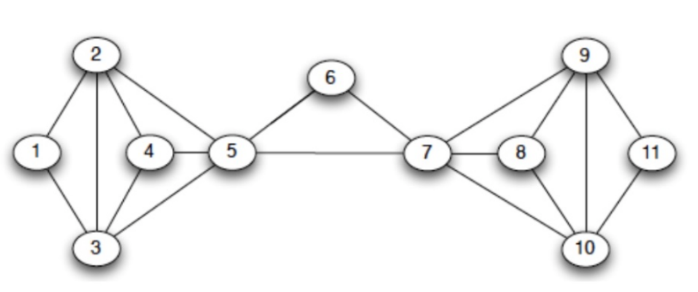
(1)



Start with the lowest degree node.

Remove the node with the lowest degree from the graph, along with its incident edges.

Update the degrees of the remaining nodes.

Assign the removed node a core number equal to its degree just before it was removed.

Repeat steps 2-4 until all nodes have been removed and assigned a core number.

|  |  |  |  |
| --- | --- | --- | --- |
| 1: 2 (2,3)  2: 4 (1,3,4,5)  3: 4 (1,2,4,5)  4: 3 (2,3,5)  5: 5 (2,3,4,6,7)  6: 2 (5,7)  7: 5 (5,6,8,9,10)  8: 3 (7,9,10)  9: 4 (7,9,10,11)  10:4 (7,8,9,11)  11:2 (9,10) | 2: 3 (3,4,5)  3: 3 (2,4,5)  4: 3 (2,3,5)  5: 4 (2,3,4,7)  7: 4 (5,8,9,10)  8: 3 (7,9,10)  9: 3 (7,9,10)  10:3 (7,8,9) | 5: 1 (7)  7: 1 (5) |  |

K=1, remove all nodes with degree less than 1, no nodes removed.

K=2, remove all nodes with degree less than 2, no nodes removed.

K=3, remove all nodes with degree less than 3

Node 1, 6 11 (degree=2) removed

Updating the graph:

K=4, remove all nodes with degree less than 4

Node 2, 3, 4, 9, 10 (degree=3) removed

After updating the graph, Node 5, 7(degree=3) removed

Node 1: 2-core

Node 2: 3-core

Node 3: 3-core

Node 4: 3-core

Node 5: 3-core

Node 6: 2-core

Node 7: 3-core-

Node 8: 3-core

Node 9: 3-core

Node 10: 3-core

Node 11: 2-core

Algorithm:

1. Start from k=1

2. Remove all nodes with degree no greater than k and adjust degrees of neighbor nodes of removed nodes

3. Set core number of each removed node as k

4. k=k+1

5. If the graph still has nodes, go to 2

6. If the graph is empty, terminate

Using the Linked-List data structure described in lecture slides, we can always maintain all nodes sorted based on their degrees. Every time when we need to reduce the degree of one node by 1, the time cost is only O(1). Thus, using the Linked-List Data structure, the total time complexity is O(m+n), where m is the number of edges and n is the number of nodes.