BIỂU DIỄN ĐỒ THỊ

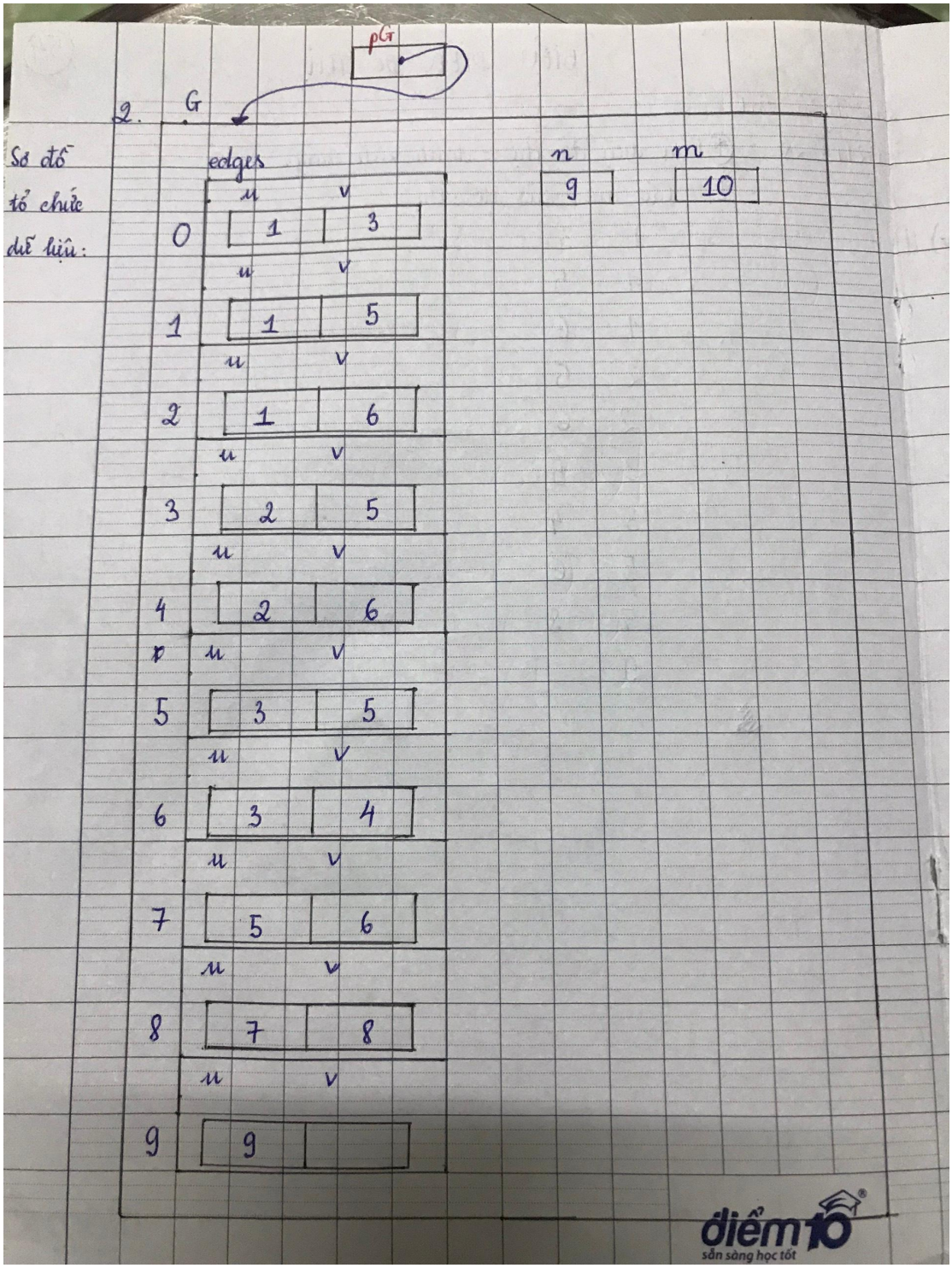
Lê Tuấn Đạt
B2113328

(*) Biểu diễn đồ thị = danh sách cung.

1. Các cung của đồ thị:

1 3
1 5
1 6
2 5
2 6
3 5
3 4
5 6
7 8
9

2. G



Bài tập lập trình.

3.
```
void init_Graph(Graph *pG, int n){
    pG->n = n;
    pG->m = 0;
}
```

4.
```
void add_edge(Graph *pG, int u, int v){
    pG->edges[pG->m].u = u;
    pG->edges[pG->m].v = v;
    pG->m++;
}
```

5.
```
int degree(Graph *pG, int u){
    int e, deg_u = 0;
    for(e = 0; e < pG->m; e++){
        if(pG->edges[e].u == u) //Cung có dạng (u, -)
            deg_u++;
        if(pG->edges[e].v == u) //Cung có dạng (-, u)
            deg_u++;
    }
    return deg_u;
}
```

6.
```
int adjacent(Graph *pG, int x, int y){
    int e;
    for(e=0; e<pG->m; e++)
        if((pG->edges[e].u == u && pG->edges[e].v == v) ||
           (pG->edges[e].u == v && pG->edges[e].v == u))
            return 1;
    return 0;
}
```

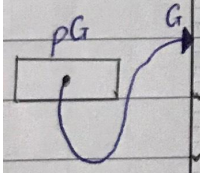
7.
```
void neighbours(Graph *pG, int u){
    int v;
    for(v=1; v<=pG->n; v++)
        if(adjacent(pG, u, v))
            printf("%d ", v);
    print("\n");
}
```

④ Biểu diễn đồ thị = ma trận kề (ma trận đỉnh đỉnh)

8. Ma trận kề:

	1	2	3	4	5	6	7	8	9
1	0	1	1	1	0	0	0	0	0
2	1	0	1	0	0	0	0	0	0
3	1	1	0	0	1	0	1	0	0
4	1	0	0	0	0	0	0	0	0
5	0	0	1	0	0	1	1	0	0
6	0	0	0	0	1	0	0	0	1
7	0	0	1	0	1	0	0	1	0
8	0	0	0	0	0	0	1	0	0
9	0	0	0	0	0	1	0	0	0

9. Sơ đồ tổ chức:



A	1	2	3	4	5	6	7	8	9
1	0	0	1	0	1	1	0	0	0
2	0	0	0	0	1	1	0	0	0
3	1	0	0	1	1	0	0	0	0
4	0	0	1	0	0	0	0	0	0
5	1	1	1	0	0	1	0	0	0
6	1	1	0	0	1	0	0	0	0
7	0	0	0	0	0	0	0	1	0
8	0	0	0	0	0	0	1	0	0
9	0	0	0	0	0	0	0	0	0

n
9

m
10

10.

```
void init_graph(Graph *pG, int n){
    pG->n = n;
    pG->m = 0;
    for(int u = 1; u <= n; u++)
        for(int v = 1; v <= n; v++)
            pG->A[u][v] = 0;
}
```

11.

```
void add_edge(Graph *pG, int u, int v){
    pG->A[u][v] = 1;
    pG->A[v][u] = 1;
    pG->m++;
}
```

12. Đồ thị không có khuyên:

```
int degree(Graph *pG, int u){
    int deg_u = 0;
    for(int v = 1; v <= pG->n; v++)
        deg_u += pG->A[u][v];
    return deg_u;
}
```

13.
```
int adjacent(Graph *pG, int u, int v){
    return pG->A[u][v] > 0;
}
```

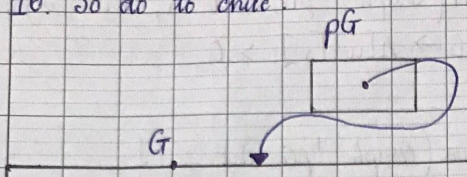
14.
```
void neighbours(Graph *pG, int x){
    int y;
    for(y = 1; y <= pG->n; y++)
        if(pG->A[x][y] != 0)
            printf("%d ", y);
    printf("\n");
}
```

⊛ Biểu diễn = ma trận đỉnh - cung:

15.

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
1	1	1	1	1	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0
3	0	1	1	1	0	1	0	0	0
4	0	0	0	0	1	1	0	0	1
5	0	0	0	0	0	0	1	1	1
6	0	0	0	1	0	0	1	1	0
7	0	0	0	0	0	0	0	0	0

16. Sơ đồ tổ chức:



A

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	n
1	1	1	1	0	0	0	0	0	0	0	8
2	0	0	0	1	1	0	0	0	0	1	m
3	1	0	0	0	0	1	1	0	0	0	10
4	0	0	0	0	0	1	0	0	0	1	
5	0	1	0	1	0	0	1	1	0	0	
6	0	0	1	0	1	0	0	1	0	0	
7	0	0	0	0	0	0	0	0	1	0	
8	0	0	0	0	0	0	0	0	1	0	

17.
```
void init_graph(Graph *pG, int n, int m){
    int u,e;
    pG->n = n;
    pG->m = m;
    for(u=1; u<=n; u++)
        for(e=1; e<=m; e++)
            pG->A[u][e] = 0;
}
```

```
18. void add_edge(Graph *pG, int e, int x, int y){
        pG->A[u][e] = 1;
        pG->A[v][e] = 1;
    }

20  int adjacent(Graph *pG, int x, int y){
        int e;
        for(e=1; e<=pG->m; e++)
            if(pG->A[u][e]==1 &&
                    pG->A[v][e]==1)
                return 1;
        return 0;
    }

19. int degree(Graph *pG, int x){
        int e, deg_u = 0;
        for(e=1; e<=pG->m; e++)
            if(pG->A[x][e] == 1)
                deg_u++;
        return deg_u;
    }
```

21.
```
void neighbours(Graph *pG, int x){
    int y;
    for(y=1; y<=pG->n; y++)
        if(pG->A[x][y] != 0)
            printf("%d ", y);
    printf("\n");
}
```

④ Biểu diễn = danh sách cạnh kề:

22.

adj[1] = [3, 5, 6]
adj[2] = [5, 6]
adj[3] = [1, 4, 5]
adj[4] = [3]
adj[5] = [1, 2, 3, 6]
adj[6] = [1, 2, 5]
adj[7] = [8]
adj[8] = [7]
adj[9] = []

23.

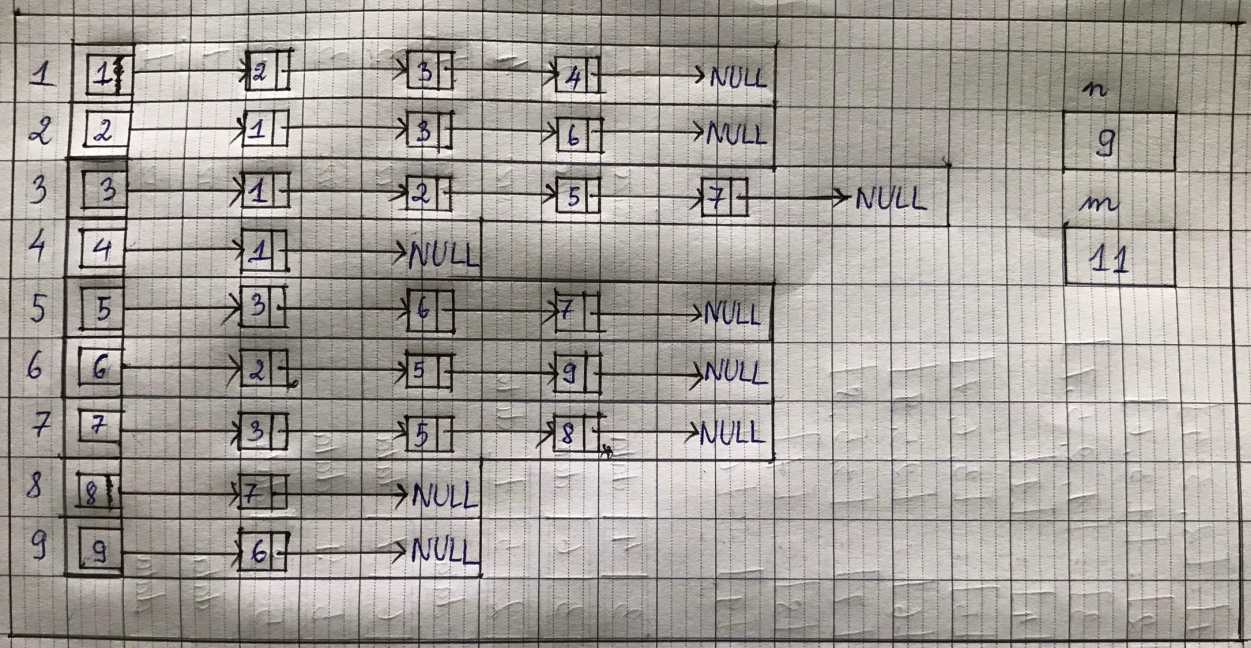
adj[1] = [2, 3, 4]
adj[2] = [1, 3, 6]
adj[3] = [1, 2, 5, 7]
adj[4] = [1]
adj[5] = [3, 6, 7]
adj[6] = [2, 5, 9]
adj[7] = [3, 5, 8]
adj[8] = [7]
adj[9] = [6]

Sơ đồ tổ chức (danh sách ~~đôi~~ đôi)

adj

1	2	3	4	data	size	3
2	1	3	6	data	size	3
3	1	2	5	7	data	size 4
4	1	data			size	1
5	3	6	7	data	size	3
6	2	5	9	data	size	3
7	3	5	8	data	size	3
8	7	data			size	1
9	6	data			size	1

23. Sơ đồ tổ chức dữ liệu bằng danh sách đính kề:
(danh sách liên kết)



Graph

```
25. void add_edge(Graph *pG, int x, int y){
        push_back(&pG->adj[x], y);
        push_back(&pG->adj[y], x);
    }
26. int degree(Graph *pG, int x){
        return size(&pG->adj[x]);
    }
```

```
27. int adjacent(Graph *pG, int x, int y){
        int u;
        for(u=1; u<=pG->adj[x].size; u++)
            if(element_at(&pG->adj[x], u) == y)
                return 1;
        return 0;
    }
```

28. void neighbours (Graph *pG, int x){
int y;
for(y=1; y<=pG->n; y++)
if (adjacent(pG, x, y))
printf("%d ", y);
printf("\n");