Maximal (Weighted) Independent Set

In an undirected graph $G = (V, E)$, where V is the vertex set and E is the edge set, $S \subseteq V$ is an independent set if no vertices in S are adjacent to one another. An independent set that has the maximum cardinality is called *maximum independent set*. If each vertex is associated with a weight (a positive real number), then S is a *maximum weighted independent set* if it is an independent set that has the maximum total weight among all such sets. Clearly, a maximum independent set is a maximum weighted independent set with uniform weight. Finding either set is known to be NP-hard, for which polynomial-time approximation is NP-hard as well.

There exist many heuristics for these two problems. One well-known greedy approach works by selecting a vertex into the set, removing it and adjacent vertices from the graph, and repeating this process on the remaining graph. The result found by the greedy approach can only be a *maximal* independent set (MIS), an independent set of which no proper superset is also an independent set. If nodes are associated with weights, then the result is a *maximal* weighted independent set (MWIS).

Existing greedy approaches to identifying an MIS/MWIS are mostly centralized. Let G denote a graph and I the MIS/MWIS to be constructed. A generic approach to the construction of I from G works in the following way: (here $N(v)$ denotes the set of v's neighbors.)

$I := \emptyset$;
while $G \neq \emptyset$ **do**
  select v from G;
  $I := I \cup \{v\}$;
  $G := G \setminus \{v\} \cup N(v)$;
end

Different approaches differ in the selection of v from G. Some approach selects the node with the minimum node degree among all when forming an MIS. One selection rule for MWIS termed GWMIN prioritizes node v that has the maximal $W(v)$ / (deg(v) + 1) value among all candidates, where $W(v)$ is the weight associated with v and deg(v) is the node degree of v.

Exercise (a)

Write a centralized program to implement GWMIN, which finds an MWIS from a given graph. The graph data should be read from a file with the following format:

1. The first line is an integer n that represents the number of nodes in the graph. All nodes are numbered 0, 1, 2, …, n – 1.
2. The second line lists the set of all weights, one for each node. All weights are assumed to be unsigned integers, so there are total n integers in the second line.

3. The third line contains the values of the adjacency vector ($a_{0,0}$, $a_{0,1}$, …, $a_{0,j}$, …, $a_{0,n-1}$) for node 0, where $a_{0,j} = 1$ if nodes 0 and j are neighbors and $a_{0,j} = 0$ otherwise ($0 \le j \le n - 1$). We assume that a node is not a neighbor of itself.
4. All other lines contain the values of all the other adjacency vectors, one for each line. In general, the (i + 3)-th line contains the values of the adjacency vector ($a_{i,0}$, $a_{i,1}$, …, $a_{i,j}$, …, $a_{i,n-1}$) for node i, where $a_{i,j} = 1$ if nodes i and j are neighbors and $a_{i,j} = 0$ otherwise.

Your program should display the ids of all nodes in the MWIS set and the total weight.

Please note that if there are more than two nodes that have the same $W(v) / (\deg(v) + 1)$ value, your program selects the one that has the smallest node id. Also note that when you program adds a node into the MWIS and removes it from G, your program should update node degrees of all its neighboring nodes remaining in G.

The TA will provide you an example file for you own test. The TA will use other test files to validate the correctness of your program.

Exercise (b)

Rewrite the centralized program in Exercise (a) into a distributed algorithm and simulate its execution to output the ids of all nodes in the MWIS set and the total weight. By “simulation” I mean your program is like a god that oversees and conducts the executions of all processes. Only one program is in execution without the need of sub-processes or threads to “emulate” processes. Here we assume virtually synchronous executions (Chapter 1), where program executions are synchronized in a round basis. All nodes (i.e., processes) execute the following three steps asynchronously within each round.

1. Decide whether it should be in the MWIS or not.
2. Send its decision to each neighboring node (assuming asynchronous send primitive)
3. Receive each neighbor’s decision (assuming non-blocking receive primitive)

Although all processes execute these three steps asynchronously, you don’t have to actually simulate this kind of asynchronous execution. It is safe to let one process execute all the three steps before the next process starting execution in your simulation. The reason is that the execution of one process in a round does not affect the execution of another process in the same round. The decision of whether to join the MWIS or not made in the current round is 1) based on the decisions of other processes received in the previous round and 2) does not affect the decision of any process made in the current round. The decision however may affect the decision of another process in the next

round (if any).

All processes are synchronized at the end of each round. That is, all processes proceed to the next round after and only after each process receives a decision from each of its neighbors in this round. However, because messages never get lost (each message sent is eventually received by the receiver) and we are not interested in measuring message delay, you don't have to simulate actual message sending and receiving events. That is, you don't have to simulated the last two steps. Just keep the semantics of the last two steps in a round that each process knows the decisions of all its neighboring nodes made in this round before proceed to the next round.

About the decision of whether to join the MWIS, each process v should compare its $W(v) / (\deg(v) + 1)$ value with those of its neighbors to make the decision. Remember uses process id to break ties. A process may revise its decision as needed and the simulation ends when no process has ever made a change in the last round.

Hint: use loops to iteratively simulate decision making by each process in each round.