Алгоритмы
и
алгоритмические языки

Лекция 14

Рекурсивные алгоритмы.
Поиск с возвратом:
задача о 8 ферзях,
поиск пути в лабиринте

Поиск с возвратами (backtracking)

Пример 1. Задача о N ферзях

На шахматной доске NxN требуется расставить N ферзей, чтобы они не били друг друга: никакие два ферзя не находятся ни на одной и той же диагонали, ни на одной и той же вертикали, ни на одной и той же горизонтали.

Найти <u>одну</u> такую расстановку

N=2 решений нет
N=3 решений нет

Рассмотрим **N=8**
4,426,165,368 вариантов
92 решения
(12 с учетом симметрии)

Решение задачи о 8 ферзях

Идея решения:

На каждом ряду доски 1 ферзь, остается найти его позицию в этом ряду (решение задачи – 8 позиций)

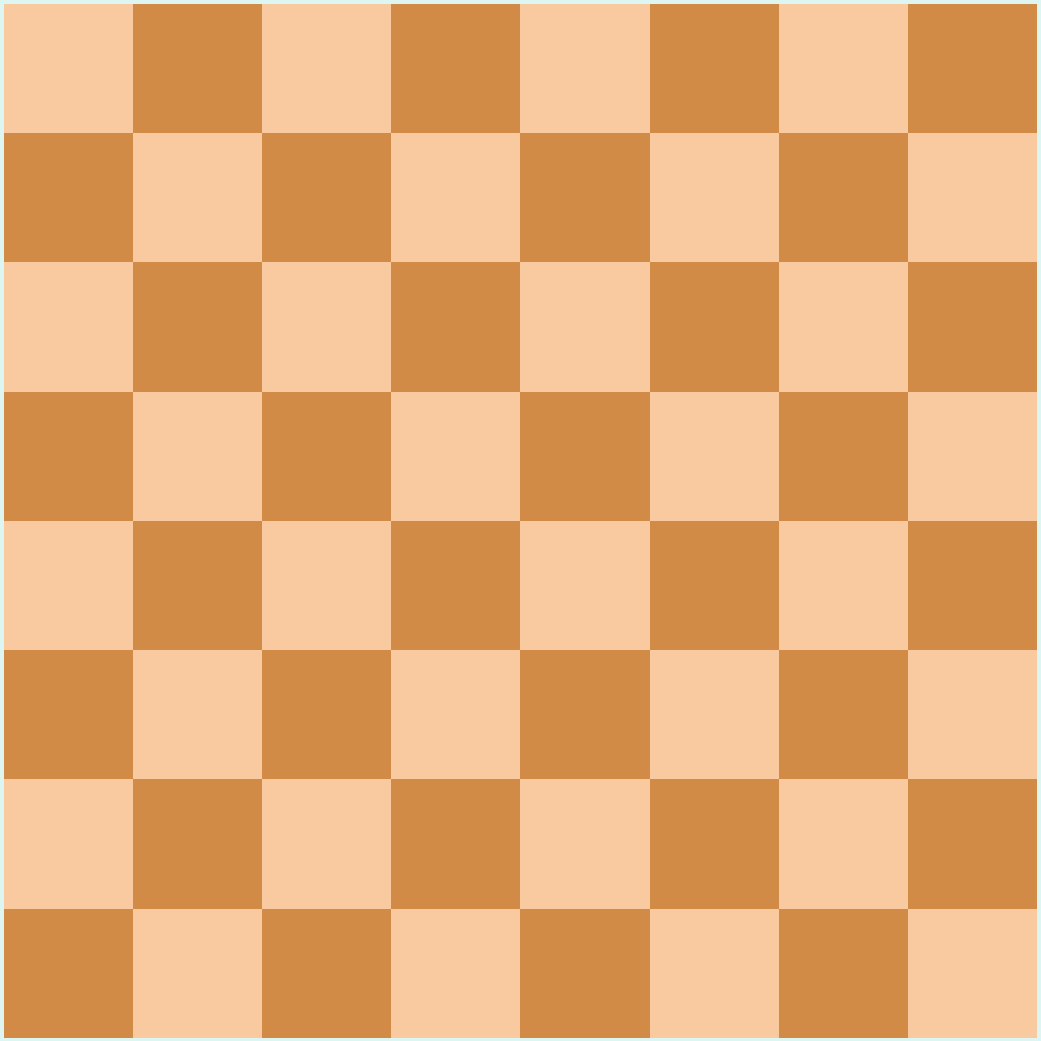
Ставим 1-го ферзя на произвольную клетку 1 ряда (все клетки допустимые),

отмечаем клетки, которые «под ударом» 1-го ферзя (недопустимые клетки),

пытаемся решить задачу расстановки для оставшихся 7 ферзей (с учетом недопустимых клеток).

Если удалось – расстановка закончена,
иначе – **возврат**: 1-го ферзя надо переставить на другую клетку его ряда и продолжить **поиск** решения

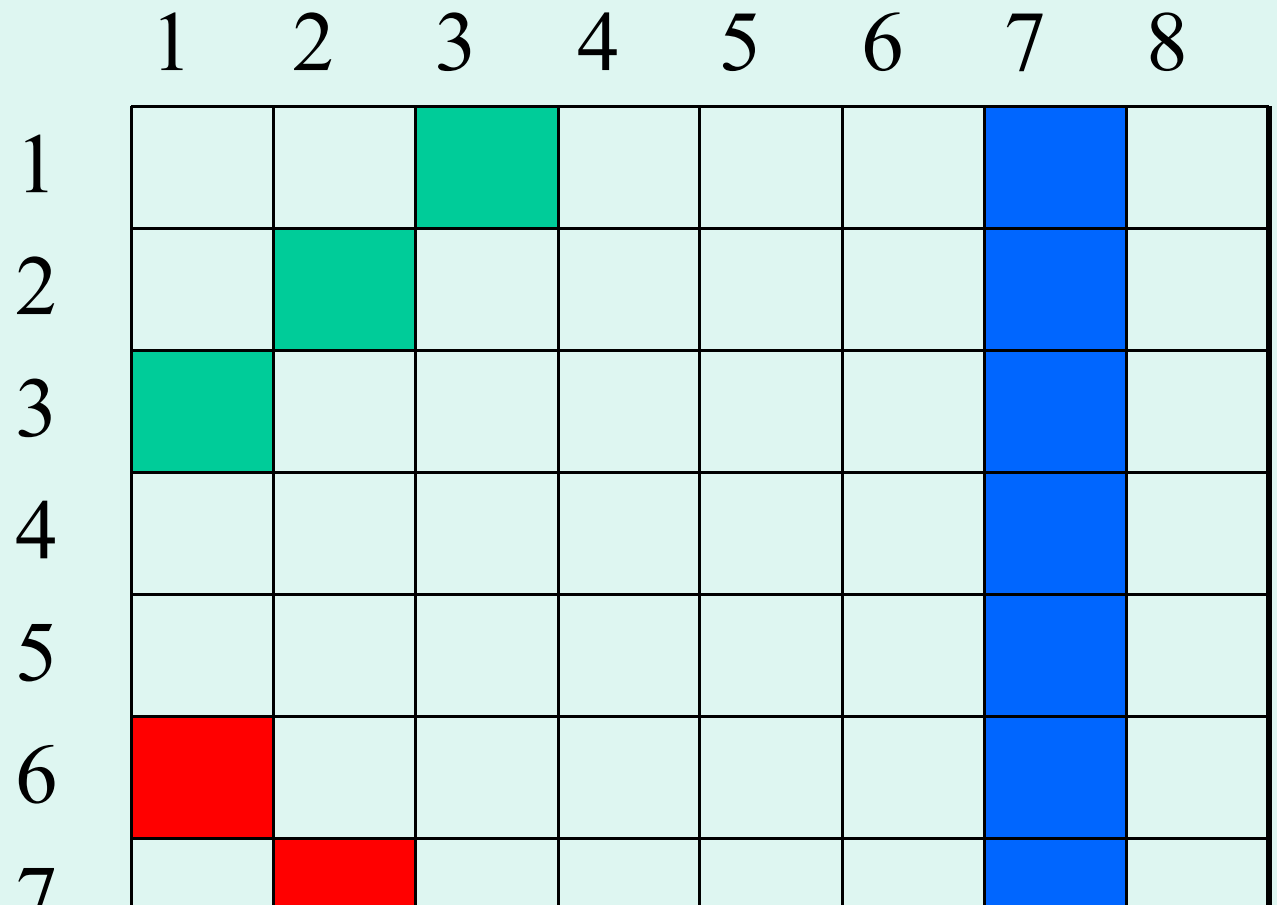
Решение задачи о 8 ферзях



Хранение информации в процессе решения

x : array[1 .. 8] of integer;

для хранения решения, в i-й координате будет получена позиция (вторая координата) i-го ферзя



a : array[1 .. 8]
of boolean;

b : array[2 .. 16]
of boolean;

c : array[-7 .. 7]
of boolean;

```
  procedure try ( i : integer; var q : boolean);
{ищем позицию ферзя в i-й строке, в  q – ответ, удалось
   найти или нет}
begin j := 0;
 repeat j := j + 1;
        {перебираем по очереди все позиции j, пока не найдем
          подходящую или пока не рассмотрим все}
   if {позиция доступна} then
      begin  {объявим ее решением}
             {отметим клетки, которые «бьет» этот ферзь}
             {если еще расставлены не все 8 – ставим
                оставшихся ферзей – рекурсия. В случае
                неуспеха – возврат назад: убираем ферзя с
                позиции j и убираем отметку клеток.
             иначе – решение найдено}
      end
 until q or (j = 8);
end;
```

Основная программа:

```
 begin
{Инициализация массивов с позициями – они все доступны}
for i := 1 to 8 do a[  i] := true;
for i := 2 to 16 do b[  i] := true;
for i := -7 to 7 do c[  i] := true;
{Ищем позицию 1-го ферзя (не противоречащую остальным)}
try( 1, q);
{Побочный эффект процедуры try – изменение массивов}

{Печать решения}
if q then
  for i := 1 to 8 do write( x[ i]:4);
writeln
end.
```

Решение задачи о 8 ферзях

Идея решения:

На каждом ряду доски 1 ферзь, остается найти его позицию в этом ряду (решение задачи – 8 позиций)

Ставим 1-го ферзя на произвольную клетку 1 ряда (все клетки допустимые),

отмечаем клетки, которые «под ударом» 1-го ферзя (недопустимые клетки),

пытаемся решить задачу расстановки для оставшихся 7 ферзей (с учетом недопустимых клеток).

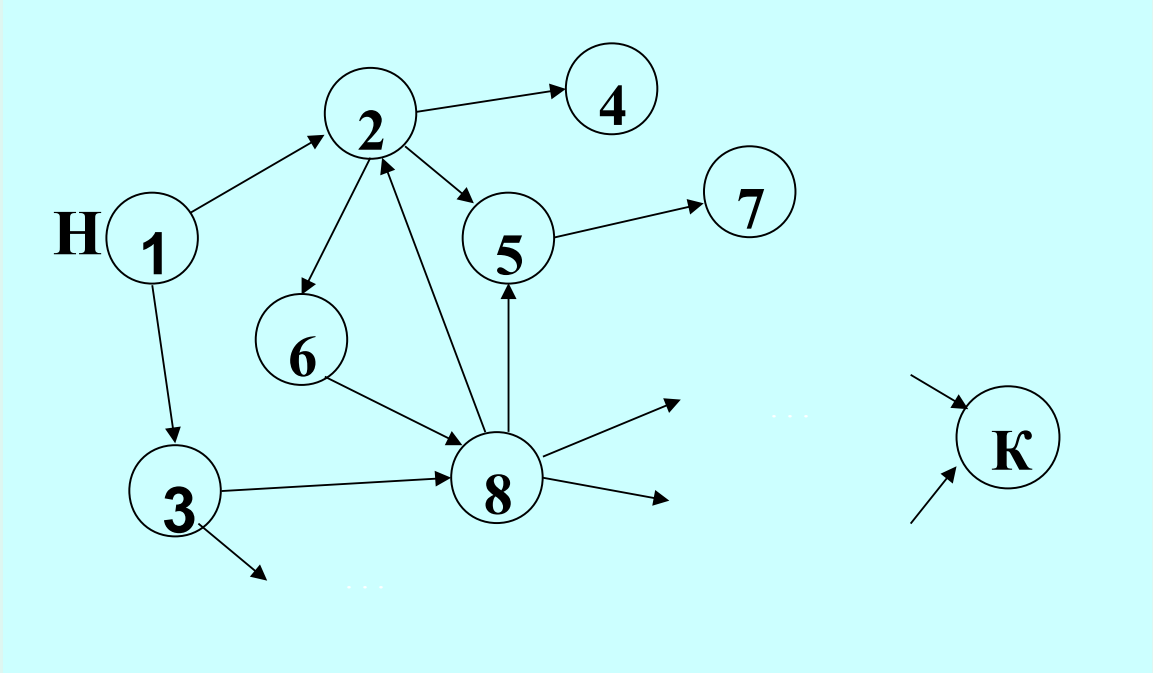
Если удалось – расстановка закончена,
иначе – **возврат**: 1-го ферзя надо переставить на другую клетку его ряда и продолжить **поиск** решения

Поиск с возвратами (backtracking)

Пример 2. Поиск пути в лабиринте

Имеется N пунктов, перенумерованных от 1 до N.
Между некоторыми пунктами есть дороги (проведены стрелки).
Даны два пункта – с номерами Н (начало) и К (конец).
Требуется найти путь от Н к К, если он есть

Пример лабиринта



Поиск пути в лабиринте

Уточнение требований:

1. Учитываем возможность зацикливания
2. Не исследуем повторно то пункты, в которых уже были

visited :array [1..n] of boolean

$$\text{visited}[i] = \begin{cases} true & - \text{уже были в пункте } i \\ false & - \text{еще не были} \end{cases}$$

Поиск пути в лабиринте

Реализация поиска с возвратами с помощью рекурсии :

Находимся в i, ищем путь в K

$$\mathrm{Path(i)} = \begin{cases} true \ - \text{есть путь из i в K} \\ false - \text{иначе} \end{cases}$$

Находим стрелку в пункт j, который еще не посещали (visited[j] = false).
Ищем путь из j в K – аналогичная задача – **рекурсия Path(j,K)**

Нерекурсивные случаи:

j=K – дошли до конечной вершины, путь найден ***true***
из i не выходит ни одной стрелки, пути нет **false**

Поиск пути в лабиринте

Представление данных :

Путь

массив way[1..N] , длина пути length :integer

Лабиринт

$$Lab[i, j] = \begin{cases} true - \text{есть стрелка из i в j} \\ false - \text{иначе} \end{cases}$$

	1	2	3	4	5	6	7	8	
1	0	1	1	0	0	0	0	0	
2	0	0	1	1	1	0	0	0	

```
const N = ...;  {число пунктов в лабиринте}
 type point = 1..N; {номера пунктов}
        labirinth = array [point, point] of boolean; {тип лабиринта}

procedure PrintPath (var Lab :labirinth; H,K :point);
var way: array[point] of point;  length: integer;
     visited: array[point] of boolean;
     j: point;
 function Path(i: point): boolean;  {есть ли путь из i в K?} ……..
 begin
   way[1]:=H; length:=1;  {начало пути}
   visited[H]:=true; {в H уже были}
   for j:=2 to N do visited[j]:=false;
   if Path(H) then {есть путь из H в K -> вывести его}
   begin
     write('путь: '); for j:=1 to length do write(way[j], '  '); writeln
   end
    else writeln('no way')
end;
```

```
function Path(i: point): boolean;
  label 99;
  var j:  point;
 begin    Path:=true;
  if i=K then goto 99;
  for j:=1 to N do
   begin
    if Lab[i,j] then
      begin
        if not visited[j] then
        begin  length:=length+1; way[length]:=j;      {включить j в путь}
                visited[j]:=true;
                 if Path(j) then goto 99;   {нашли путь из i в K}
                 length:=length-1;      {не нашли - убрать j из пути}
         end
     end
  end; {of for}
   Path:=false;  {нет пути из i в K}
99:     end; {of Path}
```

Комбинированные типы

Путь (из предыдущей задачи)
массив way[1..N] ,
длина пути length :integer } **один** объект!