использовать произвольный алгоритм f(х)-обхода, то в результате будет построен алгоритм избрания лидера, имеющий сложность по числу обменов сообщениями, приблизительно равную величине 3 \* sum (i=1,N ; f(N/i))

# Задача 52. (л11 стр 23) Временная сложность алгоритма завершения вычисления называется число единиц времени между завершением базового алгоритма и завершением контрольного алгоритма. Какова временная сложность алгоритма Дейкстры-Шолтена?

алгоритм Дейкстры-Шолтена поддерживает дерево, в которое входят все активные узлы (рассчитано на базовый алгоритм с одним инициатором, и в нём же сидит и инициатор Дейкстры-Шолтена), при этом по мере того как будет в базовом алгоритме происходить переключение узлов на passive, дерево будет сворачиваться.

Допусти граф - цепь. Допустим алгоритм запускается в одном конце и просто передаёт одно сообщение в другой конец и на этом всё. Тогда алгоритм Дейкстры построит дерево на всю цепь длинной в N вершин, причём когда базовое сообщение дойдёт до края, то дерево базовый алгоритм завершится, а дерево начнёт схлопываться, и на это потратится N времени.

сложность больше, чем N - быть не может, потому что когда базовый алгоритм завершился, то все процессы находятся в состоянии passive и лишь отправляются сигнальные сообщения sig от листьев дерева, которые схлопывают.

(вообще-то это не строгое доказательство, надо построже)

отсюда сложность = N

# Задача 53. ??? Если алгоритм Шави-Франчеза применяется к произвольной сети, процессы которой имеют уникальные идентификаторы, и при этом используется волновой алгоритм Галледжера-Хамблета-Спиры, то временная сложность такого контрольного алгоритма будет составлять (NlogN). Можно ли улучшить временную сложность до O(N) ценой обмена O(N) дополнительными сообщениями?

# Задача 54. ??? (л11 стр 53) Показать существование такого базисного вычисления, в котором происходит обмен m сообщениями, при контроле которого алгоритм Сафры использует m(N - 1) сообщений.

# Задача 55. (л11 стр 58) В алгоритме Раны предполагается, что процессы наделены отличительными признаками. Предположим теперь, что все процессы анонимны, но обладают возможностью отправлять сообщения своим последователям по кольцу, и при этом число процессов заранее известно. Внесите в алгоритм Раны необходимые изменения, позволяющие ему работать в рамках таких допущений.

Как работает Рана:

Рана основан на счётчиках (Лемпорта, например), каждый процесс инициирует волну, и приписывает этой волне величину своего счётчика, каждый процесс помнит, когда он последний раз был активным, если процессу придёт волна, которая была послана раньше, чем этот процесс стал quiet, то он забивает на эту волну и она глушится.

Для того, чтобы проверить свойство quite - ведётся подсчёт неподтверждённых сообщений (т.е. на каждый message базового алгоритма должен в ответ прийти ack), если количество = 0, и процесс passive, то quiet выполнен. (эти же сообщения и ack передают часики Лемпорта)

В функции I (л11 стр 112) каждый процесс, который перешёл в состояние quiet шлёт tok (т.е. запускает волну) процессу Nextp - т.е. следующему процессу по кольцу, причём волна запоминает в каком процессе (его отличительный признак) она началась, и если токен туда вернётся, то процесс сделает decide (Announce).

Для того, чтобы алгоритм Раны работал без отличительных признаков, необходимо, чтобы вместе с токеном и его временем (когда токен был послан), в токен запиливалось число 0, которое по мере перемещения по кольцу на каждой итерации увеличивалось, если это число в каком-то процессе станет = N, значит этот токен прошёл по всем N процессам => волну надо завершилать, нужно делать decide.

# Задача 56. (л11 стр 58) Обоснуйте корректность алгоритма Раны на основе инвариантов этого алгоритма.

тут следует глянуть подробное доказательство хранящееся в разделе “задачи прошлых лет на гугло-драйве”, оно очень подробное, его можно ужать

# Задача 57. (л11 стр 58) Внесите изменения в алгоритм Раны так, чтобы для передачи сообщений можно было использовать произвольный волновой алгоритм, а не только кольцевой алгоритм.

см. первые 2 параграфа из задачи 55, там про то, как работает Рана в общем, и как реализован Рана на слайдах.

Чтобы можно было пользовать произвольный волновой алгоритм, нужно в Ip посылать токен не Next, а туда, куда скажет волновой алгоритм (не забываем передавать часики Лемпорта), а в процедуре T, вместо сравнения p=q, нужно спросить у волнового алгоритма “decide?”

Захарову наверняка лучше написать строгий алгоритм, как в слайдах.

# Задача 58. (л12) Предположим, что снятие снимка процессом p является дополнительным внутренним событием ap. Показать, что S\* значимый срез ⇔ ∀p, q ap not≼ aq & aq not≼ ap:

Согласно теореме (л12 стр 21)

значимость S\* ⇔ сечение L, порождённое S\* согласовано, т.е.

если оно замкнуто влево относительно причинно-следственной зависимости. ⇔

⇔ S\* осуществимо, т.е. rsvd подмножество sent

(=>) следует из того, что т.к. ap и aq являются внутренними действиями (согласно условию), то если бы ap предшествовало aq, или наоборот, то согласно определению причинно-следственной связи нашлась бы цепочка из событий (происходящих последовательно в каком-то процессе, или связанных причинно-следственной связью за счёт посылки-отправки сообщения), но это значило бы, что после a в одном из p или q совершилось событие посылки сообщения, т.е. посылка сообщения не была зафиксирована в конфигурации (т.к. она случилась после фиксации), а приём сообщения был зафиксирован (т.к. он является причинным предшественником другой операции ax), а это противоречит тому, что S\* осуществимо.

(<=) допустим второе выполнено, а S\* - не значимый срез, тогда S\* не осуществимо, тогда найдутся pq, для которых сообщение было принято ap, но не послано aq, ap следует за aq, за ap было произведено внутреннее действие api, фиксирующее моментальное состояние, и до aq было произведено действие не фиксирующее aqi, т.е.

ap < api, aqi < aq, однако согласно предположению из задачи api и aqi не находятся ни в каком отношении порядка, а следовательно и ap и aq не могут, однако ap и aq - это получение и отправка сообщения, а значит они по определению упорядочены

противоречие, ура S\* - значимый срез

# Задача 59. (л2) Будем рассматривать регистрацию моментального локального состояния процесса p как еще одно внутреннее событие ap. Докажите, что S\* является значимым ⇔ ∀p, q : ap ∥ aq

это вроде как именно то, что доказывается в задаче 58, т.к. согласно определению в л2 стр 97, параллельность это невыполнение частичного порядка между a и b

т.е. доказательство такое же

# Задача 60. ??? Профессор Захаров утверждает:

После того как я прочел лекцию о моментальных состояниях, я сумел лучше понять алгоритмы обнаружения завершения вычислений. Например, в алгоритме Сафры обработку маркеров процессом p нужно понимать как вычисление моментального состояния процесса p. В построенном моментальном состоянии системы все процессы являются пассивными, так как маркер обрабатывается только пассивными процессами. Поэтому, чтобы принять решение о завершении вычисления требуется всего лишь проверить, пусты ли все каналы. Для этого в маркере указывается суммарное значение счетчиков сообщений. Однако мне неясна та роль, которую играют окраски white и black, а также как удается обеспечить значимость моментальных состояний системы. Не могли бы Вы помочь профессору?