Reversible Graph Algorithms

1. 研究分野

可逆コンピューティング
ー可逆アルゴリズム

2. 目的

時間・空間計算量の観点で効率の良い可逆アルゴリズムを提案する
(UNIVERSITY OF COPENHAGEN Thesis)

3. 背景

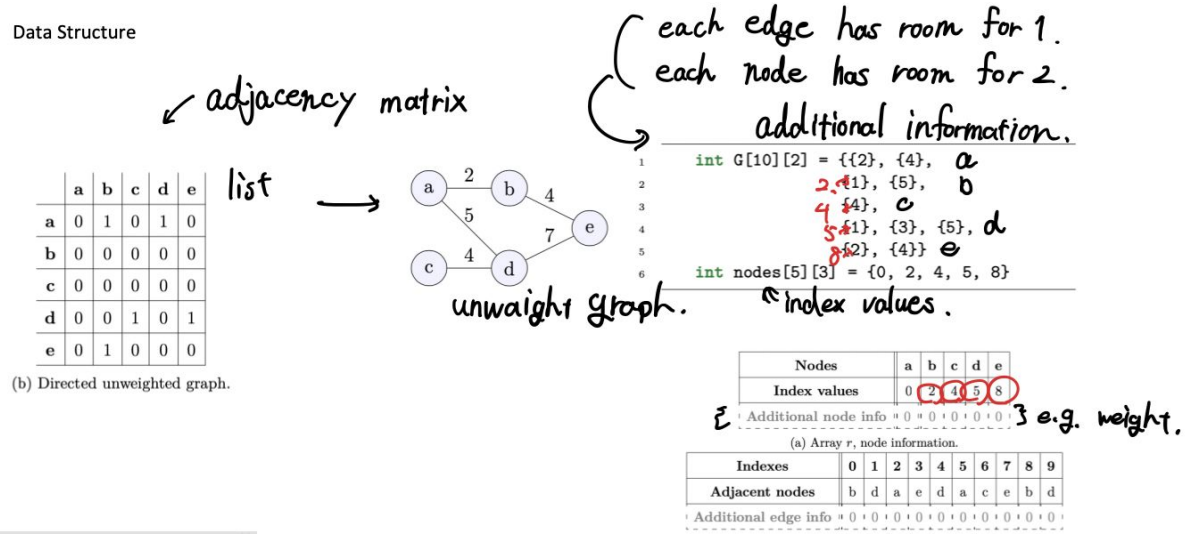
引き続き調査

4. アプローチ

・**Janusにおいて効率良くグラフを表すデータ構造**
(左) adjacency matrix：行列で表す方法(無駄が多い)
(右) adjacency list：matrixに比べて無駄がない

adjacency list：ノードを表すnodes[ ]とエッジを表すG[ ]から成り立つ
ーnodes：Index Valueを持つ(Gにおいてnodeのエッジを指す部分がどこからか)
ーG: グラフのエッジ全ての情報を持つ

＊adjacency list の情報を拡張することが可能(重みをつけるなど)
e.g. G[10][2] = {...}, {この部分の情報を自由に追加可能}



・可逆深さ優先探索

ー時間計算量・空間計算量：非可逆な可逆深さ優先探索と同じ

ーゴミ出力：入力以外のゴミ出力なし(配列に開始・終了時刻を格納する場所が必要)

Depths first search

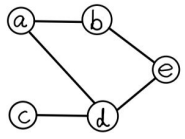
```
1  procedure DFS(int G[], int nodes[])
2      local int time = 0, int i = 0
3
4      // We run through each individual node and check if they have been visited
5      // (timestamped). If they haven't we run DFSvisit.
6      from i = 0 do
7          if nodes[i][1] = 0 then
8              call DFSvisit(G, nodes, i, time)
9          fi nodes[i][2] = time
10         i += 1
11     until i = size(nodes)
12
13     delocal int time = size(nodes) * 2, int i = size(nodes)
```

```
1  procedure DFSvisit(int G[], int nodes[], int u, int time)
2      // The time is incremented and the node gets entry-timestamped
3      time += 1
4      nodes[u][1] += time ← entry - timestamp
5
6      // end contains a pointer to the end of adjacency list
7      local int end = 0
8      if u = size(nodes) - 1 then
9          // If we try to get the last list we stop at the end of the element list
10         end += size(G)
11     else
12         // Or else, we stop where the next list starts
13         end += nodes[u + 1][0]
14     fi u = size(nodes) - 1
15
16     // The adjacent nodes are checked and if they have not been visited,
17     // DFSvisit is called with the unvisited node
18     local int r = nodes[u][0] // Pointer to the current node
19     from r = nodes[u][0] do
20         if nodes[G[r]-1][1] = 0 then // The adjacent node's first time stamp is 0
21             call DFSvisit(G, nodes, G[r]-1, time)
22         fi nodes[G[r]-1][2] = time // The adjacent node's second time matches the current time
23         r += 1
24     until r = end
25     delocal int r = end
26
27     // The time is incremented and the node gets exit-timestamped
28     time += 1
29     nodes[u][2] += time ← exit - timestamp.
30
31     // end is delocated
32     if u = size(nodes) - 1 then
33         end -= size(G)
34     else
35         end -= nodes[u + 1][0]
36     fi u = size(nodes) - 1
37     delocal int end = 0
```

e.g. if u = 2 (c) then end = 5 because. indexValue d=5

DFS.
like irreversible. DFS.
(recursive)
↳ O(E)

(1)



i = 0 , time = 0

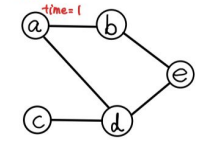
nodes[0][1] = 0 ← nodes[0] = a

↳ start-timestamp = 1 ← in DFSvisit.

nodes[2][1] = 0 ↳ nodes[2] = b

→ to b.

(2)

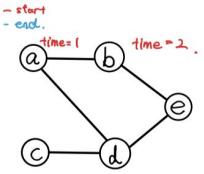


i = 0 , time = 1

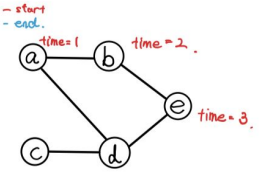
nodes[2][1] = 0 ← nodes[2] = b.

start-timestamp = 2 ← in DFSvisit.

(3)

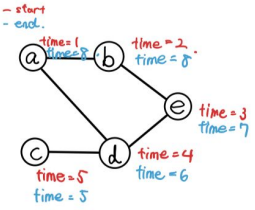


(4)



⋮

(5)



5. 結果

6. 有用性

7. 限界・短所

8. 次に何を読めばいいか？