云南大学数学与统计学院
《算法图论实验》上机实践报告

课程名称： 算法图论实验	**年级：** 2015 级	**上机实践成绩：**
指导教师： 李建平	**姓名：**	**专业：**
上机实践名称： 编程实现图搜索算法	**学号：** 20151910042	**上机实践日期：** 2018-10-16
上机实践编号： 1	**组号：**	

一、 实验目的

1. 理解广度优先搜索算法的具体步骤；
2. 学会读开源的代码库，并逐步学会使用这些代码库完成扩展性的实验。

二、 实验内容

1. 用形式化伪代码表示图的广度优先搜索算法；
2. 借助开源代码库，完成高质量的广度优先搜索算法编程。

三、 实验平台

Windows 10 Pro 1809；

Cygwin GCC, G++编译器；

四、 算法设计

本次理论课上所讲的 Searching 算法在图论中一般被称为广度优先的图遍历算法（Breath First Searching, BFS）。在一定规则下循环地使用这个算法可以对一个图进行遍历，并得到所有的连通子图（连通分支）。这个算法十分重要，它是 Dijkstra 算法以及更一般的 Prim 算法的基础与原型。

下面对 Searching 算法（广度优先图遍历算法）进行形式化描述。

Algorithm 图的反圈遍历算法，记此算法为SEARCH

Input 图$G=(V,E)$，并假定图G是无向图；
图G中的某个起点v_1

Output 自v_1出发所有有路可到达的点以及路过的边所构成的诱导子图，记之为$\varepsilon-\mathbf{CLOSURE}=$

Begin $\mathrm{SEARCH}(\mathbf{G},\ \mathbf{v_1})$

Step 1

$X=\{v\in S|v \text{ is } M-\text{unsatuated vertex}\}$
for each vertex $v\in X$
  for each vertex $u\in B.V-\{v\}$
    $u.\mathbf{color}=\mathbf{White}$

Step 2

    $u.\mathbf{d}=\infty$

$u.\boldsymbol{\pi} = \text{NIL}$

$v.\mathbf{color} = \mathbf{Gray}$

Step 3 $v.\mathbf{d} = 0$

$v.\pi = \text{NIL}$

Step 4 $Q = \emptyset$

$\text{ENQUEUE}(Q,\ v)$

Step 5

while $Q \neq \emptyset$:

$u = \text{DEQUEUE}(Q)$

if $u \in S$:

$\boldsymbol{J} = \{x \in B.\text{ADJ}[u] \mid (\text{u},\ \text{x})$ **is not** M_{edge} **and** x **is White**$\}$

if $\boldsymbol{J} = \emptyset$:

continue

else:

End **for each** $x \in \boldsymbol{J}$:

$x.\mathbf{color} = \mathbf{Gray}$

$x.\mathbf{d} = u.\mathbf{d} + 1$

$x.\boldsymbol{\pi} = u$

$\text{ENQUEUE}(Q,\ x)$

else if $\text{u} \in \text{T}$:

if u **is** $M-$unsatuated:

return $\mathbf{PATH}(\text{u} \to \text{v})$

goto End

else:

$\boldsymbol{J} = \{x \in B.\text{ADJ}[u] \mid (\text{u},\ \text{x})$ **is** M_{edge} **and** x **is White**$\}$

if $\boldsymbol{J} = \emptyset$:

continue

for each $x \in \boldsymbol{J}$:

$x.\mathbf{color} = \mathbf{Gray}$

$x.\mathbf{d} = u.\mathbf{d} + 1$

$x.\boldsymbol{\pi} = u$

$\text{ENQUEUE}(Q,\ x)$

$x.\mathbf{color} = \mathbf{Black}$

根据这个算法，可以很快写出图的极大连通分支搜索算法SUB-GRAPH(G)、图的连通性判断算法IS$-$CONNECTED(G)等。

五、 程序代码

```c
//
//  bfs.c
//  Algorithms - Graph breadth-first search
//
//  Created by YourtionGuo on 08/05/2017.
//  Copyright © 2017 Yourtion. All rights reserved.
//

#include <stdlib.h>

#include "bfs.h"
#include "graph.h"
#include "list.h"
#include "queue.h"

#pragma mark - Public

int bfs(Graph *graph, BfsVertex *start, List *hops) {
    Queue       queue;
    AdjList     *adjlist, *clr_adjlist;
    BfsVertex   *clr_vertex, *adj_vertex;
    ListElmt    *element, *member;

    /// 初始化图的所有顶点

    for (element = list_head(&graph_adjlists(graph)); element != NULL; element = list_next(element)) {

        clr_vertex = ((AdjList *)list_data(element))->vertex;

        if (graph->match(clr_vertex, start)) {

        /// 初始化起点
        clr_vertex->color = gray;
        clr_vertex->hops = 0;

    } else {

            /// 初始化其他顶点
            clr_vertex->color = white;
            clr_vertex->hops = -1;

        }
    }

    /// 根据起点邻接表初始化队列

    queue_init(&queue, NULL);

```

```
49      if (graph_adjlist(graph, start, &clr_adjlist) != 0) {
50
51          queue_destroy(&queue);
52          return -1;
53      }
54
55      if (queue_enqueue(&queue, clr_adjlist) != 0) {
56
57          queue_destroy(&queue);
58          return -1;
59      }
60
61      /// 执行广度优先搜索
62
63      while (queue_size(&queue) > 0) {
64
65          adjlist = queue_peek(&queue);
66
67          /// 在当前邻接表遍历每个顶点
68
69          for (member = list_head(&adjlist->adjacent); member != NULL; member = list_next(member))
    {
70
71              adj_vertex = list_data(member);
72
73              /// 确定下个邻接顶点的颜色
74
75              if (graph_adjlist(graph, adj_vertex, &clr_adjlist) != 0) {
76
77                  queue_destroy(&queue);
78                  return -1;
79              }
80
81              clr_vertex = clr_adjlist->vertex;
82
83              /// 将白顶点着色为灰色并将邻接表入队
84
85              if (clr_vertex->color == white) {
86
87                  clr_vertex->color = gray;
88                  clr_vertex->hops = ((BfsVertex *)adjlist->vertex)->hops + 1;
89
90                  if (queue_enqueue(&queue, clr_adjlist) != 0) {
91
92                      queue_destroy(&queue);
93                      return -1;
94                  }
95              }
96          }
```

```
97
98          /// 将当前邻接表出队并将其着色为黑色
99
100         if (queue_dequeue(&queue, (void **)&adjlist) == 0) {
101
102             ((BfsVertex *)adjlist->vertex)->color = black;
103
104         } else {
105
106             queue_destroy(&queue);
107             return -1;
108         }
109     }
110
111     queue_destroy(&queue);
112
113     /// 回传每个顶点的跳数到表中
114
115     list_init(hops, NULL);
116
117     for (element = list_head(&graph_adjlists(graph)); element != NULL; element = list_next(ele-
    ment)) {
118
119         /// 去掉跳数为 -1 的（不可达）
120
121         clr_vertex = ((AdjList *)list_data(element))->vertex;
122
123         if (clr_vertex->hops != -1) {
124
125             if (list_ins_next(hops, list_tail(hops), clr_vertex) != 0) {
126
127                 list_destroy(hops);
128                 return -1;
129             }
130         }
131     }
132
133     return 0;
134 }
```

六、 参考文献

[1] **林锐**. 高质量 C++/C 编程指南 [M]. 1.0 ed., 2001.

[2] 算法精解：C 语言描述：https://github.com/yourtion/LearningMasteringAlgorithms-C