УДК 372.4+004.7

Плаксин М.А.

Национальный исследовательский университет «Высшая школа экономики» (Пермский филиал), г. Пермь, Россия

О НЕКОТОРЫХ МЕТОДИЧЕСКИХ СРЕДСТВАХ ПРОПЕДЕВТИКИ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ В ШКОЛЬНОЙ ИНФОРМАТИКЕ*

АННОТАЦИЯ

Статья посвящена вопросу о включении в школьный курс информатики темы «параллельные вычисления». Описаны некоторые методические материалы, подготовленные в ходе работ над «пермской версией» пропедевтического курса информатики (авторский коллектив М.А. Плаксин, Н.И. Иванова, О.Л. Русакова).

КЛЮЧЕВЫЕ СЛОВА

Информатика; параллельное программирование; параллельные вычисления; параллельные алгоритмы; начальная школа; средняя школа; методика; пропедевтика; ТРИЗформатика, ТРИЗформашка; синхронизация потоков; семафоры; игра-ходилка; ярусно-параллельная форма.

Mikhail Plaksin

National Research University Higher School of Economics (Perm branch), Perm, Russia

SOME METHODICAL TOOLS FOR PROPAEDEUTICS PARALLEL COMPUTING IN SCHOOL INFORMATICS

ABSTRACT

The article is devoted to inclusion of the topic "parallel computing" in the school informatics . Some methodical materials prepared in the course of work on the "Permian version" of a propaedeutic course of computer science (the author team is M.A. Plaksin, N.I. Ivanova, O.L. Rusakova) are described.

KEYWORDS

Informatics; parallel programming; parallel computing; parallel algorithms; elementary school; secondary school; teaching methods; propaedeutics; TRIZformatika, TRIZformashka; thread synchronization; semaphores; board game; stacked-parallel form, multilevel structure.

Современный этап развития computer science связан с массовым распространением параллелизма вычислений на всех уровнях (многомашинные кластеры, многопроцессорные ЭВМ, многоядерные процессоры). Это делает актуальным включение пропедевтики параллельного программирования в школьный курс информатики. Год назад, выступая на X Международной научно-практической конференции «Современные информационные технологии и ИТ-образование» [7], автор данной статьи назвал некоторые методические и психологические проблемы, связанные с этим процессом, предложил перечень вопросов для начального знакомства с параллельной тематикой в школьной информатике и перечислил результаты, достигнутые в данном направлении в ходе работ над «пермской версией» пропедевтического курса информатики [5, 6] (учебники «пермской версии» печатаются издательством «БИНОМ. Лаборатория знаний»; авторский коллектив: М.А. Плаксин, Н.И. Иванова, О.Л. Русакова; рабочее название курса – «ТРИЗформатика»). Настоящая статья детализирует некоторые из результатов, названных в 2015 г. и полученных за прошедший год.

* **Труды XI Международной научно-практической конференции «Современные информационные технологии и ИТ-образование» (SITITO'2016), Москва, Россия, 25-26 ноября, 2016**

Игра-«ходилка» для изучения механизмов синхронизации потоков/процессов

В рамках «пермской версии» пропедевтического курса информатики для освоения представления алгоритмов в виде блок-схемы используются «игры-ходилки». Смысл такой игры в том, чтобы провести свою фишку по некоторому заранее заготовленному маршруту, состоящему из цепочки упорядоченных позиций. Продвижение фишки по игровому полю определяется броском игральной кости. В «пермских» играх-ходилках маршрут на игровом поле представляет собой блок-схему. Учащийся должен пройти по маршруту, соблюдая правила выполнения блок-схем (повторение циклов, развилки, выполнение подпрограмм, ввод данных). В ходе игры игрок многократно проходит по различным алгоритмическим конструкциям. Это должно обеспечить закрепление знаний. А поскольку в игре участвует сразу весь класс (разбитый на группы в 3-4 человека), получается, что в освоении материала задействованы все сто процентов учащихся.
Такой подход дает возможность увеличить объем материала, изучаемого в теме «Алгоритмы». Кроме традиционных для школы алгоритмических конструкций – развилки, циклы с предусловием, циклы с заданным числом повторений – учащиеся знакомятся с командами ввода данных, с понятиями «подпрограмма» («вспомогательный алгоритм») и «параметр». Понятие «способ передачи параметров» при этом не обсуждается. В самих играх-ходилках используются только входные параметры целые числа.

Опыт использования игр-ходилок для изучения блок-схем навел на мысль применить этот же механизм для освоения механизмов параллельного программирования, а именно, механизмов синхронизации потоков/процессов. Поскольку традиционные блок-схемы описывают только последовательные алгоритмы, для параллельных пришлось расширить их новыми блоками.

Общее число механизмов синхронизации, используемых в современных вычислительных системах весьма велико: блокируемые команды, критические секции, семафоры, мьютексы, считающие семафоры, события (событие «конец потока/процесса» и события, управляемые вручную), механизм рандеву в языке Ада. Рассмотреть их все в начальном курсе невозможно, да и не нужно. Для включения в игру (и в пропедевтический курс) были отобраны семафоры и события. Семафоры – только бинарные. Для них оставлены только две операции: закрыть и открыть. С каждым семафором связана «критическая секция». Но не как отдельный объект диспетчеризации, а как фрагмент алгоритма, доступ в который регулируется семафором. На входе в каждую критическую секцию образуется очередь. Понятие «событие» было трансформировано в понятие «сигнал», с которым связаны три операции: включить, выключить и ждать. Смена термина («сигнал» вместо «события») связана с тем, что в программировании событие может находиться в одном из двух состояний («совершилось» / «не совершилось») и может переходить из одного состояния в другое. Но в обыденном восприятии событие, уже совершившееся, вернуться обратно в состояние в состояние «не совершилось» уже не может. Поэтому вместо совершившихся и не совершившихся событий были введены сигналы, которые могут быть включены и выключены. Переход сигнала из состояния включен в состояние выключен и обратно трудностей восприятия не вызывает.

Правила игры в «ходилку-семафорку» выглядят так.

«Блок-схема-Семафорка» – игра-«ходилка». Цель – переместить фишку по шагам от начала до конца блок-схемы. Количество шагов на каждом ходе определяется бросанием кубика. Развилки определяется правилами записи блок-схем: условие (ветвление и цикл ПОКА), цикл ПОВТОРИТЬ.

Фишки в данной игре называются «потоками».

Блоки перенумерованы. Номера играют чисто информационную роль, служат «именами» блоков и предназначены для однозначного определения позиции.

Играть может любое количество игроков. Разумное число – от 2 до 4.

Каждый игрок двигает свой «поток», отличный от других.

В отличие от традиционных игр-ходилок в данной игре два блока «Начало». Половина игроков начинает с одного блока. Вторая половина – с другого.

Первоначально все «потоки» ставятся на блоки «Начало».

Игроки по очереди бросают кубик и передвигают свой «поток» по блокам на выпавшее количество шагов.

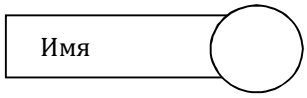
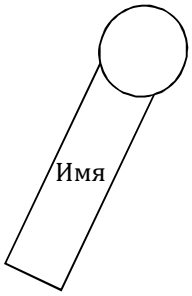
Если на пути встречается заголовок цикла ПОВТОРИТЬ (блок-шестиугольник), то фишка должна зайти в тело цикла и повторить цикл указанное число раз. Игрок должен отслеживать, сколько повторов цикла он уже выполнил.

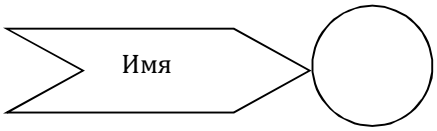
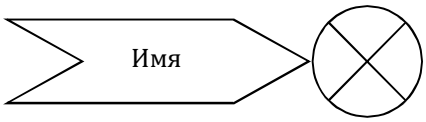
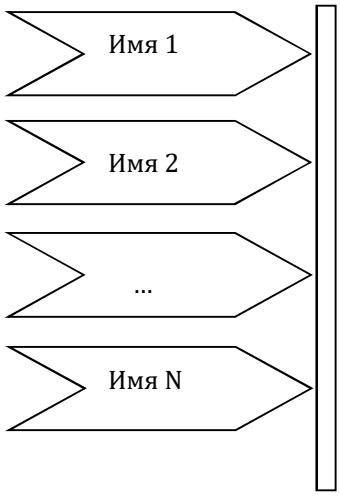
Если на пути встречается блок условия (блок-ромб), то число, выпавшее на кубике (количество шагов на данном ходе), подставляется в записанное в ромбе условие. Результат сравнения определяет направление дальнейшего движения. Если фишка попала на блок условия

на последнем шаге данного хода, то направление движения будет определяться следующим броском кубика (на следующем ходе).

Если на пути встречается блок вызова подпрограммы (блок-прямоугольник с двойными стенками), то игрок должен увести свой «поток» на блок-схему названной подпрограммы, пройти ее от начала до конца и после этого продолжить выполнение с блока, следующего за блоком вызова подпрограммы.

Кроме этих традиционных блоков на игровом поле присутствуют блоки нетрадиционные: блоки закрытия и открытия семафоров, обозначение критической секции, блоки включении, выключения и ожидания сигналов. Их обозначение, название и смысл следующие:

Изображение блока	Название блока	Смысл блока
	Закрыть семафор с указанным именем	Если семафор находился в состоянии «открыт», он переходит в состояние «закрыт». «Поток», закрывший семафор, может продолжать движение по игровому полю. Если семафор находился в состоянии «закрыт», то «поток», попытавшийся закрыть семафор, прекращает движение по игровому полю и встает в очередь «потоков», ожидающих открытия этого семафора.
	Открыть семафор с указанным именем	Если семафор находился в состоянии «открыт», он остается в состоянии «открыт». Если семафор находился в состоянии «закрыт», он переходит в состояние «открыт». Если с этим семафором связана очередь «потоков», то первому «потоку» из этой очереди разрешается войти в критическую секцию.
	Критическая секция	Критическая секция обводится пунктирной линией для наглядности. Никаких действий линия не вызывает. Но! Внутри критической секции в каждый момент может находиться только один поток! Первым блоком внутри критической секции должно быть закрытие семафора, последним – открытие этого же семафора. В начале секции отводится место для очереди «потоков», ожидающих вхождения в секцию. Вход в критическую секцию производится только через эту очередь. Если очередь пуста, то поток проходит через нее, не задерживаясь, и сразу идет на первый блок секции. Однако, если в очереди уже находится хотя бы один поток, то вновь прибывший поток встает в конец очереди и сможет войти в критическую секцию только после всех потоков, стоящих в очереди перед ним.

Изображение блока	Название блока	Смысл блока
	Включить сигнал с указанным именем	Если сигнал находился в состоянии «включен», он остается в состоянии «включен». Если сигнал находился в состоянии «выключен», он переходит в состояние «включен». «Поток», включивший сигнал, может продолжать движение по игровому полю.
	Выключить сигнал с указанным именем	Если сигнал находился в состоянии «выключен», он остается в состоянии «выключен». Если сигнал находился в состоянии «включен», он переходит в состояние «выключен». «Поток», выключивший сигнал, может продолжать движение по игровому полю.
	Ждать сигналов с указанными именами	Количество сигналов может быть любым. Если все перечисленные сигналы находятся в состоянии «включен», «поток» может продолжать движение по игровому полю. Если хотя бы один из сигналов находится в состоянии «выключен», «поток» прекращает движение по игровому полю. Возобновление движения «потока» на очередном ходе будет возможно только в том случае, если в этот момент все перечисленные сигналы находятся в состоянии «включен».

Для отслеживания состояния семафоров и сигналов выбирается специальный игрок. По ходу игры он заполняет следующую таблицу. Запись «Закр» напротив имени семафора означает, что семафор закрыт, запись «Откр» – что семафор открыт. Знак «–» напротив имени сигнала означает, что сигнал выключен, знак «+» – что сигнал включен. При начале игры все сигналы выключены, все семафоры открыты. Поэтому первоначально таблица имеет такой вид:

Сигнал S1	–										
Сигнал S2	–										
Сигнал S3	–										
Семафор R1	Откр										
Семафор R2	Откр										

Игровое поле делится на два листа: лист программы и лист подпрограмм. Эти листы представлены на рис.1, 2.

Игра «Блок-схема-Семафорка» была апробирована на занятиях с учащимися начальной школы лицея №10 г.Перми. Фото с урока см. на рис.3.

Задачи на построение, анализ и оптимизацию ярусно-параллельных форм

Переход к параллельному программированию вызвал интерес к новому свойству алгоритма

– способности к распараллеливанию. Возникла потребность в представлении алгоритма в виде, пригодном для совместного выполнения несколькими исполнителями (вычислительными ядрами). Одним из таких представлений является граф, именуемый ярусно-параллельной формой (ЯПФ) [2, 3].

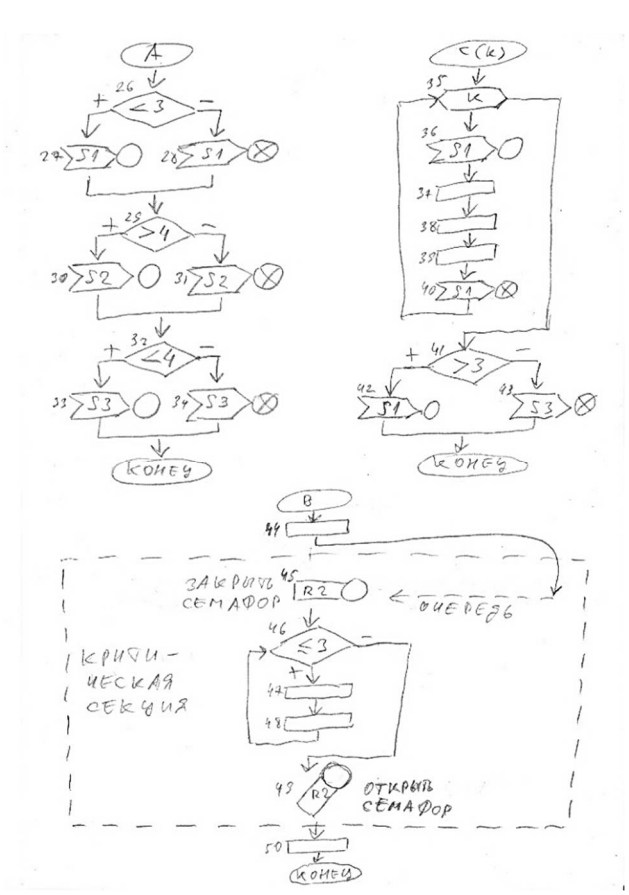


Рис. 1. Игровое поле игры «Блок-схема-Семафорка». Главная программа

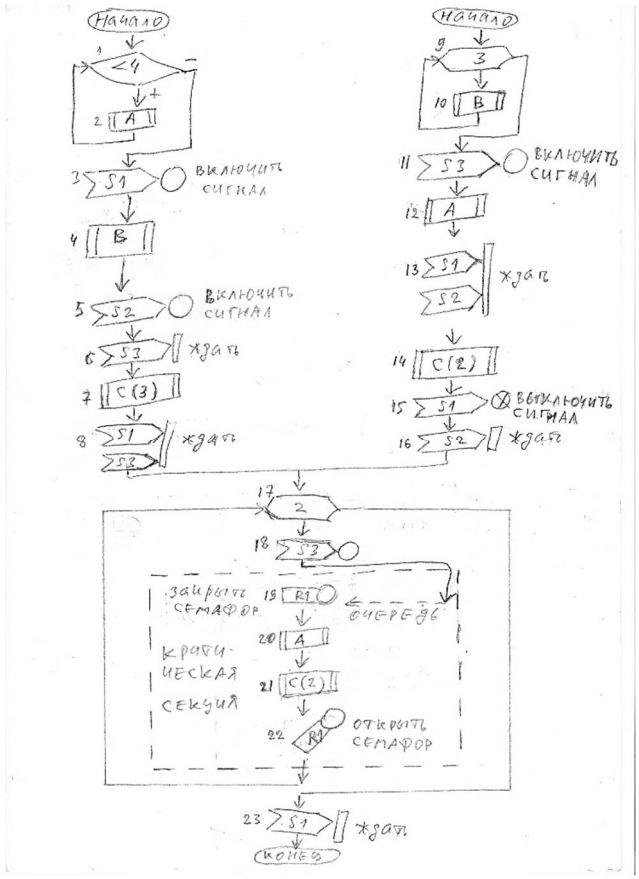


Рис. 2. Игровое поле игры «Блок-схема-Семафорка». Подпрограммы

Рис. 3. Фото с урока в лицее №10 г.Перми

Для построения ЯПФ алгоритм должен быть представлен в виде «информационного графа»:

в виде последовательности некоторых «вычислительных блоков» («зерен», «гранул» [1]), связи между которыми обозначают передачу информации. Дуга, направленная из вершины A в вершину B, означает, что в процессе выполнения блока A будет подготовлена некоторая информация, используемая при выполнении блока B (вычислено значение некоторой переменной, применяемое при вычислениях в блоке B). Всегда существует некоторое множество «начальных блоков», не имеющих входных дуг. Такие блоки называются «входными вершинами» и располагаются на первом ярусе ЯПФ. На втором ярусе располагаются блоки, зависящие только от вершин первого яруса. На третьем – блоки, зависящие от вершин первого и второго ярусов и так далее. На каждом ярусе располагаются вершины, зависящие от вершин, расположенных на предыдущих ярусах. Вершины последнего яруса, не имеющие выходных дуг, называются «выходными вершинами». Количество вершин, расположенных на одном ярусе называют шириной яруса. Максимальную ширину ярусов называют шириной ЯПФ, количество ярусов – высотой ЯПФ.

Поскольку между вершинами одного яруса информационные связи отсутствуют, все вычислительные блоки одного яруса могут быть выполнены одновременно. То есть ширина ЯПФ определяет возможную степень распараллеливания алгоритма. Если все вычислительные блоки требуют для своего выполнения примерно одинакового времени, то время выполнения алгоритма можно охарактеризовать высотой ЯПФ – количеством составляющих ее ярусов.

Вообще говоря, для одного алгоритма можно построить несколько ярусно-параллельных форм. Для этого достаточно перемещать вершины с одного яруса на другой. Возможность перемещения ограничена входными дугами (дуга в ЯПФ обязательно идет сверху вниз). Для вершин, расположенных на самом последнем ярусе, ограничения на смещение вниз нет. Достаточно добавить в граф новый ярус. Перемещение вершин с более высоких ярусов может потребовать смещения связанных с ними блоков, расположенных ниже.

Поскольку ширина ЯПФ задает допустимую степень распараллеливания алгоритма, а высота – время его выполнения, перестраивая ЯПФ, мы можем менять потребность в вычислительных ядрах и скорость выполнения.

Выше, при описании построении ЯПФ, мы руководствовались правилом:

Каждый вычислительный блок размещается на ярусе, непосредственно следующем за ярусом, на котором закончено вычисление нужных для него данных. Расположение вершины на ярусе N означает, что последние необходимые для ее выполнения данные были вычислены на ярусе (N-1).

Легко заметить, что построенная таким образом ЯПФ будет иметь минимальную высоту и максимальную ширину. То есть она будет задавать параллельный алгоритм, который будет выполняться максимально быстро, но потребует максимального количества вычислительных ядер.

При этом сразу же возникает ряд вопросов, связанных с оптимизацией ЯПФ.

Можно трансформировать ЯПФ таким образом, чтобы сохранить ее высоту, но уменьшить ширину? Иными словами, можно ли ускорить выполнение алгоритма, не увеличивая потребности в вычислительных ресурсах?

Какое максимальное количество вычислительных ядер имеет смысл задействовать для выполнения данного алгоритма?

Какую минимальную высоту может иметь ЯПФ, если ширина ее ограничена некоторым заранее заданным числом? Как построить такую ярусно-параллельную форму? Последний вопрос интересен для компилятора, который должен сгенерировать код программы, рассчитанный на выполнение на конкретной вычислительной системе, имеющей определенное число ядер.

В «Пермской версии» материалы для освоения понятия ЯПФ впервые появились в 2013 г. в конкурсе «ТРИЗформашка-2014».

(Конкурс «ТРИЗформашка» [4] – ежегодный межрегиональный Интернет-конкурс по информатике, системному анализу и ТРИЗ (теории решения изобретательских задач) для школьников и студентов. В марте 2017 г. конкурс состоится в 17-й раз. Возраст участников – от первого класса до четвертого курса. Среднее количество команд – около 100 (рекордное – 202). География конкурса – от Владивостока до Риги. Сайт конкурса www.trizformashka.ru. В рамках работ над курсом ТРИЗформатки конкурс «ТРИЗформашка» выступает эффективной экспериментальной площадкой).

Согласно подходу, декларированному в [7], материал должен быть представлен на примерах, не связанных с работой ЭВМ. Исполнители должны манипулировать материальными объектами. Конкретно, в конкурсе «ТРИЗформашка-2014» соответствующая задача выглядела следующим образом.

Дана схема «дворца», который должен быть построен из каменных глыб, взгроможденных друг на друга (см. рис.4). В роли строителей дворца выступали джины. Руководить строительство

вызывались три царских сына. Причем каждый из них предлагал свой критерий успешности проекта.

– О, Отец! – воскликнул старший сын. – Доверьте сооружение Дворца мне. Я призову великое множество джинов. И они построят дворец за неделю!

– О, Отец! – отозвался средний сын. – Зачем нам такое количество джинов. Ведь Дворец нужен нам ко дню приезда Принцессы. А это произойдет только через две недели. Доверьте стройку мне. Я призову ровно столько джинов, чтобы дворец был готов к тому дню, когда Принцесса въедет в наш город!

– О, Отец! – промолвил младший сын. – Джины – могучи. Каждый из них способен за день вырубить из скалы и уложить одну каменную глыбу, из которых и будет построен Дворец. Но за содержание каждого джина приходится платить две золотых монеты в день. Это – дорогое удовольствие. А доберется ли Принцесса за две недели, еще неизвестно. Я думаю вот о чем. Как только джины достроят Дворец, на следующий день к нему хлынут толпы приезжих, чтобы полюбоваться чудесным сооружением. С каждого из них мы будем брать одну очень мелкую медную монету. Но приезжих будет столько, что за день мы наберем медяков на полноценный золотой. Поэтому, если Вы поручите стройку мне, я призову такое количество джинов, чтобы строительство Дворца стоило нам как можно дешевле.

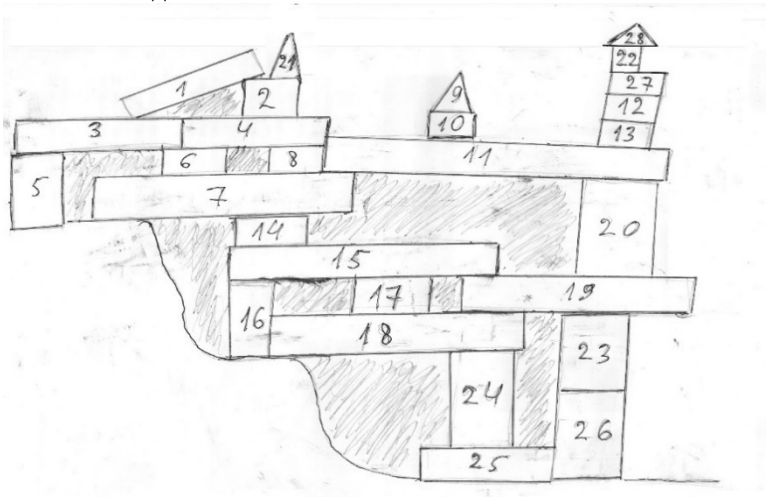
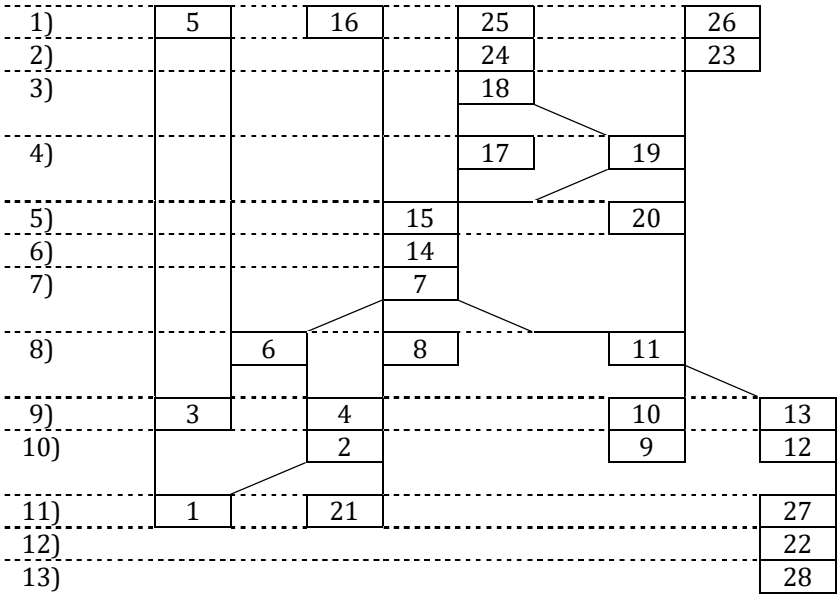


Рис. 4. Схема «дворца» из задачи на ярусно-параллельную форму конкурса «ТРИЗформашка-2014»

Требовалось определить, сколько джинов должен был бы призвать каждый из братьев для того, чтобы выполнить свое обещание?

Поскольку все джины работают одновременно, для ответа на заданные вопросы необходимо представить алгоритм построения дворца в ярусно-параллельной форме. Очевидно, что любая глыба может быть установлена только после того, как поставлены на место глыбы, на которые она опирается. Используя номера каменных глыб для обозначения вершин, получим следующий граф (слева проставлены номера ярусов, которые используются для удобства ссылок):

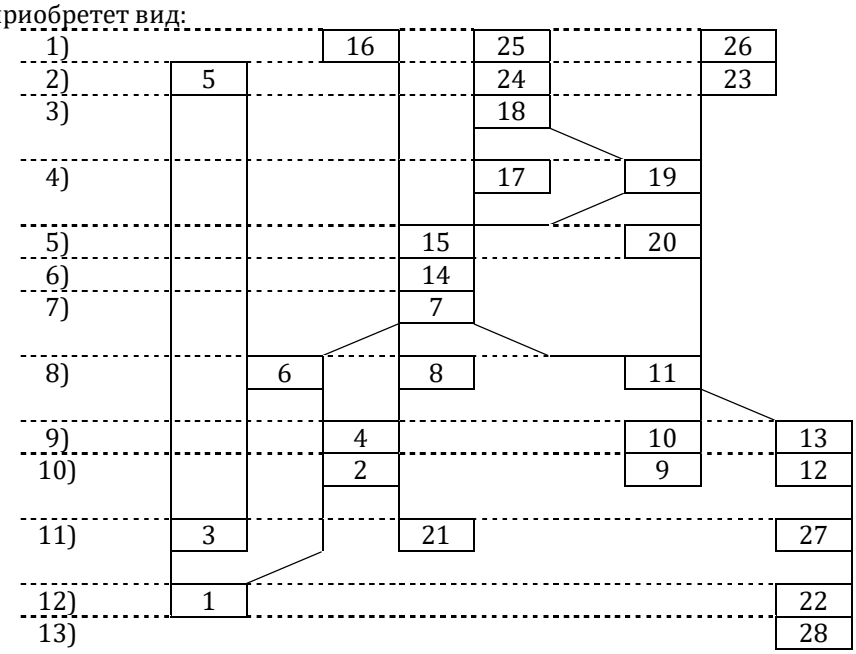


Высота ЯПФ равна 13, ширина – 4 (ширина определяется самыми широкими ярусами – первым и девятым).

Поскольку высота ЯПФ определяет минимально возможное время выполнения алгоритма, а ширина – максимально возможную степень распараллеливания, минимальное время постройки дворца равно тринадцати дням, а максимальное количество одновременно работающих джинов – четырем.

Очевидно, что ЯПФ можно легко трансформировать таким образом, что высота ее не изменится (останется равна тринадцати), а ширина уменьшится до трех. Только два яруса имеют ширину 4: первый и девятый. При этом существует семь ярусов с шириной меньше трех. Вершину 5 можно переместить на любой из ярусов со второго по седьмой. Перемещение вершины 3 потребует сдвига вершины 1. Но для этого в графе есть место. Вершину 1 можно перенести на один ярус вниз, а вершину 3 поместить на ее место.

ЯПФ приобретет вид:



Дальнейшее сужение графа (до ширины два) без увеличения его высоты уже невозможно. Камнем преткновения здесь станет ярус 7 (вершина 7). Выше его сужение возможно. В верхней части графа находятся два яруса шириной три (1 и 2), но и два яруса шириной один (3 и 6), куда можно передвинуть избыточные вершины со слишком широких ярусов. Да и ярус 7 состоит из одной вершины. Мы можем, например, вершину 16 сместить на уровень 3 (к вершине 18), а вершину 5 на ярус 6 (к вершине 14). Однако ниже яруса 7 мы имеем 4 яруса шириной три (8-11), один ярус шириной два (12) и только один ярус шириной один (13). Нам надо уменьшить ширину четырех ярусов (8-11). А увеличить мы можем ширину только одного (13). Не хватает места для трех вершин!

При ширине два для их размещения потребуется еще два яруса. То есть при ширине два граф будет иметь высоту пятнадцать.

Ответим на поставленные в задании вопросы.

Старший сын выполнить свое обещание построить дворец за неделю не сможет. Высота ЯПФ равна тринадцати. Это значит, что завершить строительство быстрее, чем за тринадцать дней, невозможно независимо от количества строителей. Критический путь, который определяет минимальный срок строительства – это путь от камней 25 и 26 до камня 28.

Средний сын, чтобы построить дворец за две недели, должен призвать трех джинов.

Выше мы сузили ярусно-параллельную форму до ширины три при сохранении высоты в тринадцать и показали невозможность ее сужения до ширины два без увеличения высоты до пятнадцати (каковая для нас уже является недопустимо большой).

В задании здесь поставлена ловушка, связанная с «линейным» восприятием процесса строительства. «Линейное» восприятие подсказывает такую цепочку рассуждений: «Нам надо установить 28 камней за 14 дней. 28:14=2. Ответ: для строительства нужны два джина». Ловушка в том, что при движении по критическому пути один раз возникает состояние, когда одновременно может работать только один джин. Второй будет простаивать.

На критическом пути лежит камень 7. Положить его можно только на седьмом шаге. На предыдущих шести шагах могут работать сразу 2 джина. Для первых пяти шагов эта работа является частью критического пути. Для шестого шага у нас есть независимый камень 5.

Но после этого складывается ситуация, когда ВСЕ остальные камни опираются на камень 7. Положить их ранее или одновременно с камнем 7 невозможно. Значит, на седьмом ходе будет уложен только один камень – камень 7. А всего за первые семь ходов будут уложены 13 камней. То есть останется 15 камней, 7 ходов и 2 джина. За 7 дней 2 джина могут уложить только 14 камней. Один камень остается не уложенным.

Младший брат должен призвать двух джинов.

Один джин построит дворец за 28 дней (по одному камню в день) и получит за это 56 монет. То есть через 28 дней дворец будет готов и будет стоить 56 золотых.

Два джина построят дворец за 15 дней и получат за это 60 золотых. После этого дворец начнет приносить доход по 1 золотому в день. За (28 – 15) = 13 дней доход составит 13 золотых. То есть через 28 дней окажется, что дворец стоил (60 – 13) = 47 золотых.

Три джина построят дворец за 13 дней и получат за это 78 золотых. После этого дворец начнет приносить доход по 1 золотому в день. За (28 – 13) = 15 дней доход составит 15 золотых. То есть через 28 дней окажется, что дворец стоил (78 – 15) = 63 золотых.

Можно посчитать срок окупаемости дворца. В конкурсной задаче этого не требовалось, но можно рассматривать срок окупаемости для оценки стоимости постройки дворца.

Дворец приносит доход в 1 золотой в день. Через сколько дней окупятся затраты на строительство дворца?

Один джин построит дворец за 28 дней и получит за это 56 монет. Дворец начнет приносить доход через 28 дней. Доход в 56 монет будет получен за 56 дней. Значит, срок окупаемости равен 28 + 56 = 84 дня.

Два джина построят дворец за 15 дней и получат за это 60 золотых. Дворец начнет приносить доход через 15 дней. Доход в 60 монет будет получен за 60 дней. Значит, срок окупаемости равен 15 + 60 = 75 дней.

Три джина построят дворец за 13 дней и получат за это 78 золотых. Дворец начнет приносить доход через 13 дней. Доход в 78 монет будет получен за 78 дней. Значит, срок окупаемости равен 13 + 78 = 91 день.

В данной статье представлены только две методические разработки по пропедевтике параллельных вычислений в школьном курсе информатики. Желающих более подробно ознакомиться с «пермским опытом» мы отошлем к серии статей, публикация которой началась в данное время в журналах «Информатика и образование» и «Информатика в школе» [8, 9].

Литература

1. Баканов В.М. Программный инструментарий анализа информационной структуры алгоритмов по их информационным графам. URL: https://www.hse.ru/pubs/share/direct/demo_document/177868104.
2. Воеводин В.В. Параллельные вычисления. /В.В. Воеводин, Вл.В. Воеводин – Спб: БХВ-Петеребург, 2002. – 608 с.
3. Воеводин В.В. Вычислительная математика и структура алгоритмов: 10 лекций о том, почему трудно решать задачи на вычислительных системах параллельной архитектуры и что надо знать дополнительно, чтобы успешно преодолевать эти трудности. – 2-е издание, стереотипное. – М.: Издательство Московского университета, 2010. – 168 с.

4. Иванова Н.Г., Плаксин М.А., Русакова О.Л. ТРИЗформашка. //Информатика. N05 (606), 1-15.03.2010. С.3-19.
5. Плаксин М.А. Информатика: учебник для 3 класса: в 2 ч. /М.А.Плаксин, Н.Г.Иванова, О.Л.Русакова. – М.: БИНОМ. Лаборатория знаний, 2013.
6. Плаксин М.А. Информатика: учебник для 4 класса: в 2 ч. /М.А.Плаксин, Н.Г.Иванова, О.Л.Русакова. – М.: БИНОМ. Лаборатория знаний, 2013.
7. Плаксин М.А. «Суперкомпьютеры» vs «параллельное программирование». «Параллельное программирование» vs «совместная деятельность». Как изучать тему «параллельные вычисления» в средней школе? // Современные информационные технологии и ИТ-образование. Научный журнал. Том 1 (№11), 2015. С.302-309.
8. Плаксин М.А. Комплект деловых игр для начального знакомства с параллельными вычислениями //Информатика в школе, 2016, №5 (118). С.6-16.
9. Плаксин М.А. Пропедевтика параллельных вычислений в школьной информатике. Тема «Совместная деятельность». //Информатика в школе, 2016, №8 (121). С.5-9.

References

1. Bakanov V.M. Programmnyy instrumentariy analiza informatsionnoy struktury algoritmov po ikh informatsionnym grafam. URL: https://www.hse.ru/pubs/share/direct/demo_document/177868104.
2. Voevodin V.V. Parallel'nye vychisleniya. /V.V. Voevodin, Vl.V. Voevodin – Spb: BKhV-Petereburg, 2002. – 608 s.
3. Voevodin V.V. Vychislitel'naya matematika i struktura algoritmov: 10 lektsiy o tom, pochemu trudno reshat' zadachi na vychislitel'nykh sistemakh parallel'noy arkhitektury i chto nado znat' dopolnitel'no, chtoby uspeshno preodolevat' eti trudnosti: uchebnik. M.: MGU, 2010. – 168 s.
4. Ivanova N.G., Plaksin M.A., Rusakova O.L. TRIZformashka [The contest TRIZformashka] //Informatika [Informatics] N05 (606), 1-15.03.2010. P.3-19.
5. Plaksin M.A. Informatika: uchebnik dlja 3 klassa: v 2 ch. [Informatics: the textbook for grade 3: 2 parts]. /M.A.Plaksin, N.G.Ivanova, O.L.Rusakova. – M.: BINOM. Laboratorija znanij, 2013.
6. Plaksin M.A. Informatika: uchebnik dlja 4 klassa: v 2 ch. [Informatics: the textbook for grade 4: 2 parts]. /M.A.Plaksin, N.G.Ivanova, O.L.Rusakova. – M.: BINOM. Laboratorija znanij, 2013.
7. Plaksin M.A. «Superkomp'yutery» vs «parallel'noe programmirovanie». «Parallel'noe programmirovanie» vs «sovmestnaya deyatel'nost'». Kak izuchat' temu «parallel'nye vychisleniya» v sredney shkole? // Sovremennye informatsionnye tekhnologii i IT-obrazovanie. Nauchnyy zhurnal. Tom 1 (№11), 2015. S.302-309.
8. Plaksin M.A. Komplekt delovykh igr dlya nachal'nogo znakomstva s parallel'nymi vychisleniyami //Informatika v shkole, 2016, №5 (118). S.6-16.
9. Plaksin M.A. Propedevtika parallel'nykh vychisleniy v shkol'noy informatike. Tema «Sovmestnaya deyatel'nost'». //Informatika v shkole, 2016, №8 (121). S.5-9.

Поступила 21.10.2016

Об авторе:

Плаксин Михаил Александрови, доцент кафедры информационных технологий в бизнесе Пермского филиала Национального исследовательского университета «Высшая школа экономики», кандидат физико-математических наук, mapl@list.ru.