CS6023: GPU Assignment - 3

Deadline: 2nd April, 2023 , 11:55 PM on moodle

/***/

Activation Game

1. Problem statement

Consider a graph G with |V| vertices and |E| edges.
Nodes are arranged in level order hierarchy. Starting from level L=0,1,2…..
And vertices are numbered from 0 to |V|-1.

Node 0 to (V_0 - 1) are at level 0.

$Node\ V_0$ to ($V_1 - 1$) are at level 1.

……

……

Node V_{L-2} to (V_{L-1} - 1) at the last level.

$V_0, V_1, \ldots V_{L-1}\ are\ unknown.$

Each edge present in the graph is going from a node present in level K to some node present in level (K+1). All edges are directed.

Each node present in the graph has some activation point requirement (APR).
For node v activation point requirement is APR[v].
APR[v]>0 for all vertices except the vertices present at level zero. Vertices present at level zero will have APR[V] = 0.

Define active in-degree (AID) of vertex v as the number of edges coming from active nodes to vertex v.
There are two rules in the game.

For a vertex v,

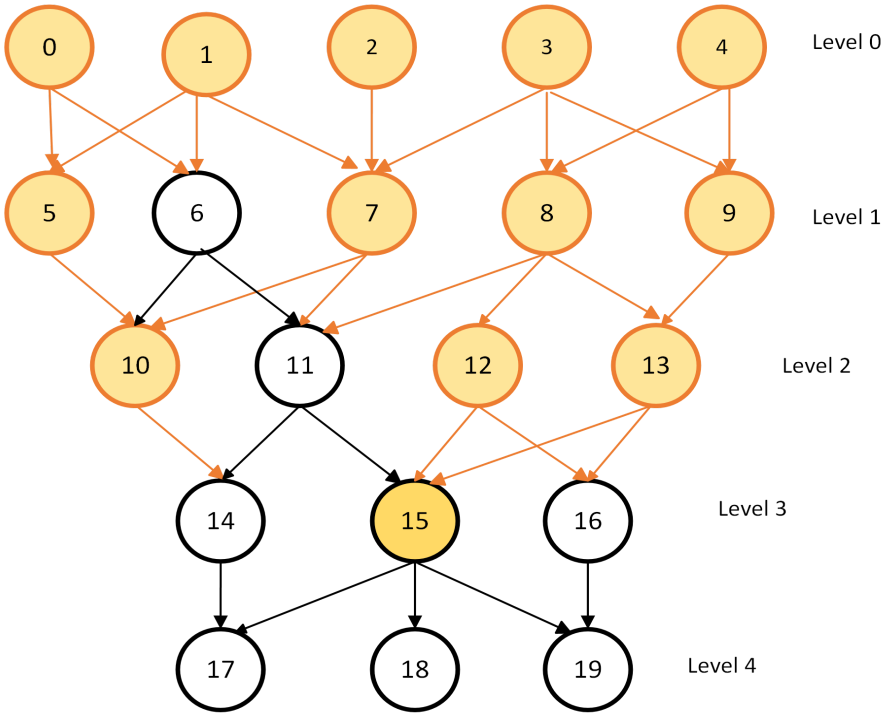
1. Activation rule: If AID(v) >= APR(v) then vertex v will get activated.
2. Deactivation rule: If vertices (v-1) and (v+1) are inactive then vertex v will become inactive if all the three vertices (v-1), v , (v+1) are on the same level. Consider only if both v-1 and v+1 exist.

Initially nodes present at level 0 will be active as their APR is zero.
In output print the number of active nodes in each level after processing starting from level zero.

Note: In each level of the graph, there can be at max 10000 vertices and the maximum number of levels in the graph is 1000.
So,
|V| <= 10M, |E|<= 100 M

2. Input:
First line of input contains 3 integers L (number of levels in the graph), V (number of vertices in the graph) and E (total number of edges in the graph)
Next E lines will contain 2 positive integers source and destination of each edge of the graph.
Next V lines contain the active point requirement of vertices.

3. **Example**
// active point required for each node starting from node 0.
apr[] = [0, 0, 0, 0, 0, 2, 4, 2, 2, 2, 2, 3, 1, 2, 2, 2, 3, 1, 1, 1]



All the colored nodes are active nodes except node 15 which gets deactivated because of the second rule.

Output: 5 4 3 0 0

5. Point to be noted:

1. In the code we have taken care of input. We have generated csr array, offset array and apr[] array for you. You have to work with them.
2. Use the main.cu file present in the code folder. Rename it as your rollnumber.cu.
3. Make sure to use comments to make your code more understandable.
4. Do not write any print statements inside the kernel.
5. Write your code in mentioned area only
6. Use the evaluation script provided by us to test your code.
7. Create one folder inside the SUBMIT folder with your roll number and place your rollnumber.cu file there.
8. if your code's performance (total time on all the testcases) is < average / 2, you will get one bonus mark. If it is > 2 * average, you will lose two marks. Average is computed using each submitted code's total time on all the testcases.
9. For more on CSR (refer to this link)

7. Submission guideline:

- Compress the file 'rollnumber.cu', which contains the implementation of the above-described functionality to ROLL NUMBER.zip.
- Submit only the ROLL NUMBER.zip file on Moodle.
- After submission, download the file and make sure it was the one you intended to submit.
- Kindly adhere strictly to the above guidelines.