Development of algorithms that simulate the solution of a problem by a person

Iurii V. Filatov[1,2]

[1]*Kryvyi Rih State Pedagogical University, 54 Gagarin Ave., Kryvyi Rih, 50086, Ukraine*
[2]*Saksahanskyi Natural Science Lyceum of the Kryvyi Rih City Council in Dnipropetrovsk oblast, 32A Meleshkina Str., Kryvyi Rih, 50071, Ukraine*

Abstract. Some algorithms, which are often based on the use of elements of higher mathematics, possessing high speed and compact coding in algorithmic languages, are poorly mastered by most students. It can be assumed that this is due to the difficulty of presenting the principles of their work in the form of human actions in ordinary situations. Thus, a certain contradiction arises between the way of solving the problem that a person resorts to without using a computer and the way we force our computer to solve this problem. Comparison of the process of explaining algorithms speaks in favor of algorithms imitating human thinking. The discussion of the advantages of the algorithms themselves is beyond the scope of this article and undoubtedly deserves a separate study. If artificial intelligence is created, then its creator or creators will certainly be ranked among the outstanding geniuses in the history of civilization, no matter what algorithms it uses. However, so far there is no one to solve problems for us and create algorithms, so we will use all available means and try to teach this to children.

Keywords: algorithms, solution of the problem

«Движенья нет» – сказал мудрец брадатый.
Другой смолчал и стал пред ним ходить,
Сильнее бы не смог он возразить.
Хвалили все ответ замысловатый.

А. С. Пушкин [1]

Некоторые алгоритмы, в основе которых зачастую лежит использование элементов высшей математики, обладая высоким быстродействием, и компактной кодировкой на алгоритмических языках, плохо усваиваются большинством учащихся. Можно предположить, что это связано с трудностью представления принципов их работы в виде действий человека в обычных ситуациях. Таким образом, возникает некоторое противоречие между тем способом решения задачи, к которому прибегает человек без использования компьютера и тем, которым мы заставляем решать эту задачу наш компьютер.

Рассмотрим возникновение такого противоречия на конкретном примере. Для этого

KMITO 1999: Conference on Computer Simulation and Information Technology in Education, April 19–21, 1999, Kryvyi Rih, Ukraine
imfilatov@mail.ru (I. V. Filatov)

ACNS Conference Series: Social Sciences and Humanities

сравним алгоритмы “пузырьковой” сортировки и быстрой сортировки Хоара (Hoare) [2]. Если мы будем сравнивать быстродействие этих алгоритмов, то заметим, что результаты прогонов на массивах разного размера, несортированных вовсе и почти упорядоченных, а также при различной настройке компилятора различны. Тем не менее, алгоритм Хоара работает в большинстве случаев значительно быстрее (исключение составит большой массив, в котором требуется поменять местами несколько пар стоящих рядом элементов). Посмотрим теперь, как обстоит дело с объяснением механизма работы этих алгоритмов.

Для объяснения работы “пузырьковой” сортировки будем ходить вдоль длинного ряда карточек с числами слева направо (можно просто записать числа на доске, но в этом случае придется отвлекаться на стирание и переписывание чисел). Напомним детям, что все видели, как учитель физкультуры выстраивает новый класс по росту, проходя вдоль шеренги учеников, которые встали в произвольном порядке, и, меняя местами двух стоящих рядом юных спортсменов, если более высокий стоит ближе к левому флангу. (Здесь уместно обратить внимание учащихся на то, что человек и компьютер сравнивают за один раз только две величины, т.е. кажущаяся возможность сравнить сразу больше двух величин, на самом деле сводится к некоторому количеству попарных сравнений)

Итак, последуем примеру учителя физкультуры, при этом будем держать в руке флажок, поднимая его в начале каждого прохождения от начала к концу шеренги чисел. При этом в поле нашего зрения одновременно находятся только две лежащие рядом карточки, и если числа на них нарушают избранный порядок (возрастание или убывание чисел в ряду должно соответствовать поставленной задаче), то мы меняем карточки местами и опускаем флажок (объясняем, что если он не был поднят, то эта процедура проделывается вхолостую, дабы избежать лишней проверки состояния флажка). В конце каждого такого похода проверяем поднят ли флажок. Если да, то, стало быть, карточки местами не менялись и следовательно сортировка окончена.

Понятно, что для столь наглядной демонстрации принципа работы быстрой сортировки Хоара потребуется минимум три человека, которые будут перемещаться вдоль ряда чисел и перекликаться, выясняя чьё число больше, но кроме этого нужен хранитель стека (или массива заменяющего стек при не рекурсивной реализации алгоритма), и, желательно, менеджер, который будет регламентировать очерёдность действий этих исполнителей. (Можно предложить построение по росту методом Хоара для КВНа.) При этом наибольшую трудность вызывает имитация стека, т.к. объём “стека” нормального человека, как правило, невелик. Попробуйте проверить глубину стека, предлагая выворачивать наизнанку числа; пока дело касается примеров “17”—“71” ошибок почти не бывает, но попробуйте на “3.244.179” получить ответ “9.714.423”.

Итак, сравнение процесса объяснения алгоритмов говорит в пользу алгоритмов имитирующих мышление человека. Обсуждение же достоинств самих алгоритмов выходит за пределы темы данной статьи и, несомненно, заслуживает отдельного исследования. Если искусственный интеллект будет создан, то его создатель или создатели наверняка будут причислены к выдающимся гениям в истории цивилизации, независимо от того какие алгоритмы он будет использовать.

Однако, пока некому решать за нас задачи и создавать алгоритмы, поэтому будем использовать все имеющиеся средства и постараемся научить этому детей.

Прежде чем перейти к решениям конкретных задач, для которых мне удалось построить алгоритмы имитирующие решение задачи человеком, приведу схему коллективной работы над задачами. Применение этой схемы позволило развить в учениках способность совмещать поиск оригинального решения с модернизацией известных алгоритмов.

1. Постановка задачи, уточнение требований, разбор предельных и вырожденных случаев.
2. Конференция на тему «Как бы я решил эту задачу, не имея компьютера и не зная алгоритмических языков». Каждый докладчик описывает последовательность своих действий, а слушатели оппонируют, предлагая "неудобные" для данного алгоритма случаи.
3. Обсуждение проблем кодирования на Turbo Pascal, доклады о встреченных ранее недостатках некоторых конструкций.
4. Самостоятельная модернизация и кодирование алгоритма, принятого за базовый в процессе обсуждения докладов.
5. Сравнение работоспособности и эффективности полученных программ, доклады победителей тестирования.
6. Необходимая индивидуальная доработка программ.

Следует отметить, что подобный подход даёт наилучшие результаты во внеклассной деятельности, при работе с небольшими группами учащихся, но ведь задача массового воспитания программистов и не ставится.

Задача о направлении обхода.

Постановка задачи:

Произвольный N-угольник задан координатами (x, y) своих вершин, пары координат перечислены в той последовательности, в которой соединяются вершины, последняя вершина соединяется с первой. Требуется определить соединялись вершины обходом по часовой стрелке или против.

Необходимое уточнение:

Первая из перечисленных точек не обязательно лежит “с краю”, кроме того, поскольку N-угольник произвольный, то точка $K + 1$, а равно и $K - 1$ расположены произвольно относительно точки с номером K.

Задача имеет красивое и компактное решение на языке векторной алгебры.

Разработка алгоритма решения:

Представим себе, что N-угольник представляет собой систему пронумерованных колышков, соединённых бечевой, а мы находимся примерно посредине этого лабиринта, возле колышка с номером K. Краёв системы не видно, зато мы хорошо видим, что севернее (юго-западнее и т.д.) находится колышек с номером K, а южнее (северо-восточнее и т.д.) с номером $K + 1$ (см. рис. 1).

Понятно, что такая ситуация не прояснит направления обхода, даже если мы точно определим направление на соседние колышки. Сравнение рисунков 1 и 2 убеждает нас в этом.

Так, находясь у третьего колышка мы обнаружим, что направление на второй и четвёртый одинаково, но на первом рисунке обход осуществлялся по часовой стрелке, а на втором против.

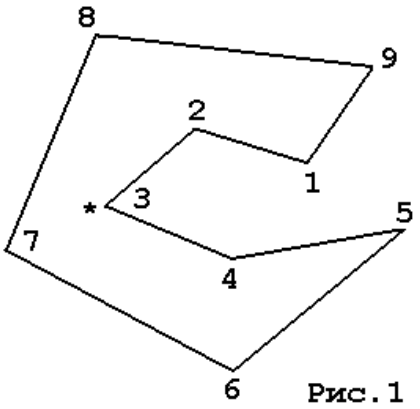

Рис.1

Попробуем теперь идти от стороны 2-3 к стороне 3-4. Отыскивая отличия, заметим, что при движении внутри замкнутой фигуры угол увеличивается, если обход совершался по часовой стрелке (рис. 1) и уменьшается в противном случае (рис. 2). При этом угол будем измерять от оси X до направления из вершины K на вершины $K-1$ и $K+1$ (как в тригонометрии, если считать вершину K началом координат).

Для того же чтобы не сомневаться, что мы внутри, выберем например верхнюю левую вершину, тогда оба угла будут наверняка больше 180° и меньше 360° (один из них может быть равен 360°). Заметим, что можно выбирать правую нижнюю (в этом случае углы будут лежать в пределах от 0° до 180°) и любую другую самую крайнюю вершину.

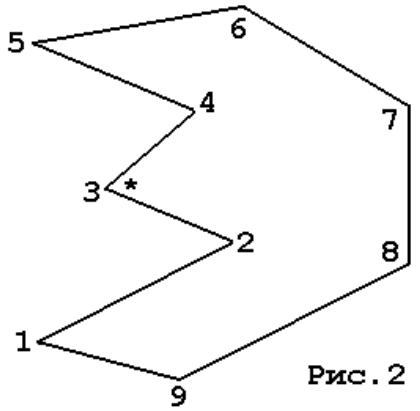

Рис.2

Таким образом, однозначно определив углы и выяснив какой больше, получаем ответ на поставленный вопрос.

Попробуем теперь преобразовать наши рассуждения в алгоритм подходящий для реализации на каком-либо языке программирования:

1. Найдём верхнюю левую вершину, для этого достаточно найти вершину с максимальной ординатой, а если таких окажется несколько, то среди них — ту, у которой абсцисса минимальна. Запомним номер этой вершины (допустим K).
2. Определим углы на $K-1$ и $K+1$ вершины. Не забудем, что последняя вершина соединяется с первой и, следовательно, если $K=N$, то следующая за K вершина — первая, а если $K=1$, то предыдущая — N. Оба угла больше 180°, кроме того надо считать угол равным 270°, если ордината $K-1$ или $K+1$ вершины равна ординате K вершины, а если соответственно равны абсциссы, то угол равен 360°, а не 0° или 180° (как было замечено выше). Поскольку нас интересует относительная величина этих углов, то перевод радиан в градусы, при написании программы, несущественен.
3. Сравним углы. Если вершины соединялись (т.е. перечислялись в условии) обходом по часовой стрелке то угол наклона стороны соединяющей вершины $K-1$ и K меньше, чем угол наклона стороны соединяющей вершины K и $K+1$ (см. рис. 3).

4. Таким образом, вычислив и сравнив углы, мы получаем ответ на поставленный вопрос.

Такой подход к разработке алгоритма и реализованные здесь идеи позволяют не только решить поставленную задачу, но и дают учащимся возможность самостоятельно строить алгоритмы для целого класса задач связанных с геометрией на плоскости, например классическую задачу о точке внутри многоугольника.

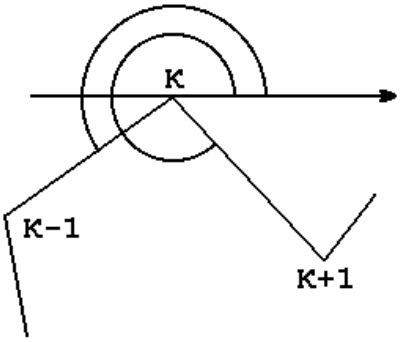

Рис. 3

Задача о точке, лежащей внутри N-угольника, достаточно известна, имеет несколько вариантов формулировки и, соответственно, несколько алгоритмов решения. Остановимся на решении задачи о нахождении точки лежащей внутри произвольного N-угольника (частные алгоритмы работающие только на выпуклых многоугольниках заслуживают отдельного разговора).

Задача о нахождении точки, лежащей внутри произвольного N-угольника.

Постановка задачи:

Произвольный N-угольник задан координатами (x, y) своих вершин, пары координат перечислены в той последовательности, в которой соединяются вершины, последняя вершина соединяется с первой. Требуется найти координаты точки, которая лежит внутри этого N-угольника.

Необходимое уточнение:

Необходимо указать минимальное расстояние d от искомой точки до ближайших сторон N-угольника, так как точность компьютерных вычислений ограничена разрядной сеткой или количеством шагов специального алгоритма, то точка математически лежащая на стороне N-угольника, программно может быть определена, как лежащая внутри (или вне) этого N-угольника. (Попробуйте, например, сравнить значение выражений $\sin 45°$ и $0.5\sqrt{2}$ на своём компьютере. Скажем, в среде Turbo Pascal 7.0 условие SIN(PI/4)=0.5*SQRT(2) выполняется, если правильно настроить компилятор, при {N+}-True, {N-}-False.)

Принимая во внимание, что при бесконечном уменьшении просвета между сторонами, попасть внутрь невозможно, разумно потребовать, чтобы расстояние между несмежными сторонами превышало $2d$.

Разработка алгоритма решения:

Итак, уяснив требования к решению, приступим к разработке алгоритма. Допустим, что нам дан многоугольник (см. рис. 4).

Мы хотим найти точку, которая заведомо лежит внутри него. Отметим, что существует алгоритм для определения, лежит ли данная точка внутри данного многоугольника, путём подсчёта количества пересечений луча, проведённого из этой точки со сторонами многоугольника. Этот алгоритм нетрудно использовать для решения данной задачи если ставить точку наугад до тех пор пока не попадём внутрь. Однако здесь возникают

определённые трудности. Во первых, как учитывать пересечения с вершинами (на рис. 5 точка лежит внутри фигуры А — нечётное количество пересечений и вне фигуры Б — чётное, а на рис. 6 количество пересечений чётное для обеих фигур; потребуется дополнительно определять лежат ли рёбра, выходящие из вершины, по одну сторону от луча или нет).

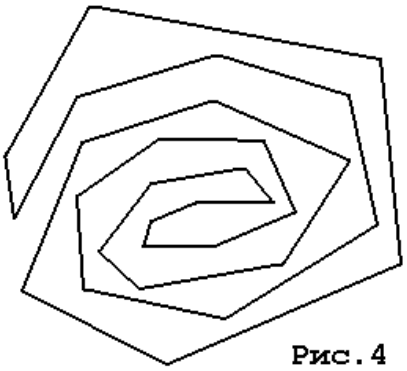

Рис. 4

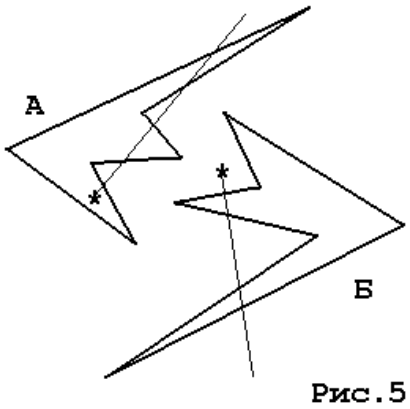

Рис. 5

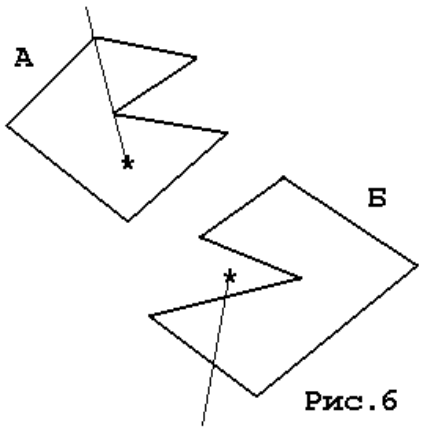

Рис. 6

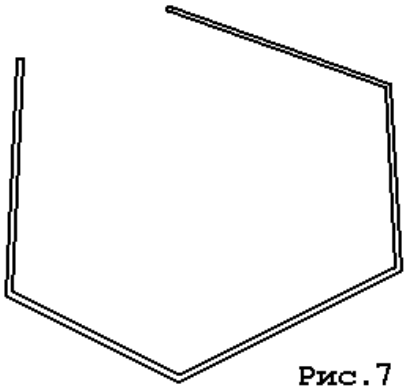

Рис. 7

Во-вторых, как долго нам придётся “стрелять”, пока мы попадём внутрь фигуры подобной той, что приведена на рис. 7.

Попробуем придумать свой алгоритм, используя метод из предыдущей задачи.

Итак, имеется достаточно запутанный случай (см. рис. 4). Трудно предположить, что человек поставит точку наугад в центре сооружения, а затем начнёт определять попал ли он внутрь, особенно если учесть, что при этом возможно мы не видим всего многоугольника. Скорее всего, человек постарается поставить точку где-нибудь поближе к краю (см. аналогичные соображения в предыдущей задаче). Дальнейшие действия очевидны:

1. По аналогии с предыдущей задачей найдем углы наклона сторон соединяющих верхнюю левую вершину с предыдущей (угол A) и последующей (B).

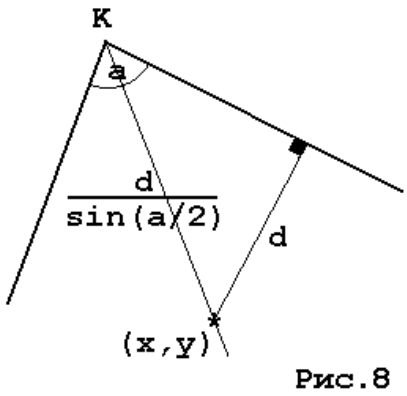

Рис.8

2. Таким образом угол наклона отрезка соединяющего искомую точку с K вершиной лежит между углами найденными в п. 1, для простоты можно считать его средним арифметическим этих углов ($C = (A + B)/2$).
3. Остаётся с помощью несложных тригонометрических расчётов найти расстояние от вершины, которое обеспечит удалённость искомой точки от сторон прилежащих к K вершине и определить (x, y) координаты этой точки (см. рис. 8). При этом α = больший угол − меньший, и расстояние от K до искомой точки $r = d/\sin(\alpha/2)$. Тогда координаты: $x = x_K + r\cos C$ и $y = y_K + r\sin C$.

References

[1] Pushkin, A.S., 1947. *Dvizhenie [Motion].* Izd-vo AN SSSR, p.432. Polnoe sobranie sochinenii, 1837—1937. Tom 2. Stikhotvoreniia, 1817—1825. Litceiskie stikhotvoreniia v pozdneishikh redaktciiakh. Kn. 1. Available from: http://feb-web.ru/feb/pushkin/texts/push17/vol02/y21-432-.htm.

[2] Sedgewick, R., 1978. Implementing quicksort programs. *Commun. ACM*, 21(10), p.847–857. Available from: https://doi.org/10.1145/359619.359631.