Патрушев Глеб Олегович
Кандидат физико-математических наук, доцент Красноярского государственного педагогического университета; Института физики, математики и информатики.

Алгоритм организации кругового турнира

В статье рассматриваются алгоритмические подходы к организации кругового турнира (по принципу «каждый с каждым»). Предлагаются как рекурсивные, так и нерекурсивные версии алгоритма для различного числа команд. Исследуются вопросы об оптимальной продолжительности такого турнира.

Началась эта история достаточно буднично. Ко мне подошёл преподаватель физкультуры и попросил посодействовать в проведении кругового турнира по мини-футболу. Вся сложность заключалась в составлении расписания для этого турнира. Попытка алгоритмизации составления расписания натолкнулась на определённые трудности, процесс преодоления которых, думаю, весьма поучителен.

Составление матчевой сетки, в которой каждая команда должна сыграть со всеми остальными, является не такой лёгкой задачей, как могло бы показаться на первый взгляд. Процесс осложняется дополнительными требованиями, а именно:

- каждая команда должна играть не более одного матча в день;
- проведение всего турнира должно занять как можно меньше времени.

Ясно, что при нечётном числе команд каждый день, в лучшем случае одна команда не задействована в матчах (ей нет пары). В худшем случае количество неиграющих команд может, видимо, оказаться и большим – если они уже встречались в матчах ранее.

Можно попытаться оценить минимальное количество дней, необходимое для проведения такого турнира. Пусть в нём принимают участие N команд. Тогда полное количество сыгранных матчей есть число сочетаний из N по 2 (в каждом матче принимают участие 2 команды)

$$C_N^2 = \frac{N!}{2!(N-2)!} = \frac{(N-1)N}{2}.$$

Если число команд чётное, то в наилучшем случае каждый день задействованы все команды $\left(\frac{N}{2} \text{ пар}\right)$. Таким образом, турнир никак не может занять меньше $N-1$ дней.

Расписание турнира удобно строить в виде таблицы, столбцы которой соответствуют дням турнира, строки – номерам команд, участвующих в турнире. Элементом на пересечении строки i и столбца j является номер команды, с которой команда i должна провести матч в j-й день турнира. Нолик в соответствующих клетках говорит о том, что в этот день команда отдыхает.

На рисунке 1 приведено возможное распределение 5 команд по 5 дням турнира. Ясно, что этот вариант соответствует целой группе расписаний с точностью до перестановки строк и столбцов.

		Дни турнира				
		1	**2**	**3**	**4**	**5**
Номера команд	**1**	4	5	0	3	2
	2	5	3	4	0	1
	3	0	2	5	1	4
	4	1	0	2	5	3
	5	2	1	3	4	0

Рис. 1. Вариант расписания для 5 команд

Приведённое расписание было составлено вручную. Это было сделано для того, чтобы подметить какие-либо закономерности в размещении команд, понять принцип составления такого расписания и впоследствии автоматизировать его. Оказалось, что схема действий человека очень подходит под алгоритм *backtracking*'а, иначе называемый «метод отбора-отказа» или «метод проб и ошибок». Он известен широкому кругу программистов по знаменитой «задаче о ферзях» [1] – [2].

Следуя этой идеологии, вначале расставляем соперников для первой команды, затем, с учётом уже занятых команд, – для второй. Если на каком-то шаге получается слишком большое число неиграющих команд в один из дней (в идеале больше одной), возвращаемся на шаг назад, пробуем другой вариант расстановки и снова идём вперёд. Если на каком-то шаге перебор всех возможных вариантов расстановки не приводит к успеху, значит, необходимо отступать ещё на шаг назад и так далее.

Для небольшого числа команд такой метод вполне приемлем, но после многочисленных попыток составить расписание всего для 9 команд автор капитулировал. Слишком большое количество информации оказывается необходимо запоминать в стеке возвратов на каждом шаге, а именно – очерёдность расстановки команд на матчи, с тем, чтобы при откате на шаг назад отказаться именно от последнего варианта. Соответственно, следует ожидать, что и компьютерный вариант алгоритма будет излишне громоздок.

Итак, лобовой способ решения задачи потерпел фиаско. Настала пора посмотреть, а может быть, мы ломимся в открытую дверь, и для подобных задач о расстановке давно придуманы эффективные алгоритмы?

Поиск в литературе, увы, не привёл к изобилию подсказок. В основном фигурировали уже готовые таблицы турнира для 4-х, 10-ти команд и пр. Только в книге Ахо, Хопкрофта и Ульмана [3] обнаружились рекомендации и подходы к этой задаче.

Правда, авторы рассматривают частный случай, а именно, когда число команд составляет степень двойки – 2, 4, 8, 16 и так далее. Для составления матчевой таблицы применяется метод декомпозиции, который получил широкое применение не только при разработке алгоритмов, но и в проектировании электронных схем, построении математических доказательств и в других сферах.

Метод декомпозиции позволяет составить расписание для половины команд. Это расписание составляется на основе рекурсивного примене-

ния данного алгоритма для половины этой половины команд и так далее. Когда количество команд будет сокращено до двух, возникает «базовая ситуация», в которой мы просто устанавливаем порядок проведения встреч между ними.

Допустим, в турнире участвуют восемь команд. Расписание для команд 1 – 4 заполняет верхний левый угол (4 строки × 3 столбца) составляемого расписания. Нижний левый угол (4 строки × 3 столбца) этого расписания должен свести между собой команды с более высокими номерами (5 – 8). Эта часть расписания получается путём прибавления числа 4 к каждому элементу в верхнем левом углу.

Итак, нам удалось упростить задачу. Теперь остаётся свести между собой команды с низкими и более высокими номерами. Сделать это нетрудно: надо на 4-й день свести в пары команды, имеющие номера 1 – 4, с командами 5 – 8 соответственно, а в последующие дни просто циклически переставлять номера 5 – 8. Этот процесс показан на рисунке 2.

Составить программу согласно приведенным рекомендациям оказывается нетрудно. Её текст с минимально необходимыми комментариями приведён ниже.

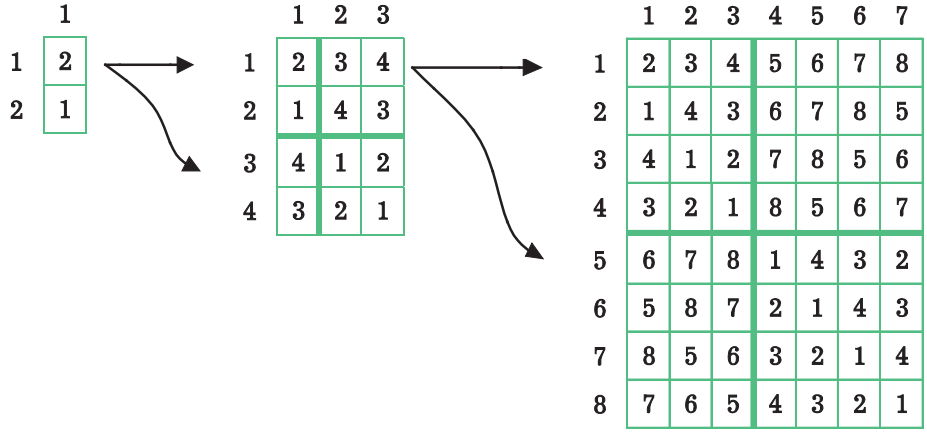

Рис. 2. Конструирование расписания методом декомпозиции для 2^k команд

```
{ рекурсивная версия }
const N = 16;     { число команд – степень двойки }
var
   { матчевая таблица }
   A : array [1..N-1, 1..N] of Byte;

procedure PrintTable;
{ печать матчевой таблицы }
var i, j : Integer;
Begin
     for j:= 1 to N do begin
         for i:= 1 to N-1 do Write (A[i,j]:3);
         Writeln;
     end;
End; { PrintTable }
```

```
procedure TableForm (K : Integer);
{ формирование матчевой таблицы для K команд }
var i, j : Integer;
Begin
     if K<>2 then TableForm (K div 2)
             else { определяем очерёдность явно }
            begin
                 A[1,1]:= 2 ; A[1,2]:= 1;
             end;
     { левый нижний угол таблицы }
     for j:= 1 to K div 2 do
         for i:= 1 to (K-1) div 2 do
             A[i, j+K div 2]:= A[i,j]+K div 2;
     { правый верхний угол таблицы }
     for j:= 1 to K div 2 do
             A[K div 2, j]:= j+K div 2;
     for i:= K div 2 +1 to K-1 do begin
         for j:= 1 to K div 2 -1 do
             A[i,j]:= A[i-1,j+1];
         A[i,K div 2]:= A[i-1,1]
     end;
     { правый нижний угол таблицы }
     for j:= K div 2+1 to K do
             A[K div 2, j]:= j-K div 2;
     for i:= K div 2 +1 to K-1 do begin
         for j:= K div 2 +1 to K do
             A[i,j]:= A[i-1,j+1];
         A[i,K]:= A[i-1,K div 2+1]
     end;
End; { TableForm }

BEGIN
     TableForm(N);
     PrintTable;
END.
```

Поскольку в программе есть лишь единственный рекурсивный вызов в начале условной конструкции, то разумной будет замена рекурсии на итерацию согласно [4]. Соответствующим образом изменённая программа имеет следующий вид.

```
{ рекурсия заменена на итерацию }
const N = 16;  { число команд – степень двойки }
var
   A : array [1..N-1, 1..N] of Byte;
   K : Integer;
procedure PrintTable;
```

```
{ печать матчевой таблицы }
var i, j : Integer;
Begin
     for j:= 1 to N do begin
         for i:= 1 to N-1 do Write (A[i,j]:3);
         Writeln;
     end;
End; { PrintTable }

procedure TableForm;
var i, j : Integer;
Begin
     { левый нижний угол }
     for j:= 1 to K div 2 do
         for i:= 1 to (K-1) div 2 do
             A[i, j+K div 2]:= A[i,j]+K div 2;
     { правый верхний угол }
     for j:= 1 to K div 2 do
             A[K div 2, j]:= j+K div 2;
     for i:= K div 2 +1 to K-1 do begin
         for j:= 1 to K div 2 -1 do
             A[i,j]:= A[i-1,j+1];
         A[i,K div 2]:= A[i-1,1]
     end;
     { правый нижний угол }
     for j:= K div 2+1 to K do
             A[K div 2, j]:= j-K div 2;
     for i:= K div 2 +1 to K-1 do begin
         for j:= K div 2 +1 to K do
             A[i,j]:= A[i-1,j+1];
         A[i,K]:= A[i-1,K div 2+1]
     end;
End; { TableForm }

BEGIN
   { определяем очерёдность явно для 2-х команд }
     A[1,1]:= 2 ; A[1,2]:= 1; K:= 1;
     while K<>N do begin
           K:= 2*K;
           TableForm;
     end;
     PrintTable;
END.
```

В [5] делается утверждение, что когда число команд является степенью двойки $N = 2^k$, турнир всегда можно провести в $N - 1$ дней. Неясными остаются два момента:

- Охватывает ли $N = 2^k$ все случаи проведения турнира в минимально допустимые $N - 1$ дней?
- Как построить расписание, если число команд отличается от

$N = 2^k$, и какова в этом случае будет продолжительность турнира?

Ответ на первый вопрос звучит отрицательно. Метод проб и ошибок даёт возможность провести турнир для $N = 6$ команд (а это не степень двойки!) за 5 дней. Соответствующая схема может выглядеть так.

	1	2	3	4	5
1	4	5	6	3	2
2	5	3	4	6	1
3	6	2	5	1	4
4	1	6	2	5	3
5	2	1	3	4	6
6	3	4	1	2	5

Рис. 3. Расписание турнира для 6 команд

Приведённые в [6] далее рекомендации по второму вопросу звучат следующим образом. При нечётном количестве участников каждый день один из них должен полностью освобождаться от проведения матчей, а проведение турнира в целом займет N, а не $N-1$ дней. Если, однако, сформировать две группы с нечётным количеством участников, тогда команды, освобождённые от проведения матчей в каждой такой группе, могут провести матч друг с другом.

Следование этому совету означает отход от ясной и простой логики половинного деления и ввязывание в дебри проверки условий типа: чётной или нечётной является очередная «половинка», на которые мы разбили список команд. Кроме того, остаётся неясным, почему верхняя граница длительности турнира при нечётном числе участников равна N? Корректного доказательства этого не приведено.

В замечательной книге венгерского математика Д. Пойа [4], которую, я считаю, следует прочесть всякому начинающему программисту, даётся совет как поступать с неподдающейся задачей. Если вольно пересказать Пойа, то её (задачу) следует видоизменить, поставить «с ног на голову», покрутить и так далее.

Именно на этом пути (покрутить задачу) неожиданно нашлось решение (неожиданно ли?), анализ которого позволил дать ответ на все поставленные выше вопросы.

Разобьём N команд на пары произвольным образом. Если число команд нечётное, дополним список команд фиктивной командой с нулевым номером. Команда, которой выпадет играть с ней, в этот день отдыхает. Полученные пары определяют матчи первого дня турнира. После чего для определения соперников в следующий день организуем «прокрутку» команд по следующей схеме (рисунок 4). Первую команду оставим на своём месте, а остальные сдвинем на одну позицию в направлении, показанном стрелками (вращать можно как в направлении движения часовой стрелки, так и против).

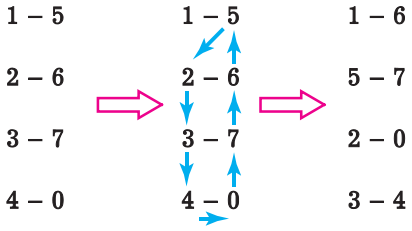

Рис. 4. Схема ротации команд при организации турнира

Получили расписание второго дня турнира. Для определения матчей третьего дня ещё раз сдвинем команды на одну позицию и так до тех пор, пока не сделаем полный круг. При таком способе ротации каждая команда сыграет со всеми своими соперниками по одному матчу без повторов.

При чётном числе команд, принимающих участие в турнире, сдвигов будет ровно на один меньше, чем команд (первая команда не принимает участие в ротации). Таким об-

разом, становится ясно, что при любом чётном количестве команд турнир можно уложить в $N-1$ день, прежде чем круг ротации команд замкнётся. При нечётном количестве команд с учётом дополнительной фиктивной команды ротация совершается за N сдвигов. Получаем, что и в этом случае N дней достаточно для проведения турнира.

Проведённое рассуждение, разумеется, не является общепринятым косвенным доказательством того факта, что круговой турнир с произвольным числом команд N можно провести в N дней. Скорее, оно относится к так называемым конструктивным доказательствам [5]. Напомним, что конструктивным называется доказательство, которое факт существования объекта с заданными свойствами (в нашем случае N-дневное расписание турнира), устанавливает, опираясь на конкретную процедуру построения такого объекта.

Программная реализация алгоритма ротации несложна.

```
const N = 11;                                  { число команд }
var
   List : array [1..N+1] of Word;       { список команд }
   A : array [0..N, 1..N+1] of Word; { таблица турнира }
   NN : Word;    { дополненное до чётного число команд }
   i : Integer;

procedure Init;
{ инициализация списка команд }
var i : Integer;
Begin
     for i:= 1 to NN do List[i]:= i;
     if Odd(N) then List[NN]:= 0;  { фиктивная команда }
End; { Init }

procedure DayToTable (k : Integer);
{ матчи k-го дня - в таблицу }
var i : Integer;
Begin
     for i:= 1 to (NN div 2) do begin
         A [List[i+NN div 2] ,k]:= List[i];
         A [List[i] ,k]:= List[i+NN div 2];
     end;
End; { DayToTable }

procedure Rotate;
{ ротация команд }
var
   Z : Word;
   i : Integer;
Begin
     Z:= List[NN div 2];
     for i:= (NN div 2) downto 3 do List[i]:= List[i-1];
     List[2]:= List [NN div 2 +1];
     for i:= (NN div 2+1) to NN-1  do List[i]:= List[i+1];
     List[NN]:= Z;
End; { Rotate }
```

```
procedure PrintTable;
{ печать таблицы }
{ горизонтальная ось - дни турнира,
  вертикаль - команды по порядку, начиная с первой }
var i, j : Integer;
Begin
     for i:= 1 to N do begin { цикл по командам }
         for j:= 1 to NN-1 do { цикл по дням }
             Write (A [i, j]:4);
         Writeln;
     end;
End; { PrintTable }

BEGIN
     { учёт фиктивной команды }
     if Odd(N) then NN:=N+1 else NN:= N;
     Init;
     for i:= 1 to NN-1 do begin
         DayToTable (i);
         Rotate;
     end;
     PrintTable;
END.
```

В заключение хочется отметить ещё одно замечательное свойство последнего способа составления расписания турнира. Этот алгоритм не только прост и изящен, но и по терминологии Кушниренко [7], он является однопроходным, то есть выполняет операцию со структурой, состоящей из N элементов, за N действий. Такие алгоритмы имеют наилучшую асимптотическую временную сложность $O(N)$.

Литература

1. *Брудно А.Л., Каплан Л.И.* Московские олимпиады по программированию. – М.: Наука, 1990. – 208 с.

2. *Зубов В.С.* Справочник программиста. – М.: Филинъ, 1999. – 266 с.

3. *Ахо А., Хопкрофт Д., Ульман Д.* Структуры данных и алгоритмы. – М.: Вильямс, 2000. – 384 с.

4. *Вирт Н.* Алгоритмы и структуры данных. – М.: Мир, 1989. – 360 с.

5. *Пойа Д.* Математика и правдоподобные рассуждения. – М.: Наука, 1975. – 464 с.

6. *Грэхем Р., Кнут Д., Паташник О.* Конкретная математика. Основание информатики. – М.: Мир, 1998. – 708 с.

7. *Кушниренко А.В., Лебедев Г.В.* 12 лекций о том, для чего нужен школьный курс информатики и как его преподавать. – М.: ЛБЗ, 2000. – 464 с.

Автор выражает благодарность
Позднякову А.В. *и* ***Саяфарову В.О.***
за обсуждение результатов настоящей работы.