```
# {} distionary/set
# [] list
# () tuple/list



a = [1,5,2,3,5]




# a possible way to implement a graph
# a "relational model" way

n = 8
edges = [[0,1],[0,2],[1,2],[1,3],[1,4],[4,5],[6,7]]




# get neighbors of each vertex
# check if (u,v) exists
# add a new neighbor to a vertex




# Adjacency matrix
adj_matrix = [[0 for i in range(n)] for j in range(n)]

for edge in edges:
        adj_matrix[edge[0]][edge[1]] = 1

# print(adj_matrix[1])




# adjacency list: a possible bad solution
adj_list = {}
for i in range(n):
        adj_list[i] = []

for edge in edges:
        adj_list[edge[0]].append(edge[1])




# adjacency list
adj_list = [[] for j in range(n)]
for edge in edges:
        adj_list[edge[0]].append(edge[1])
        # adj_list[edge[1]].append(edge[0])
# print(adj_list)




#CSR
offset = [0]*(n+1);
csr_edges = [];
for i in range(n):
        offset[i] = len(csr_edges)
        csr_edges.extend(adj_list[i])
offset[n] = len(csr_edges)
# print(cse_edges)
```

```
# print(offset)

v = 1
for i in range(offset[v],offset[v+1]):
        print(csr_edges[i])
```