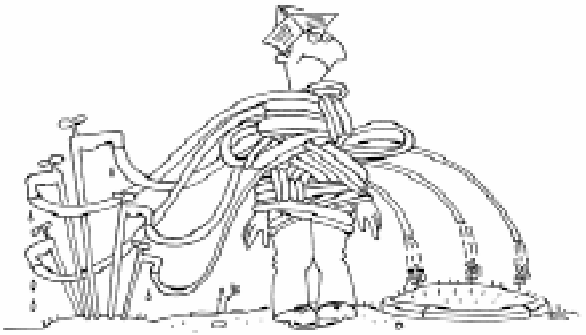
Посов Илья Александрович

ЗАДАЧА «СОРТИРУЮЩАЯ СХЕМА» КОНКУРСА КИО-2008

В этой статье мы обсудим задачу «сортирующая схема», которая была предложена на ежегодном конкурсе «Конструируй! Исследуй! Оптимизируй!» в 2008 году. В задаче требуется построить автомат, выбирающий несколько минимальных значений из заданного набора чисел. Мы узнаем условие задачи, обсудим ее решение и посмотрим на решения участников.

УСЛОВИЕ ЗАДАЧИ



В этой задаче необходимо построить схему, которая умеет из чисел, размещенных на входах X_1, X_2 ... X_n выбрать m минимальных и разместить их, соответственно, на выходах F_1, F_2 ... F_m. Схема конструируется из двух базовых элементов: минимум (min) и максимум (max). Каждый элемент имеет два входа и один выход, поэтому из двух присоединенных к нему элементов первый элемент выбирает меньший из двух, второй – больший, например, min(3;1) = 1, max(3; 2) = 3. Входы и выходы элементов можно соединять проводами, при этом во вход элемента должен входить ровно один провод, а из выхода может выходить сколько угодно. На рис. 1 изображена схема, которая выбирает минимальный элемент из трех чисел.

Чем меньше элементов использует схема, тем лучше оценивается решение. Также в конкурсе учитывались неполные решения, в которых схема не всегда давала правильный ответ. В этом случае оценивался процент перестановок чисел 1, 2, ... n, на которых схема работала правильно. Надо заметить, что схема работает правильно и выбирает первые m минимальных элементов в том и только том случае, если она работает правильно для любой перестановки чисел 1, 2, ... n. Для упрощения процесса решения в программе предусмотрена возможность объединить некоторую часто используемую конфигурацию элементов в новый элемент.

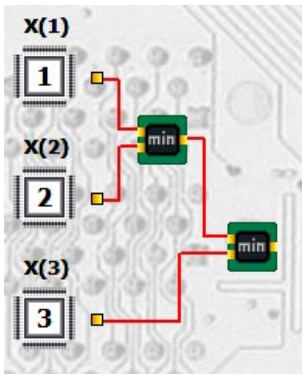


Рис. 1

Задачи первого и второго уровня отличались тем, что участники первого уровня выбирали 3 минимальных элемента из пяти, а участники второго уровня – 4 элемента из семи. Далее в статье на рисунках изображен интерфейс программы, которой участники пользовались для решения задачи.

РЕЗУЛЬТАТЫ УЧАСТНИКОВ

15 участников первого уровня и 13 участников второго сумели получить полностью работающие схемы. В разборе задачи мы приведем способ построения схемы, которым пользовались участники, получившие лучшие результаты. По некоторым схемам участников непонятно, как они решали задачи. То, что их схема работает, можно определить только по результату автоматической проверки. Пример одной из страшных, но работающих схем изображен на рис. 2.

РЕШЕНИЕ ЗАДАЧИ

Построить схему, которая выбирает один минимальный элемент из входов X_1, X_2 ... X_n не составляет труда. Достаточно воспользоваться только элементами min. Необходимо объединить входы X_1 и X_2. Далее, выход min объединить с X_3 и т. д., пока не будут использованы все входы.

Получить второй по размеру элемент уже сложнее, и здесь необходимы размышления. Сейчас мы приведем одно решение, но дальше значительно улучшим его, уменьшив количество примененных элементов. Чтобы получить второй по минимальности вход можно взять максимум из n минимумов: $\min(\cancel{X_1}, X_2 \ldots X_n)$, $\min(X_1, \cancel{X_2} \ldots X_n) \ldots \min(X_1, X_2 \ldots \cancel{X_n})$. Вычеркнутый вход X_i означает, что соответствующий минимум вычисляется от всех входов, кроме X_i. Если X_{min} является минимальным элементом, то тот минимум, в котором X_{min} вычеркнуто, будет равен второму по минимальности элементу. Остальные минимумы равны минимальному элементу. И если выбрать максимум из всех этих минимумов, будет получен второй по минимальности элемент.

Аналогично можно выбрать третий по минимальности элемент, но при таком решении количество использованных элементов будет очень большим, и мы сейчас приведем другое решение задачи.

Ограничим вид схем, которые мы будем строить при решении. Возможно, мы потеряем хорошие решения с очень маленьким числом использованных элементов, но нам хотя бы будет понятно, в каком направлении думать, чтобы построить правильную схему для выбора m минимальных элементов из n. Заведем массив A[i], значения которого вначале совпадают со значениями входов схемы X_i. Научимся делать операцию «релаксации»: если два значения A[i] и A[j] расположены в неправильном порядке, то есть $i < j$, но A[i] > A[j], то нужно поменять их местами. Это

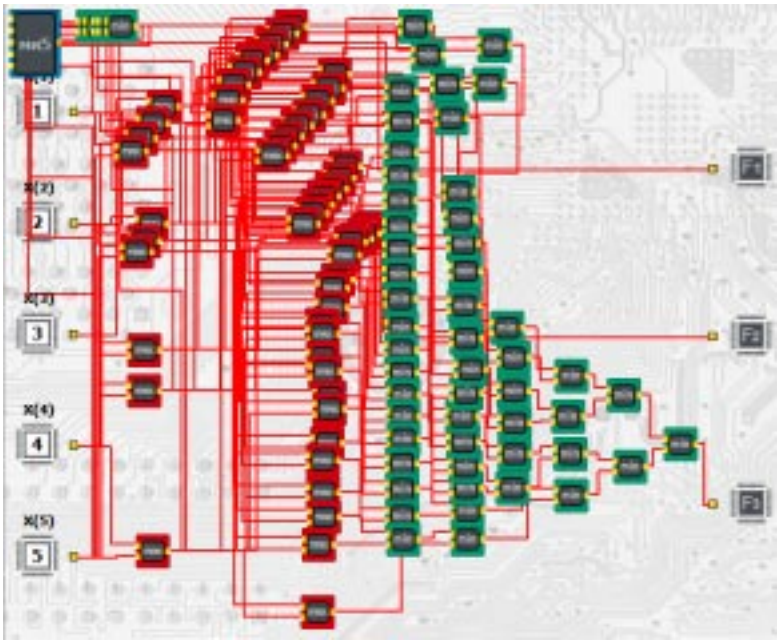


Рис. 2

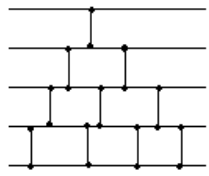
можно сделать с помощью элемента компаратор, составленного из одного min и одного max. (см. рис. 3 *а* и 3 *б* – релаксация второго и четвертого входа на схеме из пяти входов, схематичное изображение и то, как это выглядит в программе).

Мы будем составлять из компараторов схему не для выбора *m* минимальных элементов, а для сортировки всех входов схемы. Эта задача более общая, но для нее известно много методов, таких как пузырьковая сортировка, сортировка вставками или сортировка слиянием. Можно попробовать повторить их на схеме.

Сортирующие схемы подробно разобраны в книге «Искусство Программирования» Дональда Кнута, том 3. Здесь мы приведем несколько результатов оттуда, но все подробности лучше прочитать там[1].

Сначала повторим на схеме процесс сортировки методом пузырька. Что будет, если произвести релаксацию A[*n*] с A[*n* – 1], потом A[*n* – 1] с A[*n* – 2] и т. д. до A[2] с A[1]? В этом случае минимальный элемент всплывет вверх и окажется в элементе массива A[1]. Далее, можно провести серию релаксаций, которая поместит второй по размеру элемент в A[2]. И т. д. Таким способом можно отсортировать все входы. Нам нужно только выделить *m* минимальных элементов, поэтому после m серий релаксации можно остановиться. На рисунках 4 *а* и 4 *б* изображена схема,

а) *б)*



а) *б)*

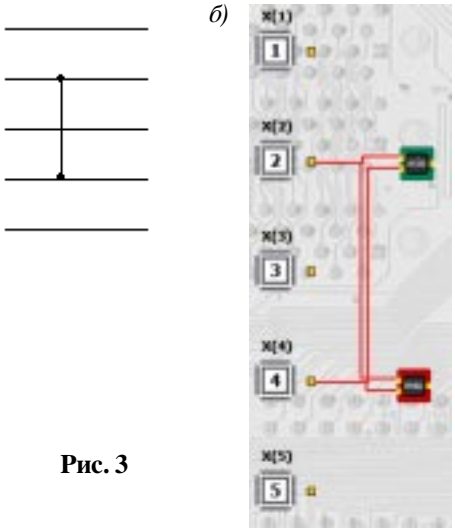


Рис. 3

решающая этим способом задачу первого уровня.

Аналогично можно повторить на схеме процесс сортировки вставками (останавливаться на этом подробно не будем). Если реализовывать это на схеме, окажется, что схема совпадает со схемой для метода пузырька.

Все лучшие решения участников используют именно приведенный только что метод эмуляции сортировки пузырьком.

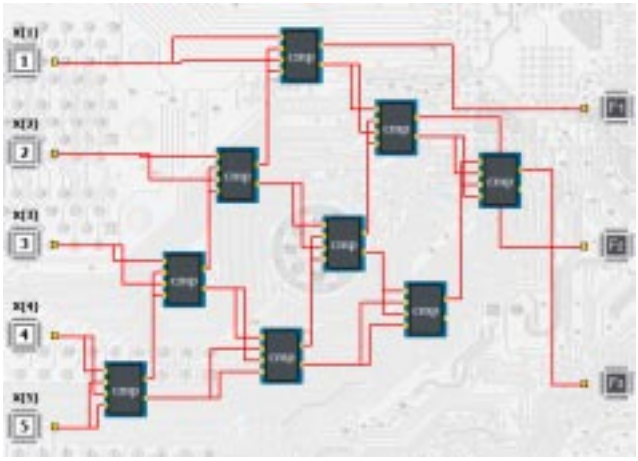


Рис. 4

[1]**Дональд Эрвин Кнут** (англ. *Donald Ervin Knuth*, родился 10 января 1938 г.) – американский учёный, почётный профессор Стэнфордского университета и нескольких других университетов в разных странах, преподаватель и идеолог программирования, автор 19 монографий (в том числе, ряда классических книг по программированию) и более 160 статей, разработчик нескольких известных программных технологий. Автор всемирно известной серии книг, посвящённой основным алгоритмам и методам вычислительной математики, а также создатель настольных издательских систем T_EX и METAFONT, предназначенных для набора и вёрстки книг, посвящённых технической тематике (в первую очередь, – физико-математических).

СОРТИРОВКА СЛИЯНИЯМИ

Более интересно повторить на схеме сортировку слияниями. Общая идея метода такова. Чтобы отсортировать массив из n элементов, мы сортируем два массива из $n/2$ элементов, а потом сливаем два отсортированных массива в один. Как можно слить два отсортированных массива с помощью сети компараторов? Повторить обычный метод слияния не получится, идея, как это сделать, принадлежит Бэтчеру, и сейчас мы ее приведем.

Чтобы понять, почему процесс слияния Бэтчера работает, нужно доказать промежуточный факт. Оказывается, чтобы проверить схему на работоспособность, можно перебирать не все $n!$ перестановок чисел от 1 до n, а всего лишь 2^n последовательностей из n нулей и единиц. Если все эти последовательности сортируются правильно, то любой набор чисел на входах также будет отсортирован правильно. Идея доказательства такова:

Пусть некоторая последовательность чисел сортируется неправильно, то есть. для некоторых $i < j$ оказалось, что после работы схемы A[i] > A[j]. Тогда заменим на входах все числа, которые не меньше A[i], на 1, а все числа, которые меньше A[i], на 0. Тогда в A[i] после работы алгоритма окажется 1, а в A[j] – 0, что противоречит тому, что любая последовательность нулей и единиц сортируется правильно.

Теперь опишем, как работает слияние двух отсортированных массивов методом Бэтчера. Построим схему S(m,n), которая сливает свои первые m входов с n последними. Пусть первый массив – это P[1..m], а второй – Q[1..n]. Если $m = n = 1$, то слить массивы очень просто, надо провести одну релаксацию P[1] с Q[1]. Если же m или n больше единицы, то сольем нечетные элементы P с нечетными элементами Q и четные элементы P с четными элементами Q (сделаем это рекурсивно). Получим два упорядоченных массива V[1..] и W[1..]. Для окончания слияния проведем следующую серию релаксаций: V[2] с W[1], V[3] с W[2] и т. д., пока это возможно. Эту серию релаксаций обозначим R(m,n).

```
V: 0 0 0 0 0 0 1 1 1 1 1 ...
W:   0 0 0 0 * * * 1 1 ...
```

Рис. 5

Чтобы проверить, что алгоритм работает, надо проверить, что он работает для произвольных последовательностей из 0 и 1. Изначально P[1..m] – это отсортированный массив, поэтому он выглядит как k идущих подряд нулей, а потом $m - k$ единиц. Аналогично массив Q[1..n] – это l нулей, а потом $n - l$ единиц. Надо проверить, что после работы схемы, построенной по методу Бэтчера, мы получим последовательность из $k + l$ нулей, а потом $m + n - k - l$ единиц.

Как устроены массивы V[1..] и W[1..]? V[1..] – это последовательность из ceil(k/2) нулей из P плюс ceil(l/2) нулей из Q. Остальное – единицы. W[1..] – это последовательность из floor(k/2) нулей из P плюс floor (l/2) нулей из Q. Остальное – единицы. Заметим, что ceil(k/2) + ceil(l/2) – floor(k/2) – floor (l/2) = 0, 1 или 2. То есть последовательность входов V и W выглядит так, как изображено на рис. 5 [1].

Символ «*» соответствует позиции, которой может начинаться последовательность 1 в W. В случае, когда разность равна 2, нули и единицы не упорядочены полностью, и для полного упорядочивания

---

[1] Функция **floor** («пол») вещественного числа x, записывается как $\lfloor x \rfloor$ или floor(x) и определяется как наибольшее целое число, не превосходящее x: $\lfloor x \rfloor = \max\{n \in \mathbb{Z} \mid n \le x\}$.

Например, floor(2.9) = 2, floor(–2) = –2, floor(–2.3) = –3. В отечественной литературе, она носит название **целая часть** или **антье** и обычно обозначается через [x].

Близко с ней связанная и реже используемая функция **ceiling** («потолок») определяется как наименьшее целое число, не меньшее x: $\lceil x \rceil = \min\{n \in \mathbb{Z} \mid x \le n\}$. Например, ceiling(2.3) = 3, ceiling(2) = 2 and ceiling(–2.3) = –2.

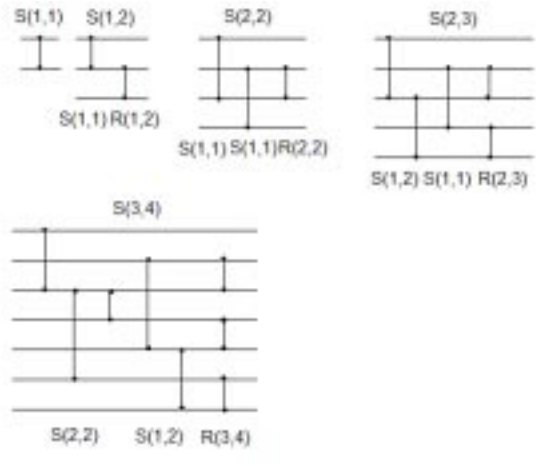


Рис. 6

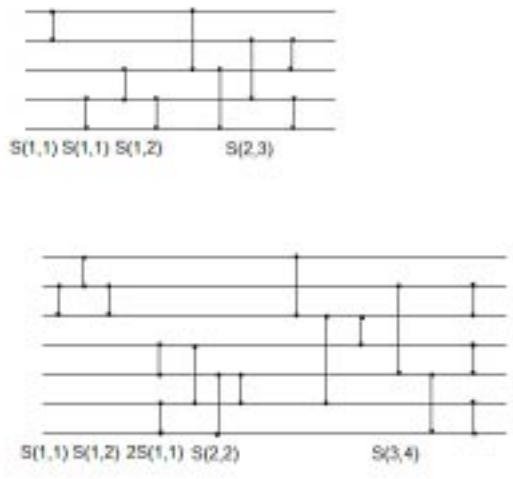


Рис. 7

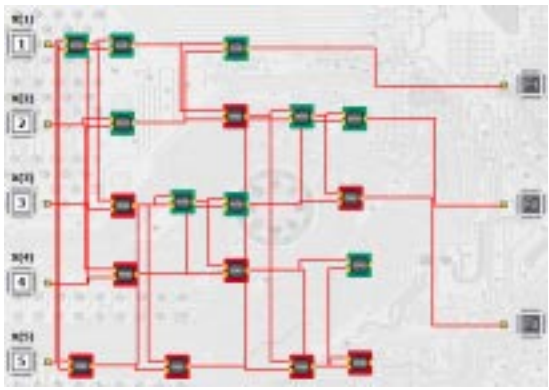
необходима описанная выше серия релаксаций R(m, n).

Применим метод, чтобы построить сортирующие сети для задач первого и второго уровня. На рис. 6 изображены сливающие сети, из которых мы построим решения задач.

На рис. 7 изображены сортирующие сети для задач первого и второго уровня, собранные из сливающих сетей. На рис. 8а и 9а изображены сети, собранные в программе в соответствии с приведенными только что сетями компараторов. Сеть на рисунке имеет несколько лишних элементов, которые нужны для сортирующей сети, но в сети для задачи их можно удалить (рис. 8б и 9б). На прилагающемся к журналу диске эти решения выложены, их можно загрузить и проверить в программе.

Построенные методом Бэтчера сети в случаях для пяти и семи входов имеют минимальное число компараторов. Для сетей больших размеров метод уже неоптимален. Конкретные значения минимального числа компараторов, необходимого для построения сортирующей сети, неизвестны. Более того, неизвестно даже, как эта величина ведет себя на бесконечности.

а)



б)

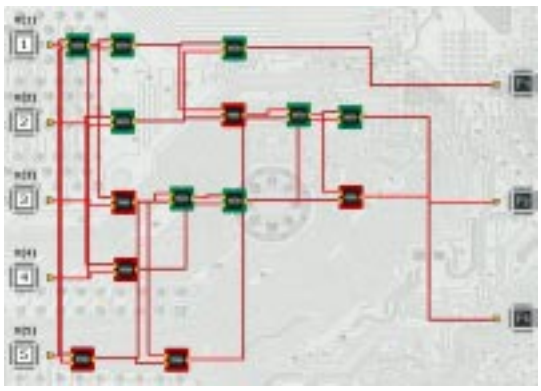
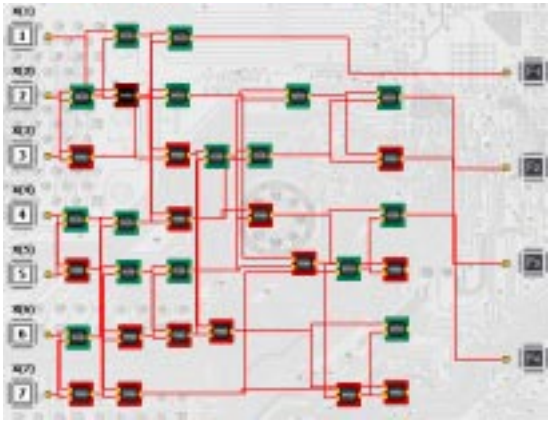


Рис. 8

а)



б)

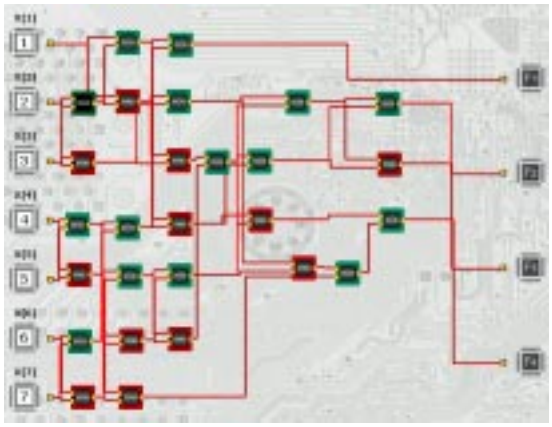


Рис. 9

Посов Илья Александрович,
аспирант математико-механического факультета СПбГУ.