**Graph**

**Instructions for students:**

* Complete the following classes/methods.
* You may use any language to complete the tasks (Java / Python).
* All your methods **for an individual task** must be written in one single **.java** or **.py** or **.ipynb** file. DO NOT CREATE separate files for each task.

**NOTE:**

* **YOU CANNOT USE ANY BUILT-IN FUNCTION EXCEPT** len **IN PYTHON. [negative indexing, append is prohibited]**
* **YOU HAVE TO MENTION SIZE OF ARRAY WHILE INITIALIZATION**
* **YOUR CODE SHOULD WORK FOR ALL RELEVANT SAMPLE INPUTS**

| **Dear Students, you have been given instructions and driver code for the last 7 labs. For the last two labs of this semester, no driver code will be given. You will develop everything (necessary functions, class, driver code, etc.) in your preferred language (Java or Python).** |
| --- |

To solve the problems, draw a graph according to your preference. However, your graph must have a minimum of **7 vertices** and a minimum of **16 edges**.   
**Include a picture of your graph with your submission.**

**Note:** Solve all the problems for both **adjacency matrix** representation and **adjacency list** representation. You can write different functions for each task to make your life easier.

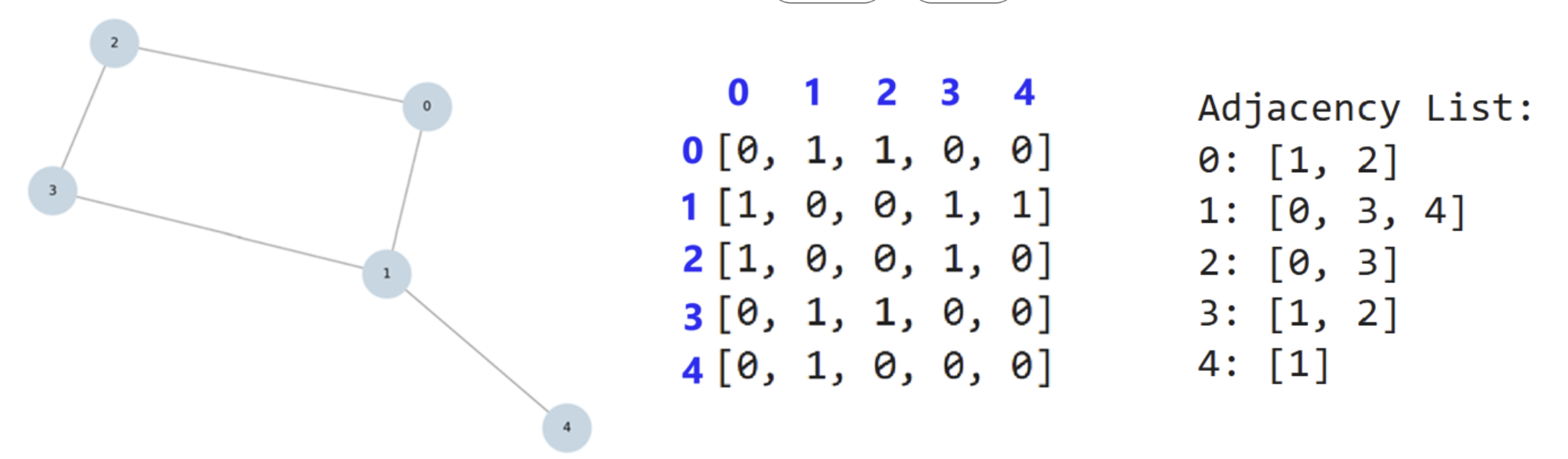
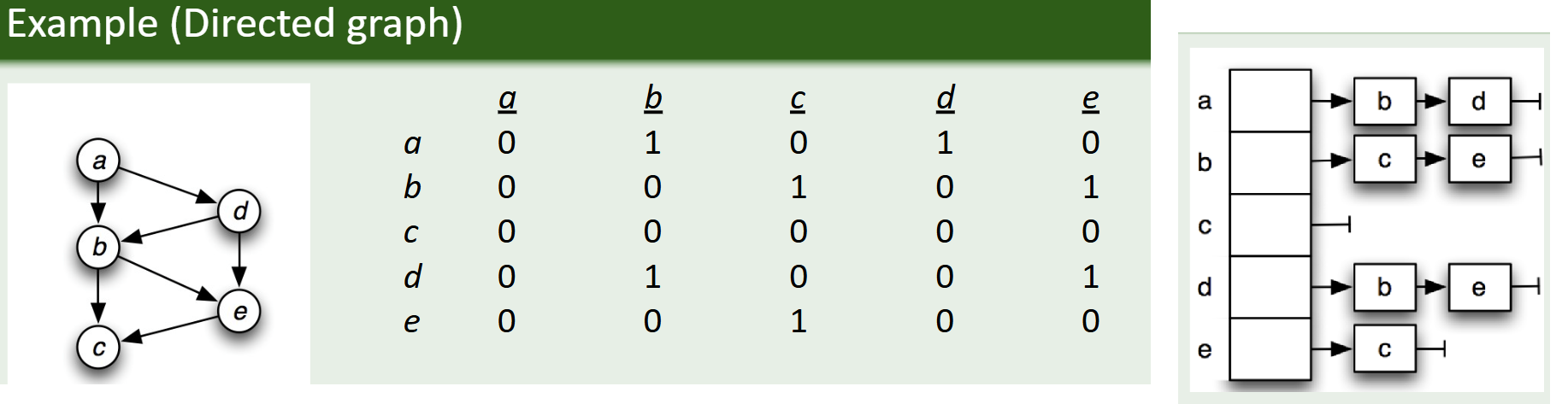
## 

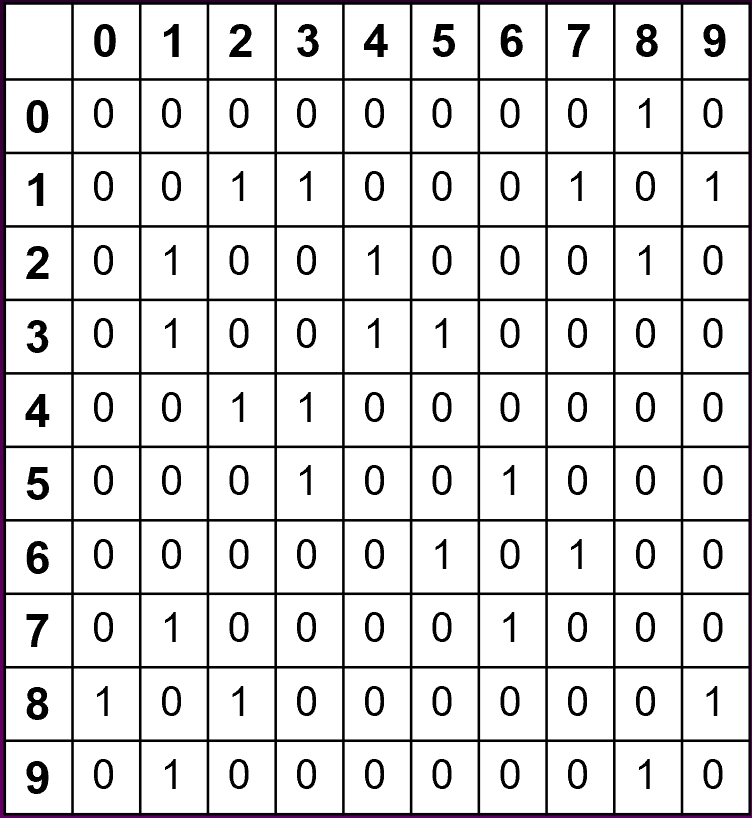
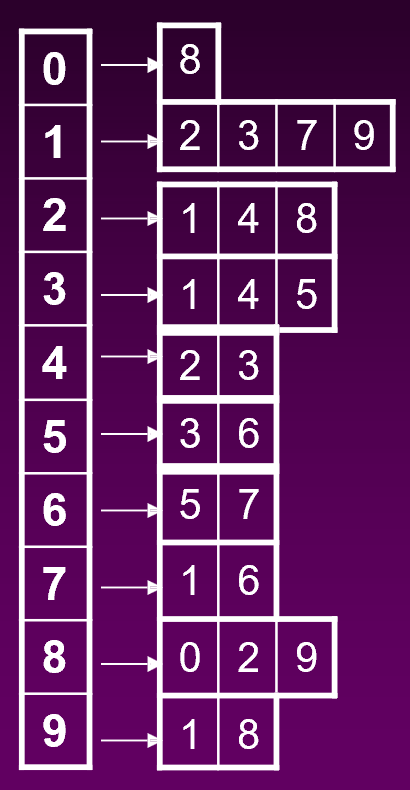
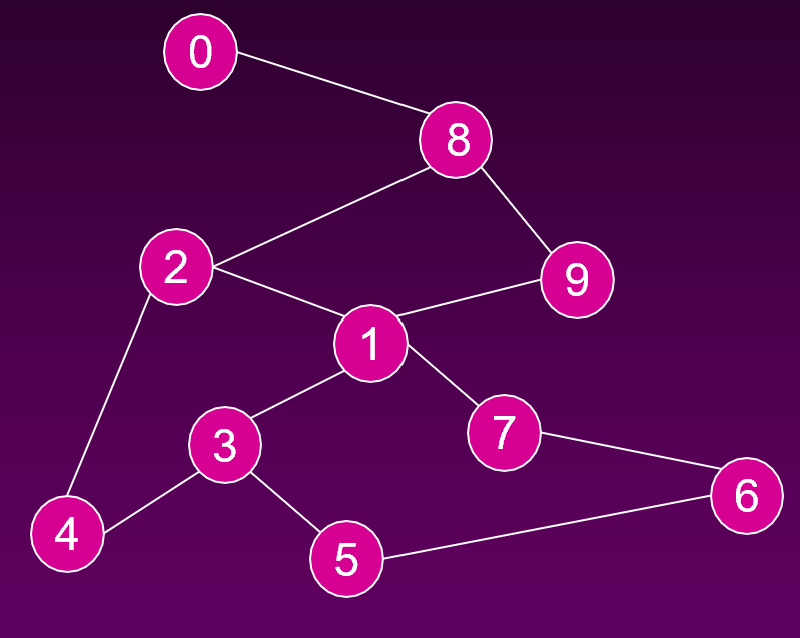
## **Task 1:**

Given an **undirected, unweighted graph** as input, write a function to find the **vertex with maximum degree**   
**Directed, unweighted graph**  
and return the degree of that vertex.   
  
Degree notes: [L16-L17 Graph Basics.pptx](https://docs.google.com/presentation/d/1w89Xj9G0a6r6dph1dzCTuDtoFJP5b-Si/edit#slide=id.g3220fd27c11_0_80)

* **For undirected graph:** 
  + **Degree (deg⁡(v))**: Number of edges connected to v
  + **Count how many 1’s are there for each vertex**
* **For directed graph:** 
  + **In-degree (indeg(v))**: Number of **incoming** edges to v (Number of edges ending at the vertex)
  + **Out-degree (outdeg(v))**: Number of **outgoing** edges from v (Number of edges originating from the vertex)
    - **The matrix and the list only hold the Out-degrees**
    - **Count the size of the LL chain for each vertex**

**Don't follow the format on right side of 1st image for adjacency list, use the chain style similar to hashing**

****



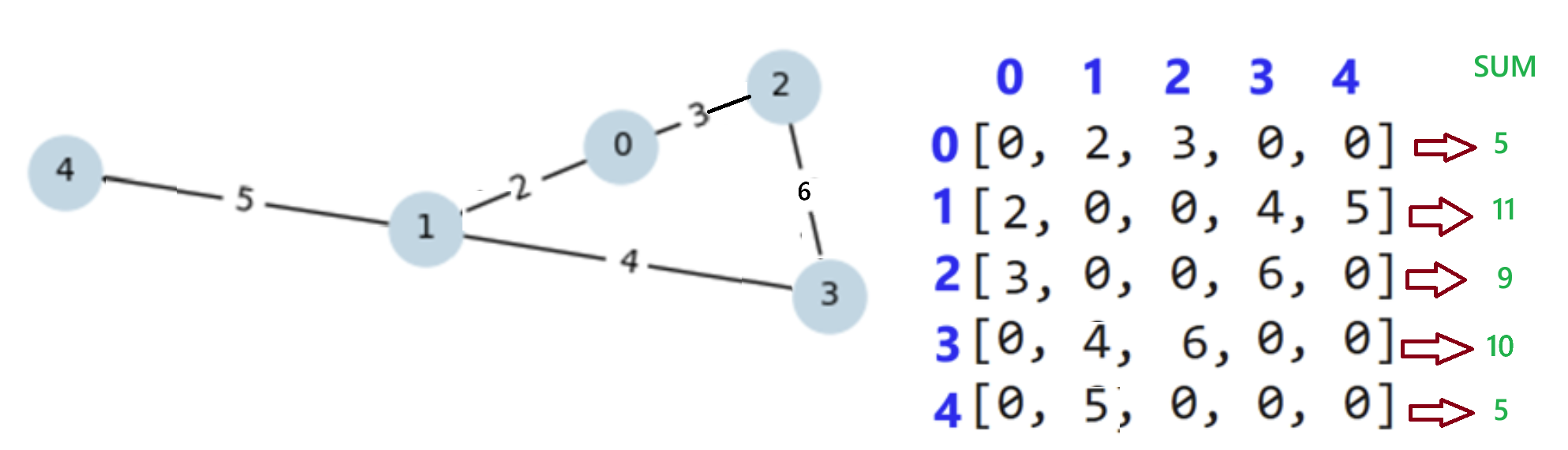
## 

## 

## **Task 2:**

Given an **undirected edge-weighted graph as input,** write a function to  
**find the vertex whose sum of edge weights is maximum.**

Given an **directed edge-weighted graph as input,** write a function to  
**find the vertex whose sum of edge weights is maximum.**

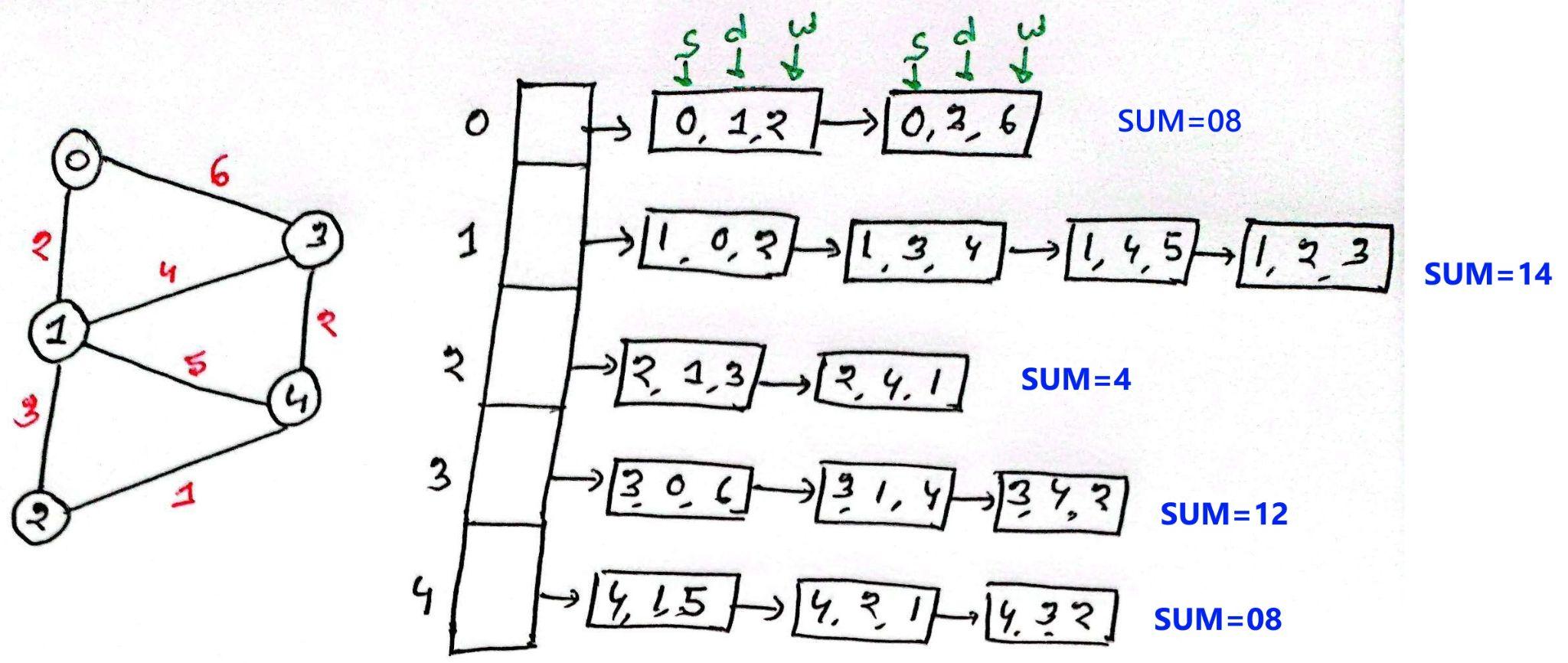
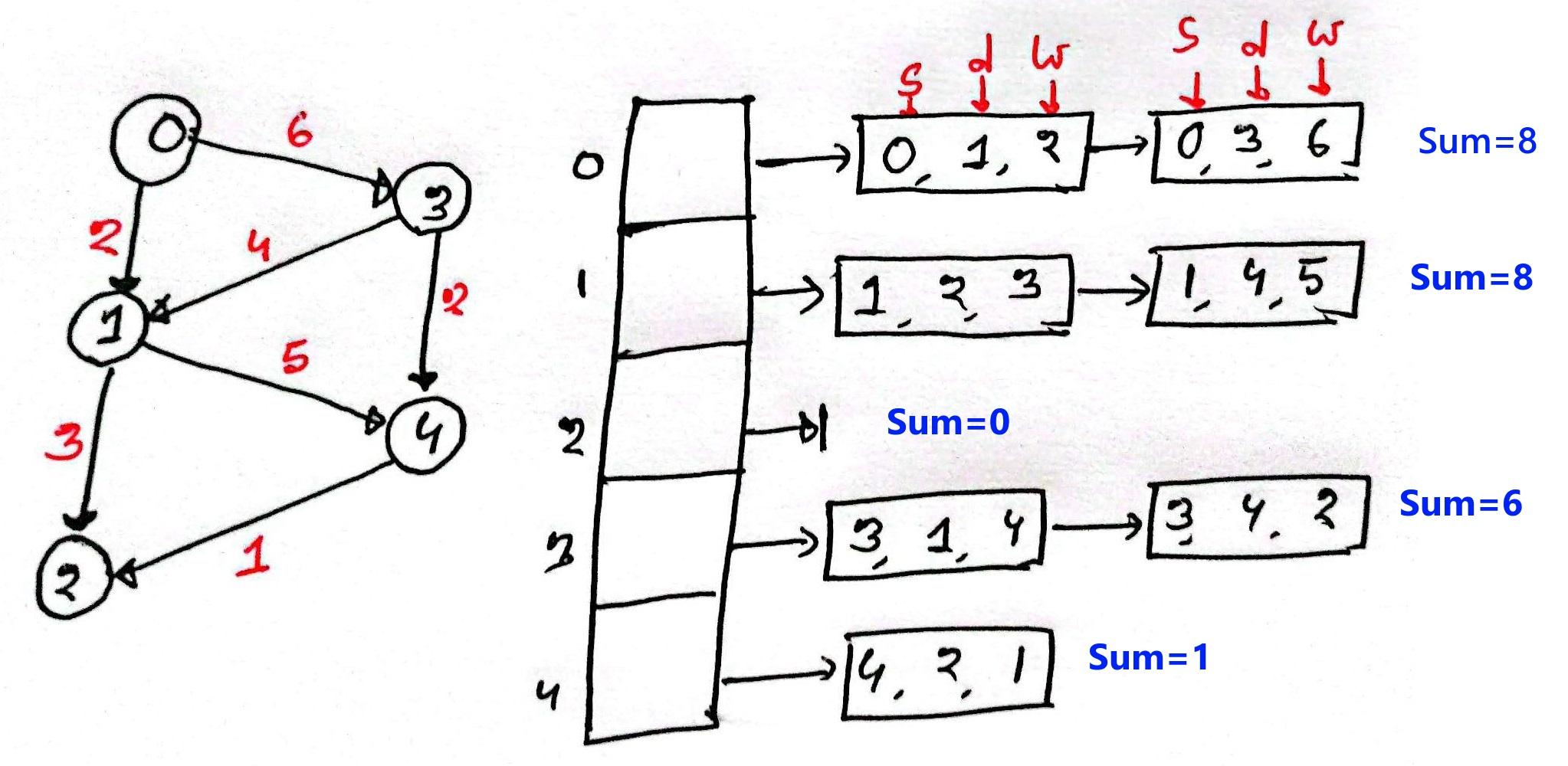


## 

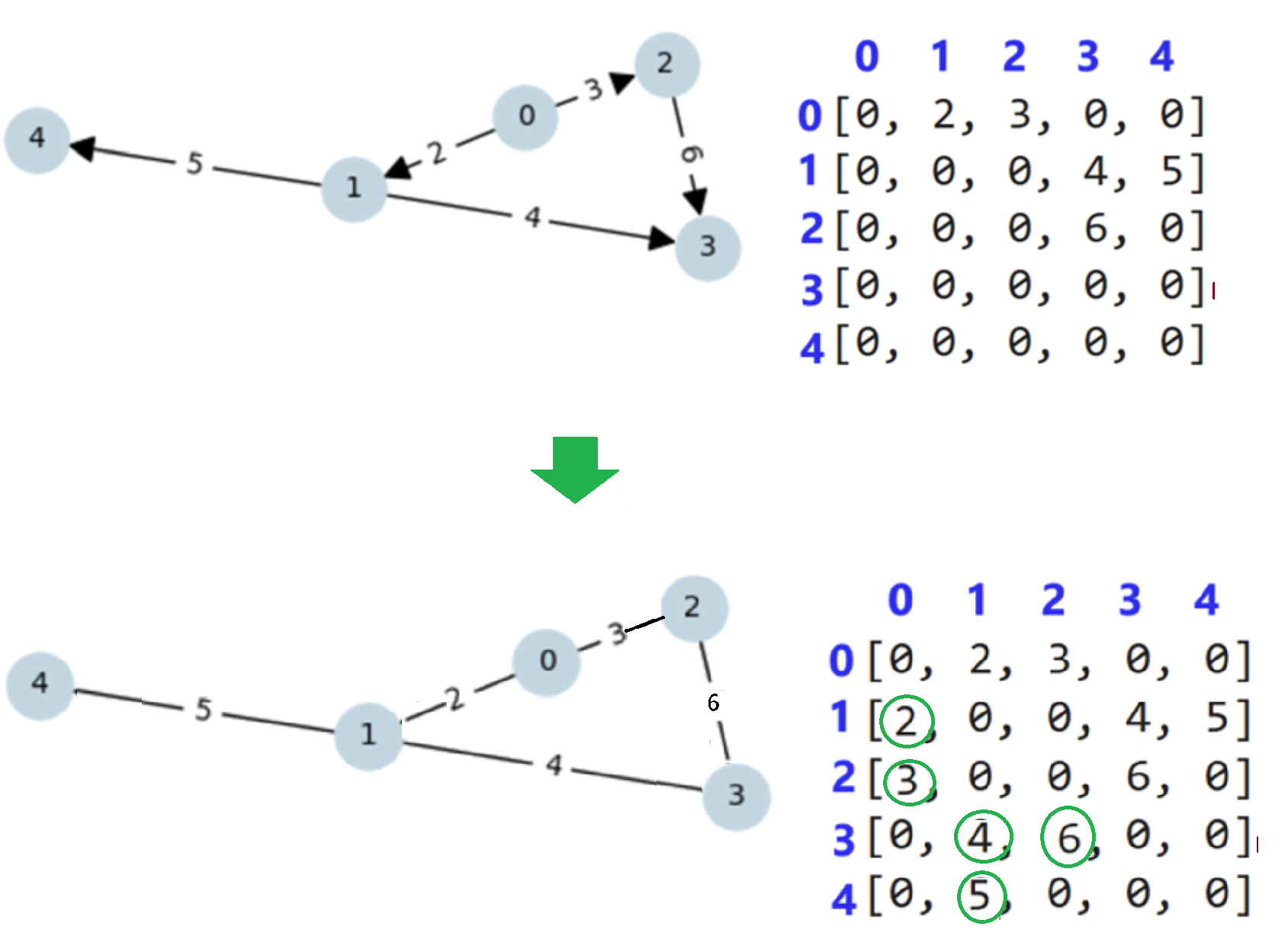
## 

## **Task 3:**

Solve Task 1 and Task 2 for directed, edge-weighted graphs considering only outgoing edges.

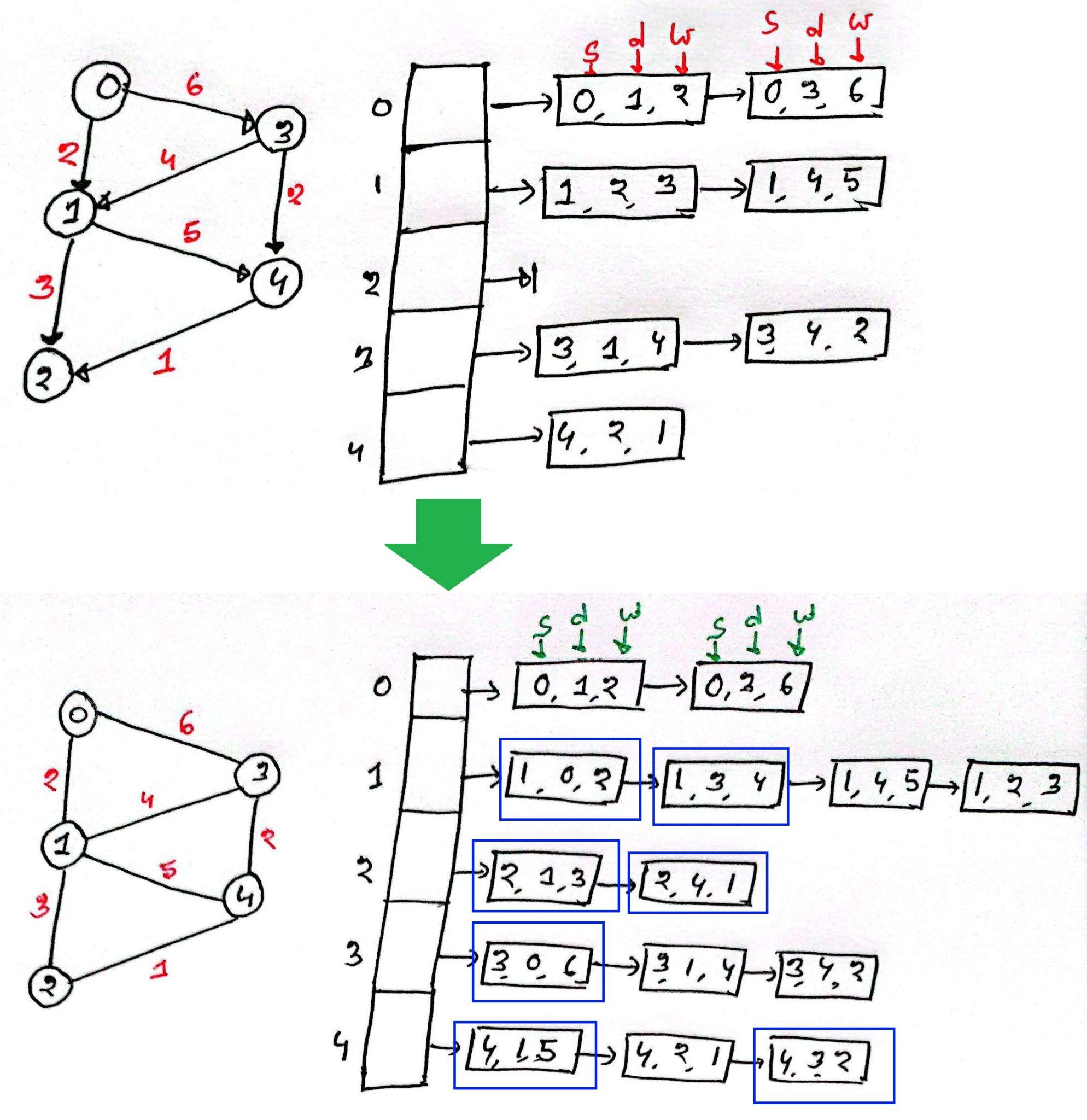
****

**Task 4:**

Write a function that will receive a **directed graph weighted** and convert it to an **undirected weighted graph**. Consider the weight of the edges will be unchanged.  
  
  
The same question appeared in the final:  
### Spring2024 Q5

\* [Spring2024 graph question](https://drive.google.com/file/d/1k6qV3Ij6hL0BrHtc3cB0AnT1GSOR2wSP/view?usp=sharing)

\* [Spring2024 graph question solution](https://drive.google.com/file/d/1sp0m0cOlfkRrHbQ5dPjOxCrH3LZv4DNU/view?usp=sharing)



**Note:** As there are 4 tasks and you must complete them for both **adjacency list and adjacency matrix,** you have to complete 8 tasks in total.