

连连看实验报告

郭一隆 (2013011189)

September 4, 2015

Contents

1 原创性	3
2 制作自己的连连看	3

List of Figures

2.1 运行连连看	3
2.2 重写 <code>detect.m</code> 后运行 <code>linkgame</code>	7
2.3 测试 <code>matchadj()</code> 功能	8
2.4 定义上边界	9
2.5 依次调用 <code>matchadj</code> 和 <code>matchborder</code> 后	11

List of Tables

List of Source Codes

2.1 <code>canlink.m(adjcross)</code>	4
2.2 <code>canlink.m(canlink0)</code>	4
2.3 <code>canlink.m(canlink1)</code>	5
2.4 <code>canlink.m(main)</code>	6
2.5 <code>detect.m(main)</code>	6
2.6 <code>omg.m(matchadj)</code>	8
2.7 <code>omg.m(matchborder)</code>	10
2.8 <code>omg.m(matchrest)</code>	12

1 原创性

本次连连看大作业由本人独立思考完成，如有雷同，纯属巧合。

2 制作自己的连连看

1. 在 MATLAB 环境下，设置当前路径为 linkgame，运行 linkgame(打开linkgame.fig或右键linkgame.p点“运行”)，熟悉游戏。

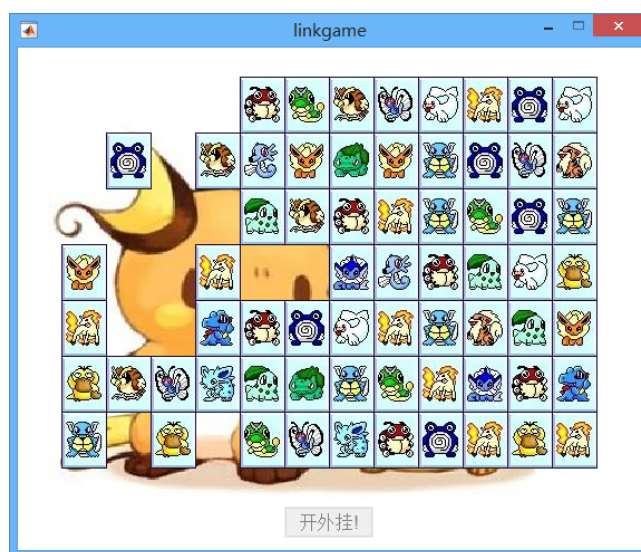


Figure 2.1: 运行连连看

2. 注意 linkgame 目录下有个 detect.p，它的功能是检测块是否可以消除。现将其删掉，然后把 linkgame\reference 目录下的detect.m复制到 linkgame 目录下。detect.m文件中是 detect 函数，函数以图像块的索引矩阵与要判断的两个块的下标为输入，如果两个块能消掉则输出 1，否则输出 0。根据文件中的注释提示，实现判断块是否可以消除的功能。写完后再次运行 linkgame，检验游戏是否可以正常运行。

算法实现分为以下几个步骤：

- (a) 画十字 adjcross：在给定点周围空白处画十字，遇到块则停止。

```

29 function [I,J] = adjcross(mtx,x,y)
30 % return index vector of can-reach blocks of mtx(x,y) on both directions
31 % notice: x >= 2 && y >= 2 && all(mtx >= 0)
32
33 % get vectors of four directions
34 left = mtx(x,1:y-1);
35 right = mtx(x,y+1:end);
36 up = mtx(1:x-1,y);
37 down = mtx(x+1:end,y);
38
39 % get zero run length adjacent to mtx(x,y)
40 lz = find(cumsum(left,'reverse')==0);
41 rz = find(cumsum(right,'forward')==0);
42 uz = find(cumsum(up,'reverse')==0);
43 dz = find(cumsum(down,'forward')==0);
44
45 I = [x*ones(1,length([lz,rz])),uz.',x+dz.'];
46 J = [lz,y+rz,y*ones(1,length([uz,dz]))];
47
48 end

```

Listing 2.1: canlink.m(adjcross)

(b) 判断是否可直连 canlink0 : 先判断横纵坐标是否在同一直线, 再确认路径上是否有障碍。

```

51 function bool = canlink0(mtx,x1,y1,x2,y2)
52 % return 1 if it's a direct link (no turns)
53
54 if x1 == x2 && ~any(mtx(x1,min(y1,y2)+1:max(y1,y2)-1))
55     bool = 1;
56 elseif y1 == y2 && ~any(mtx(min(x1,x2)+1:max(x1,x2)-1,y1))
57     bool = 1;
58 else
59     bool = 0;
60 end
61
62 end

```

Listing 2.2: canlink.m(canlink0)

(c) 判断是否可用不超过一个直角的连线连接 canlink1 : 选取两目标点之一作为起点, 画十字; 对十字上的点进行遍历, 检查是否存在可与另一目标点直连的点。

```

65 function bool = canlink1(mtx,x1,y1,x2,y2)
66 % return 1 if the turns of link path <= 1
67
68     if canlink0(mtx,x1,y1,x2,y2)
69         bool = 1;
70         return
71     end
72
73     % grow the cross of origin
74     [I,J] = adjcross(mtx,x1,y1);
75
76     for n = 1:length(I)
77         i = I(n); j = J(n);
78         if canlink0(mtx,i,j,x2,y2)
79             bool = 1;
80             return
81         end
82     end
83
84     bool = 0;
85
86 end

```

Listing 2.3: canlink.m(canlink1)

(d) **判断可连性 canlink** : 选取两目标点之一作为起点，画十字；对十字上的点进行遍历，检查是否存在可与另一目标用不超过一个直角的连线连接的点。

```

1 function bool = canlink(mtx,x1,y1,x2,y2)
2 % return 1 if these two blocks can link!
3     if mtx(x1,y1) ~= mtx(x2,y2) || ~mtx(x1,y1) || ~mtx(x2,y2)
4         bool = 0;
5         return
6     end
7
8     if canlink1(mtx,x1,y1,x2,y2)
9         bool = 1;
10        return
11    end
12
13    % grow the cross of origin
14    [I,J] = adjcross(mtx,x1,y1);
15
16    for n = 1:length(I)
17        i = I(n); j = J(n);

```

```

18         if canlink1(mtx,i,j,x2,y2)
19             bool = 1;
20             return
21         end
22     end
23
24     bool = 0;
25
26 end

```

Listing 2.4: canlink.m(main)

再次运行 linkgame, detect.m功能正常。

```

1  function bool = detect(mtx, x1, y1, x2, y2)
2
3      [m,n] = size(mtx);
4
5      % add surrounding zeros
6      mtx = [0,zeros(1,n),0;
7             zeros(m,1),mtx,zeros(m,1);
8             0,zeros(1,n),0];
9
10     origin = mtx(x1+1,y1+1);
11     target = mtx(x2+1,y2+1);
12
13     if origin == target && canlink(mtx,x1+1,y1+1,x2+1,y2+1)
14         bool = 1;
15     else
16         bool = 0;
17     end
18
19 end

```

Listing 2.5: detect.m(main)

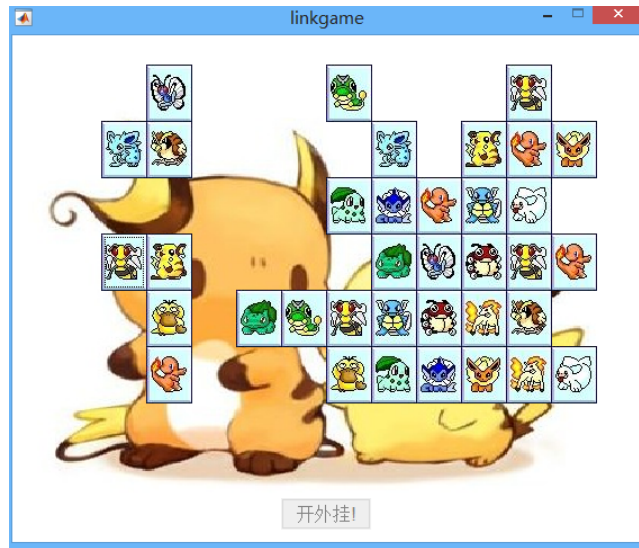


Figure 2.2: 重写 detect.m 后运行 linkgame

3. “外挂”模式逐一自动消除所有的块的功能是由 link 目录的 omg.p 实现的。删掉 omg.p 重新实现 omg.m。

根据游戏经验，算法通过以下几个步骤实现：

(a) 消去相邻的相同块 matchadj：利用自带函数 diff 实现，注意及时更新原矩阵以及保证多个相同块连续相邻时仍正确工作。

```

42 function [steps,mtx] = matchadj(mtx)
43 % match adjacent removable blocks
44
45 steps = [];
46
47 % match adjacent blocks
48 % using diff() is effective
49 % row difference
50 rdif = diff(mtx,1,1);
51 [I,J] = find(rdif==0);
52 for n = 1:length(I)
53     % empty blocks
54     if mtx(I(n),J(n)) ~= 0 && mtx(I(n)+1,J(n)) ~= 0
55         % match!
56         steps = [steps,I(n),J(n),I(n)+1,J(n)];
57         % update mtx
58         mtx(I(n),J(n)) = 0;

```

```

59         mtx(I(n)+1,J(n)) = 0;
60     end
61 end
62
63 % column difference
64 cdiff = diff(mtx,1,2);
65 [I,J] = find(cdiff==0);
66 for n = 1:length(I)
67     % ?empty blocks
68     if mtx(I(n),J(n)) ~= 0 && mtx(I(n),J(n)+1) ~= 0
69         % match!
70         steps = [steps,I(n),J(n),I(n),J(n)+1];
71         % update mtx
72         mtx(I(n),J(n)) = 0;
73         mtx(I(n),J(n)+1) = 0;
74     end
75 end
76 end

```

Listing 2.6: omg.m(matchadj)



Figure 2.3: 测试 matchadj() 功能

matchadj 函数工作正常。

- (b) 消去同一条边界上的相同块 matchborder : 先定义上边界,如图2.4中的蓝色区域。

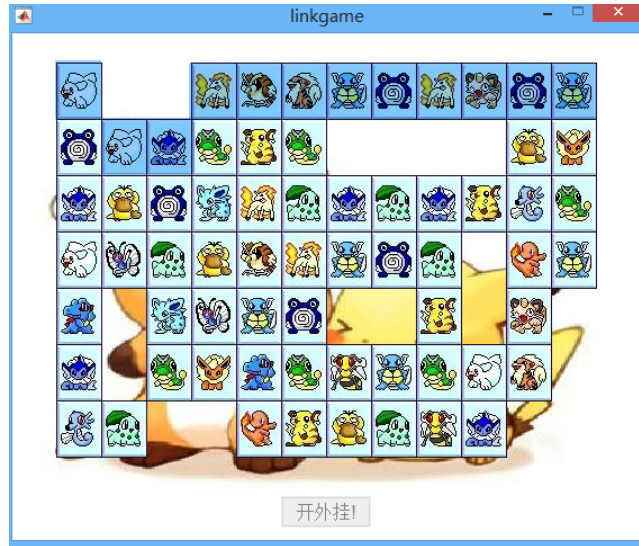


Figure 2.4: 定义上边界

显然，上边界即每一列第一个非零元素；类似地可以定义其他边界。

容易证明，在同一边界上的相同块必然可消去。

与 `matchadj` 不同，由于消去过程会使边界发生变化，故必须不断循环直至边界保持不变。则实现 `matchborder` 函数如下：

```

79 function [steps,mtx] = matchborder(mtx)
80 % match blocks on the same border
81
82 [m,n] = size(mtx);
83 steps = [];
84
85 isstable = 0; % whether mtx is stable, i.e. no more removable pairs on borders
86 while isstable ~= 4 % stable on four borders
87     for k = 1:4 % four borders
88         if k == 1
89             % upper border
90             ub = sum(cumsum(mtx)==0) + 1;
91             ub(ub>m) = nan; % a column of zeros
92             index = sub2ind(size(mtx),ub,1:n);
93         elseif k == 2
94             % bottom border
95             bb = m - sum(cumsum(mtx,'reverse')==0);
96             bb(bb<1) = nan;
97             index = sub2ind(size(mtx),bb,1:n);
98         elseif k == 3

```

```

99         % left border
100         lb = sum(cumsum(mtx,2)==0,2) + 1;
101         lb(lb>n) = nan;
102         index = sub2ind(size(mtx),1:m,lb. ');
103     else
104         % right border
105         rb = n - sum(cumsum(mtx,2,'reverse')==0,2);
106         rb(rb<1) = nan;
107         index = sub2ind(size(mtx),1:m,rb. ');
108     end
109     index = index(~isnan(index)); % delete nan
110     blocks = mtx(index);
111
112     b = unique(blocks);
113     if length(b) < length(blocks) % same blocks
114         isstable = 0;
115         for be = b
116             i = find(blocks==be);
117             if length(i) >= 2
118                 [I,J] = ind2sub(size(mtx),index(i(1:2)));
119                 % add steps
120                 steps = [steps,I(1),J(1),I(2),J(2)];
121                 % update mtx
122                 mtx(index(i(1:2))) = 0;
123                 break
124             end
125         end
126     else
127         isstable = isstable + 1;
128         if isstable == 4
129             break
130         end
131     end
132 end
133 end
134
135 end

```

Listing 2.7: omg.m(matchborder)

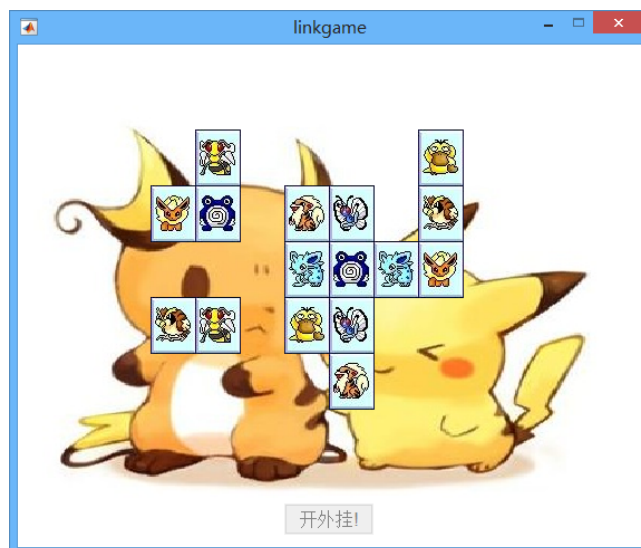


Figure 2.5: 依次调用 matchadj 和 matchborder 后

经过多次测试，发现 matchadj 与 matchborder 相结合的算法已经能消去游戏区域的绝大多数块，甚至有时可全部消去。

matchadj 的核心是 MATLAB 自带的 diff 函数，效率较高；

而 matchborder 通过 cumsum 以及 sum 函数巧妙获得各边界索引，再利用 unique 以及 find 等方法在边界内寻找相同的块，可以说十分高效。

既然高效的前两步已可以消去大多数块，那么对于剩余的块，不妨采取较暴力的算法解决。

- (c) 对于剩余的块按种类遍历尝试连接 matchrest：这里假设生成的连连看游戏是可以以任意消除顺序完全消除的（实践观测结果如此）。

```

138 function [steps,mtx] = matchrest(mtx)
139 % match rest blocks
140
141 steps = [];
142
143 while any(any(mtx)) % game NOT over
144     kinds = unique(mtx); % get kinds of blocks
145     kinds = kinds(2:end); % remove 0
146
147     for k = 1:length(kinds)
148         kind = kinds(k);
149         [I,J] = find(mtx==kind);

```

```

150         for m = 1:length(I)
151             for n = m+1:length(I)
152                 if canlink(mtx,I(m),J(m),I(n),J(n))
153                     steps = [steps,I(m),J(m),I(n),J(n)];
154                     % update mtx
155                     mtx(I(m),J(m)) = 0;
156                     mtx(I(n),J(n)) = 0;
157                     break
158                 end
159             end
160         end
161     end
162 end
163
164 end

```

Listing 2.8: `omg.m(matchrest)`

经过多次测试，均能顺利完成功能。