数据结构与算法

Data Structure and Algorithm

XVII. 图论 I

授课人：Kevin Feng

翻译　：梁少华

Review

回顾

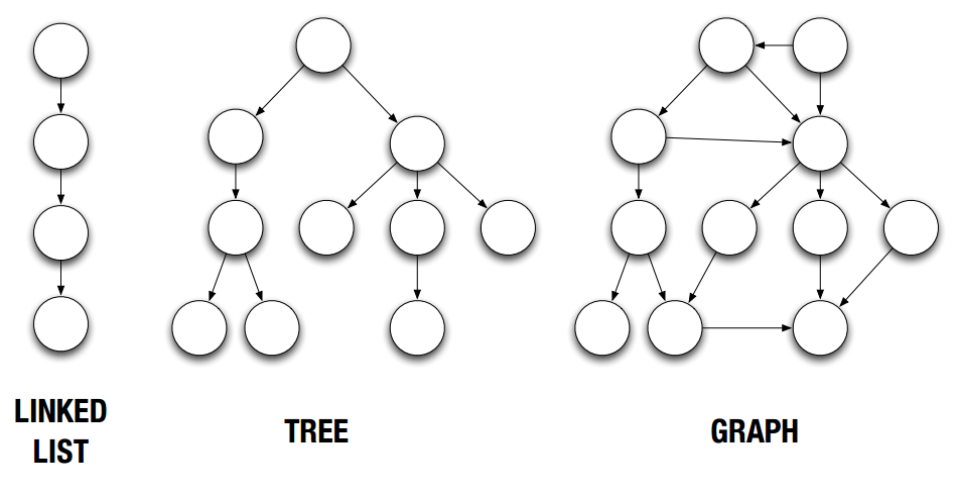
- 数据结构与算法
- 数学回顾
- 数组
- 数组列表
- 搜索和排列
- 递归与迭代
- 二进制搜索
- 分而治之
- 链接列表
- 散列表
- 树
- 堆

概述

- 树 vs. 图论
- 图形术语
- 图形实现
- DFS(深度优先搜索) vs. BFS(广度优先搜索)

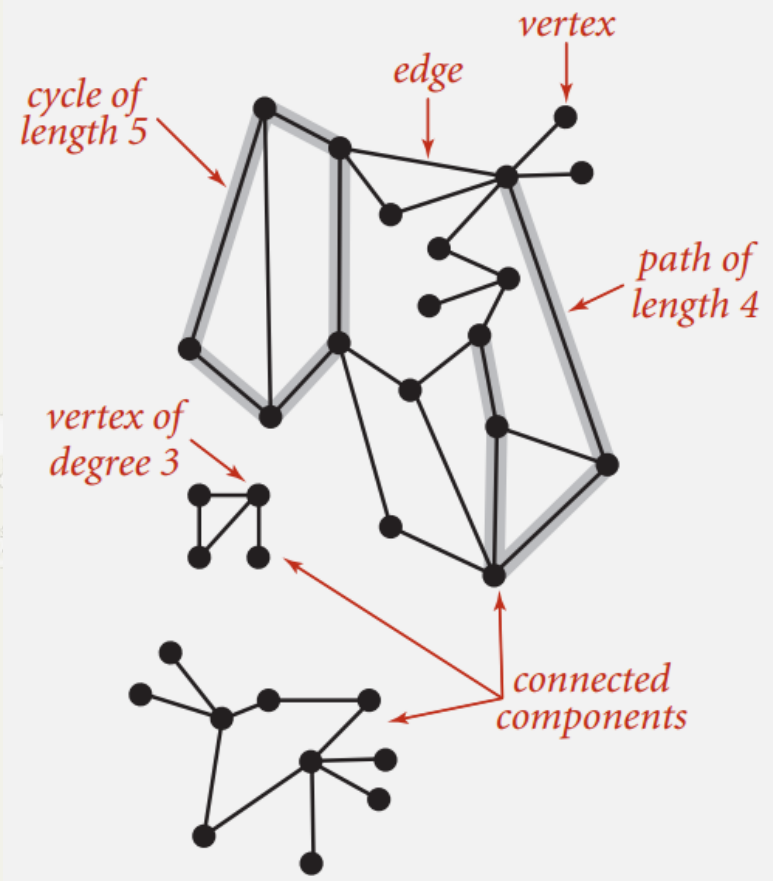
回顾-树

- 树形结构的局限性
  - 树中的项目只能出现在一个位置
  - 一个项目只有一个前驱（父节点）,除非它是根（无）
  - 唯一可用的链接是父/子链接



图形术语

- *图 G* 是*顶点V* 和*边 E*的集合
- 两个*顶点*之间：*边*
  - 如果*顶点 x* 和 *y* 共享边,则它们相邻,或者它们是相邻的
- *无向图*
  - 无向图中的一个边可以在任一方向上遍历
- *路径*: 通过边连接的顶点序列
- *周期*: 第一个和最后一个顶点相同的路径
- *入度*: 顶点的度数V是以V为端点的边数
- *出度*: 顶点的出度v是以v为起点的边的数量
- *度* 顶点的度数是其入度和出度的总和



图形问题

Problem	Description
s-t path	Is there a path between *s* and *t*?
shortest s-t path	What is the shortest path between *s* and *t*?
cycle	Is there a cycle in the graph?
Euler cycle	Is there a cycle that uses each edge exactly once?
Hamilton cycle	Is there a cycle that uses each vertex exactly once?
connectivity	Is there a way to connect all of the vertices?
biconnectivity	Is there a vertex whose removal disconnects the graph?
planarity	Can the graph be drawn in the plane with no crossing edge?
graph isomorphism	Do two adjacency lists represent the same graph?
3-coloring	Is there a way to color a graph using 3 colors, such that no two adjacent vertices share the same color?
Travelling salesman problem	*Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?*

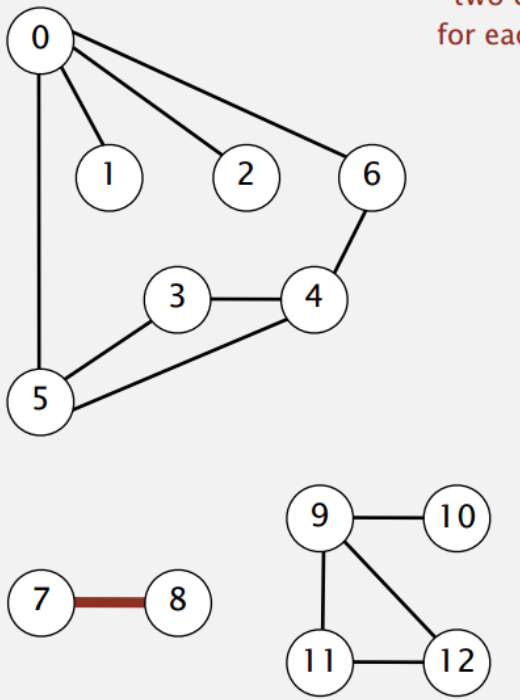
图形 ADT

- 数据成员
  - 顶点
  - 边缘
- 操作
  - 有多少顶点?
  - 有多少个边缘?
  - 添加一个新的顶点
  - 添加一个新的边缘
  - 获取所有邻居？（进出）
  - U，V连接吗?
  - 反转所有边缘？
  - 获取2跳邻居

邻接矩阵图表示法

Maintain a two-dimensional V-by-V boolean array;

for each edge v–w in graph: `adj[v][w] = adj[w][v] = true`.



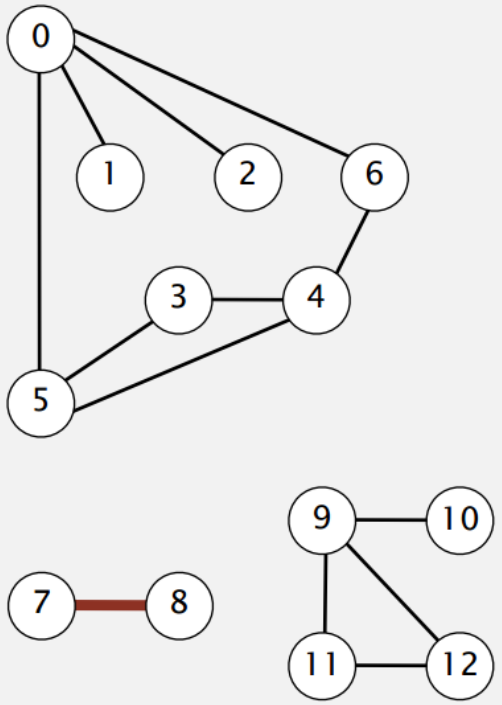
two entries for each edge

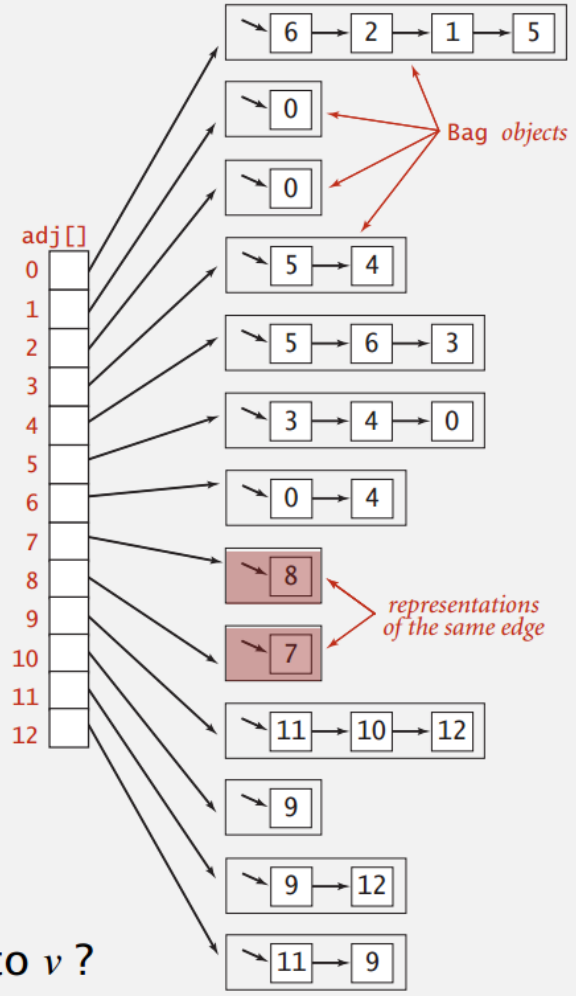
	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	1	0	0	1	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	0	0	0	0	0	0	0
4	0	0	0	1	0	1	1	0	0	0	0	0	0
5	1	0	0	1	1	0	0	0	0	0	0	0	0
6	1	0	0	0	1	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	1	0	0	0	0
8	0	0	0	0	0	0	0	1	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	1	1	1
10	0	0	0	0	0	0	0	0	0	1	0	0	0
11	0	0	0	0	0	0	0	0	0	1	0	0	1
12	0	0	0	0	0	0	0	0	0	1	0	1	0

Q. How long to iterate over vertices adjacent to v ?

邻接列表图表示

Maintain vertex-indexed array of lists.





Q. How long to iterate over vertices adjacent to v ?

比较

- 在实践中，使用邻接列表表示
  - 真实世界的图倾向于稀疏

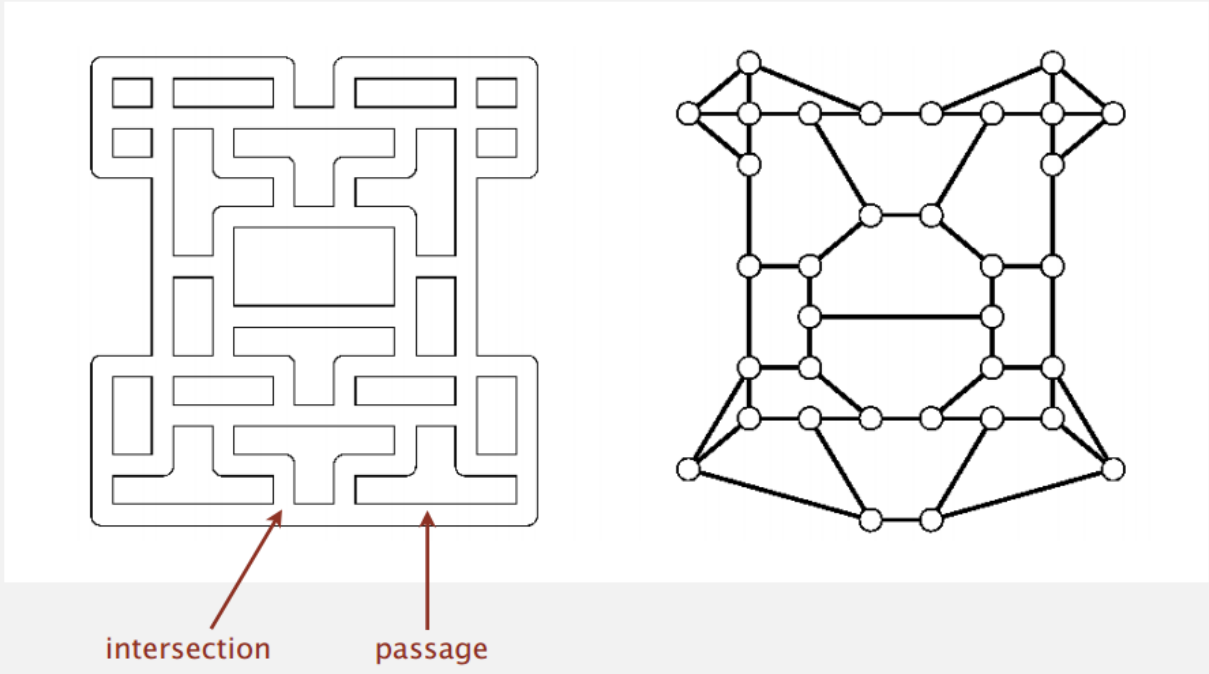
Representation	Space	Add edge	Edge between v and w	Iterate over vertices adjacent to v
Adjacency matrix	V^2	1*	1	*V*
Adjacency list	*E+V*	1	*degree(v)*	*degree(v)*

* 不支持平行边

迷宫

Maze graph.

- Vertex = intersection.
- Edge = passage.



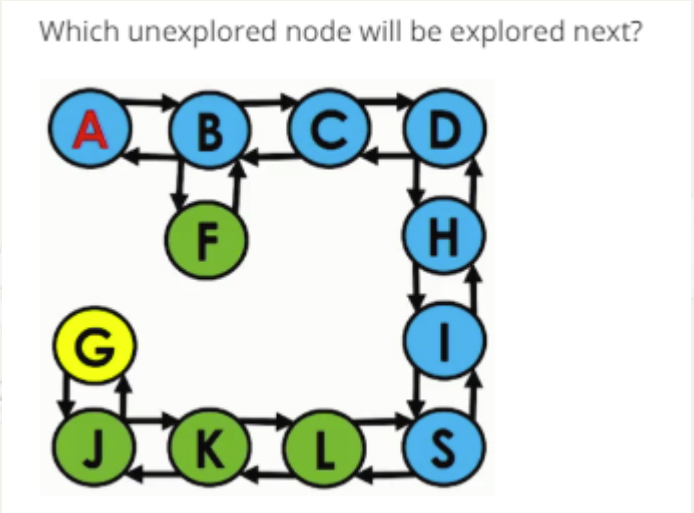
深度优先搜索

- 目标：遍历一个图
- 想法：模仿迷宫探索

DFS (to visit a vertex v)

Mark v as visited.

Recursively visit all unmarked vertices w adjacent to v.



https://www.cs.usfca.edu/~galles/visualization/DFS.html

DFS(深度优先搜索)算法

- 如何跟踪下一步搜索的位置?
  - *Stack*: 列表中只从一端添加和移除:
    - Push：添加元素
    - Pop：删除一个元素
- 如何跟踪访问过的内容?
  - *HashSet*: 常量添加，删除和搜索
- 如何跟踪从开始到目标的路径?
  - *HashMap*:将每个节点链接到发现它的节点

DFS: Algorithm

```
DFS(S, G):
  Initialize: stack, visited HashSet and parent HashMap
  Push S onto the stack and add to visited
  while stack is not empty:
    pop node curr from top of stack
    if curr == G return parent map
    for each of curr's neighbors, n, not in visited set:
      add n to visited set
      add curr as n's parent in parent map
      push n onto the stack
  // If we get here then there's no path
```

DFS: Algorithm (recursive)

```
DFS(S, G, visited, parents):
  if S == G return;
  for each of S's neighbors, n, not in visited set:
    add n to visited set
    add S as n's parent in parents map
    DFS(n, G, visited, parents)
```

BFS (广度优先搜索)算法

- 如何跟踪下一步搜索的位置?
  - *Queue*:列出你只从一端添加和移除的地方
    - enqueue: 添加一个元素
    - deque: 删除一个元素
- 如何跟踪访问过的内容?
  - *HashSet*:定时添加，删除和搜索
- 如何跟踪从开始到目标的路径?
  - *HashMap*: 将每个节点链接到发现它的节点

BFS: Algorithm

```
BFS(S, G):
  Initialize: queue, visited HashSet and parent HashMap
  Enqueue S onto the queue and add to visited
  while queue is not empty:
    dequeue node curr from front of queue
    if curr == G return parent map
    for each of curr's neighbors, n, not in visited set:
      add n to visited set
      add curr as n's parent in parent map
      enqueue n onto the queue
```

https://www.cs.usfca.edu/~galles/visualization/BFS.html

性能

```
DFS(S, G):
   Initialization of structures
   Push S on stack and add to visited
   while stack is not empty:
      pop node curr from top of stack
      if curr == G return parent map
      for each of curr's unvisited neighbors, n:
         add n to visited set
         add curr as n's parent in parent map
         push n to top of stack
   // If we get here then there's no path
```

Depth first search

```
BFS(S, G):
   Initialization of structures
   Enqueue S in queue and add to visited
   while queue is not empty:
      dequeue node curr from front of queue
      if curr == G return parent map
      for each of curr's unvisited neighbors, n:
         add n to visited set
         add curr as n's parent in parent map
         enqueue n to back of queue
   // If we get here then there's no path
```

Breadth first search

- 在最坏的情况下我们必须访问多少个顶点?
  - $|V| - 1 = O(|V|)$
- 我们还通过边缘访问每个顶点的邻居
- 在最坏的情况下我们必须穿越多少条边?
  - $O(|E|)$
- DFS和BFS的性能:
  - $O(|V|) + O(|E|)$

练习1

- 迷宫
  - 迷宫中有一个空的空间和墙壁。 球可以通过向上，向下，向左或向右滚动经过空的空间.
  - 考虑到球的起始位置，目的地和迷宫，确定球是否可以停在目的地.
  - 迷宫由二进制2D阵列表示。1意味着墙，0意味着空的空间。你可以假设迷宫的边界都是墙。开始和目标坐标由行索引和列索引表示.

```
Input 1: a maze represented by a 2D array

0 0 1 0 0
0 0 0 0 0
0 0 0 1 0
1 1 0 1 1
0 0 0 0 0

Input 2: start coordinate (rowStart, colStart) = (0, 4)
Input 3: destination coordinate (rowDest, colDest) = (4, 4)

Output: true
```

练习2

迷宫 II

- 迷宫中有一个空荡荡的空间和墙壁。球可以通过向上、向下、向左或向右滚动穿过空格. 但它不会停止滚动直到撞到墙上.当球停下来时，它可以选择下一个方向.
- 给定球的起始位置、目的地和迷宫，确定球是否可以在目的地停止.
- 迷宫由二进制2D阵列表示。1意味着墙，0意味着空的空间。你可以假设迷宫的边界都是墙。开始和目标坐标由行索引和列索引表示.

```
Input 1: a maze represented by a 2D array

0 0 1 0 0
0 0 0 0 0
0 0 0 1 0
1 1 0 1 1
0 0 0 0 0

Input 2: start coordinate (rowStart, colStart) = (0, 4)
Input 3: destination coordinate (rowDest, colDest) = (4, 4)

Output: true
Explanation: One possible way is : left -> down -> left -> down -> right -> down -> right.
```


```
Input 1: a maze represented by a 2D array

0 0 1 0 0
0 0 0 0 0
0 0 0 1 0
1 1 0 1 1
0 0 0 0 0

Input 2: start coordinate (rowStart, colStart) = (0, 4)
Input 3: destination coordinate (rowDest, colDest) = (3, 2)

Output: false
Explanation: There is no way for the ball to stop at the destination.
```

练习3

迷宫 III

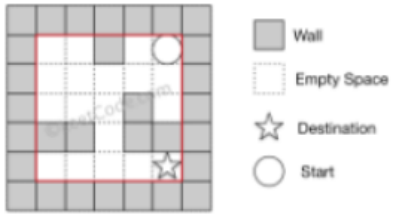
- 迷宫中有一个空荡荡的空间和墙壁。球可以通过向上、向下、向左或向右滚动穿过空旷的空间，但它不会停止滚动直到撞到墙上。当球停下来时，它可以选择下一个方向.
- 给定球的起始位置，目的地和迷宫,找到球在目的地停留的最短距离.距离由球从起始位置（排除）到目的地（包括）所经过的空白空间的数量定义。 如果球不能停在目的地，返回-1.

```
Input 1: a maze represented by a 2D array

0 0 1 0 0
0 0 0 0 0
0 0 0 1 0
1 1 0 1 1
0 0 0 0 0

Input 2: start coordinate (rowStart, colStart) = (0, 4)
Input 3: destination coordinate (rowDest, colDest) = (4, 4)

Output: 12
Explanation: One shortest way is : left -> down -> left -> down -> right -> down -> right.
             The total distance is 1 + 1 + 3 + 1 + 2 + 2 + 2 = 12.
```



练习4

迷宫 III

- 球可以通过卷起(u),向下(d),向左(l)或向右(r)穿过空的空间, 但它不会停止滚动,直到撞墙. 当球停止时,它可以选择下一个方向. 这个迷宫里也有一个洞. 如果滚到洞上,球会落入洞中.
- 给定球的位置,洞位置和迷宫, 找出球如何通过移动最短距离而落入洞中. 距离由球从开始位置(不包括)到洞(包含)所经过的空白空间的数量定义. 使用'u'，'d'，'l'和'r'输出移动方向.由于可能有几种不同的最短路径,因此应该输出字典顺序最小的方式. 如果球不能到达洞,输出“不可能”.

数据结构与算法

Data Structure and Algorithm

XVII. 图论I结束

授课人：Kevin Feng

翻译　：梁少华