MC558 - Análise de Algoritmos

João Henrique Dalben

September 24, 2016

Lista 1

22.1-1

Given an adjacency-list representation of a directed graph, how long does it take to compute the out-degree of every vertex? How long does it take to compute the in-degrees?

```
OUT-DEGREE(G)
  for each vertex v in G.V
    v.out_degree = 0
    for each vertex u in G.Adj[v]
      v.out_degree++
```

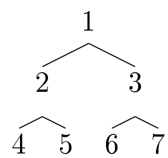
```
IN-DEGREE(G)
  for each vertex v in G.V
    v.in_degree = 0
    for each vertex u in G.Adj[v]
      u.in_degree++
```

According to the book, "if G is a directed graph, the sum of the lengths of all the adjancecy lists is $|E|$". Therefore, as both OUT-DEGREE and IN-DEGREE algorithms visit every vertex and every entry of every adjacency list once, their complexity is $\Theta(V + E)$.

22.1-2

Give an adjacency-list representation for a complete binary tree on 7 vertices. Give an equivalent adjacency-matrix representation. Assume that vertices are numbered from 1 to 7 as in a binary heap.

Tree representation:



Adjacency-matrix representation:

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

22.1-3

The transpose of a directed graph $G = (V, E)$ is the graph $G^T = (V, E^T)$, where $E^T = \{(v, u) \in V \times V : (u, v) \in E\}$. Thus, G^T is G with all its edges reversed. Describe efficient algorithms for computing G^T from G, for both the adjacency-list and adjacency-matrix representations of G. Analyze the running times of your algorithms.

```
TRANSPOSE-LIST(G)
  G_t.V = G.V
  for each vertex v in G.V
    for each vertex u in G.Adj[v]
      G_t.Adj[u].append(v)
  return G_t
```

```
1 TRANSPOSE-MATRIX(G)
2   G_t = G
3   for i from 1 to |G.V|
4     for j from i+1 to |G.V|
5       swap(G_t[i][j], G_t[j][i])
```

In TRANSPOSE-LIST, the complexity analysis is the same as exercise 22.1-1, which is $O(V + E)$.

In TRANSPOSE-MATRIX, the nested loop between lines 3 and 5 has $\sum_{n=1}^{V} V - i$ (consider V = |V|) iterations, and its complexity is

$$\sum_{n=1}^{V} V - i = \sum_{n=1}^{V} V - \sum_{n=1}^{V} i = V^2 - \frac{V(V+1)}{2} = \\ V^2 - \frac{V^2}{2} - \frac{V}{2} = \frac{V^2}{2} - \frac{V}{2} = \frac{V(V-1)}{2} = \Theta(V^2) \tag{1}$$

22.1-4

Given an adjacency-list representation of a multigraph $G = (V, E)$, describe an $O(V + E)$-time algorithm to compute the adjacency-list representation of the "equivalent" undirected graph $G' = (V, E')$, where E' consists of the edges in E with all multiple edges between two vertices replaced by a single edge and with all self-loops removed.

```
EQUIVALENT(G)
  G'.V = G.V
  for each vertex v in G.V
    for each vertex u in G.Adj[v]
      if (u != v) and (u not in G'.Adj[v])
        G'.Adj[v].append(u)
        G'.Adj[u].append(v)
  return G'
```

22.1-5

The square of a directed graph $G = (V, E)$ is the graph $G^2 = (V, E^2)$ such that $(u, v) \in E^2$ if and only G contains a path with at most two edges between u and v. Describe efficient algorithms for computing G^2 from G for both the adjacency-list and adjacency-matrix representations of G. Analyze the running times of your algorithms.

```
SQUARE-MATRIX(G)
  G' = G
  for i from 1 to |G.V|
    for j from 1 to |G.V|
      G'[i][j] = 0
      for k from 1 to |G.V|
        G'[i][j] = G[i][k]*G[k][j]
  return G'
```

```
SQUARE-LIST(G)
  G'.V = G.V
  for each vertex v in G.V
    for each vertex u in G.Adj[v]
```

```
        G'.Adj[v].append(G.Adj[u])
  for each vertex v in G.V
      COUNTING-SORT(G'.Adj[v])
      REMOVE-DUPLICATES(G'.Adj[v])
  return G'
```

xx' SQUARE-MATRIX has complexity $O(V^3)$, while SQUARE-LIST has complexity $O(VE)$.

22.1-6