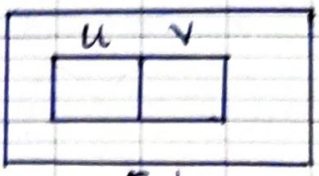
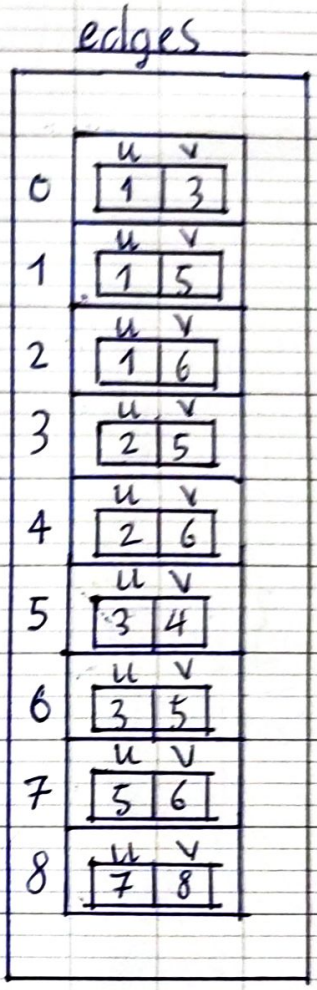
Bài tập :

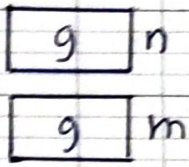
1) Liệt kê các cung

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

2) Sơ đồ tổ chức dữ liệu.







3)
```
void init_Graph ( Graph *G , int n ){
    G→n = n ; G→m = 0;
}
```

4)
```
void add_edge ( Graph *G, int x, int y ){
    G→edge[G→m].u = x;
    G→edge[G→m].v = y;
    G→m++;
}
```

Date

5) int degree (Graph *G, int x){
    int e, deg = 0;
    for(e = 1; e <= G->m; e++){
        if(G->edge[e].u == x)
            deg++;
        if(G->edge[e].v == x)
            deg++;
    }
    return deg;
}

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

No.

Date

7)
```
void neighbours (Graph *G, int x){
    int i
    for (i=1; i<= G->n; i++)
        if (adjacent (G, x, i) != 0)
            printf ("%d ", i);
    printf ("/n");
}
```

8)

-	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	1	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	1	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

Date

9)

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: 9

Graph

10)

```
void inti_Graph ( Graph *G , int n ) {
    int i, j ;
    for (i=1 ; i<=n ; i++) {
        for (j=1 ; j<=n ; j++)
            G→A[i][j]= 0 ;
}
```

No.

Date

11) void addl_Graph (Graph *G, int x, int y){
    G→A[x][y] = 1;
    G→A[y][x] = 1;
}

12) int degree (Graph *G, int x){
    int i, deg=0;
    for (i=1; i<=G→n ; i++)
        if (G→A[i][x] != 0)
            deg++;
    return deg;
}

13) int adjacent (Graph *G , int x, int y){
    return G→A[x][y] != 0;
}

14) void neighbours (Graph *G, int x){
    int i;
    for (i=1; i<=G→n; i++)
        if (G→A[i][x] != 0)
            printf("%d ", i);
}

Date

15)

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
1	1	1	1	0	1	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)

A

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
1	1	1	1	0	0	0	0	0	0	0
2	0	0	0	1	1	0	0	0	0	1
3	1	0	0	0	0	1	1	0	0	0
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

n = 8

m = 10

Graph.

17, void init_Graph (Graph *G, int n, int m){
    int i, j;
    G->n = n; G->m = m;
    for (i=1; i<=G->n; i++){
        for (j=1; j<=G->m; i++)
            G->A[i][j] = 0;
    }
}

18) void add_edge (Graph *G, int e, int x, int y){
    G->A[x][e] = 1;
    G->A[y][e] = 1;
}

19) int degree (Graph *G, int x){
    int i, deg=0;
    for (i=0; i<=G->n; i++)
        if (G->A[x][i] == 1)
            deg++;
    return deg;
}

No.
Date

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

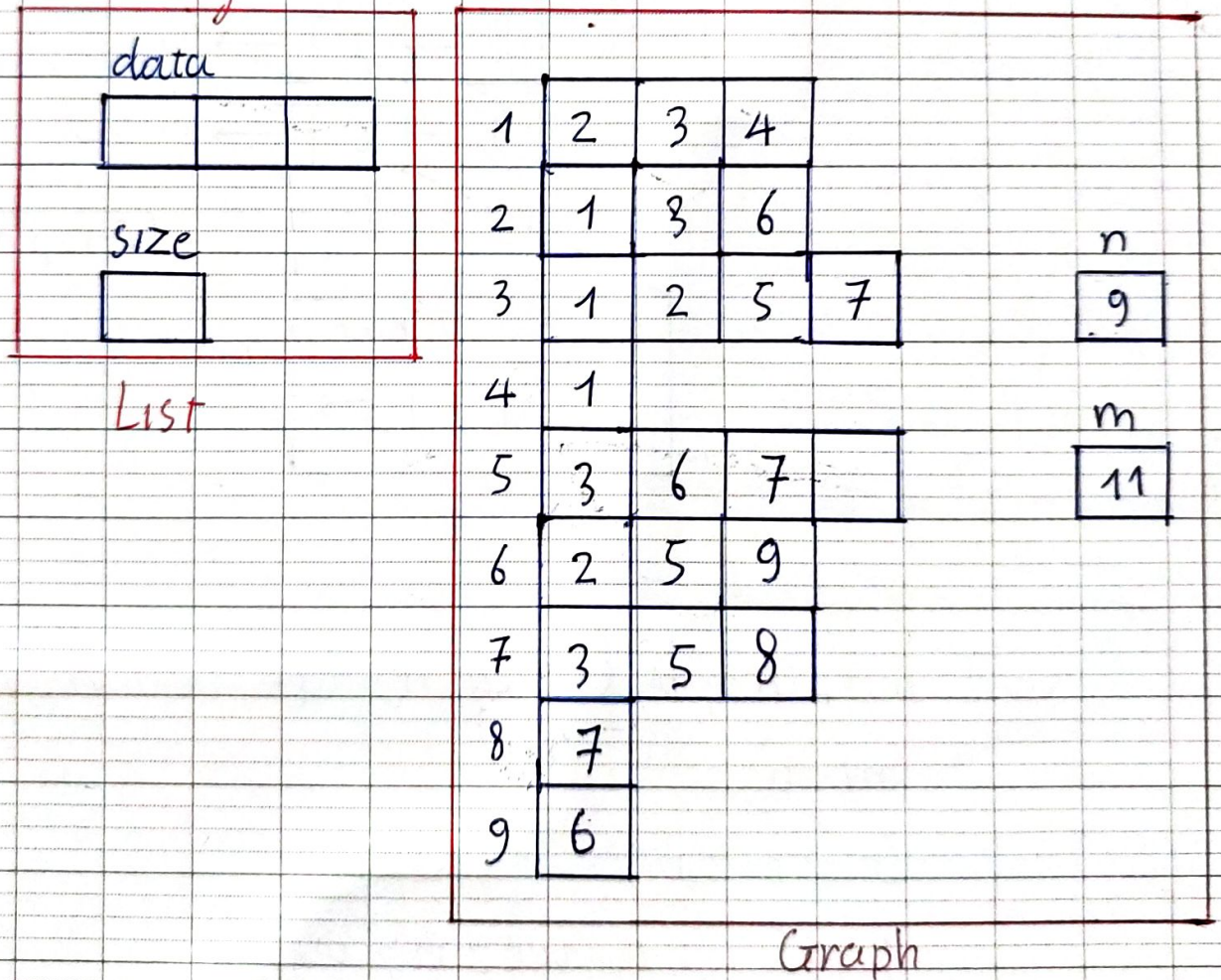
21)
```
void neighbours (Graph *G, int x){
    int e;
    for (e=1; e<=G->n; e++)
        if (G->A[y][x]==1)
            printf("%d ", e);
}
```

22) Danh sách các đỉnh kề

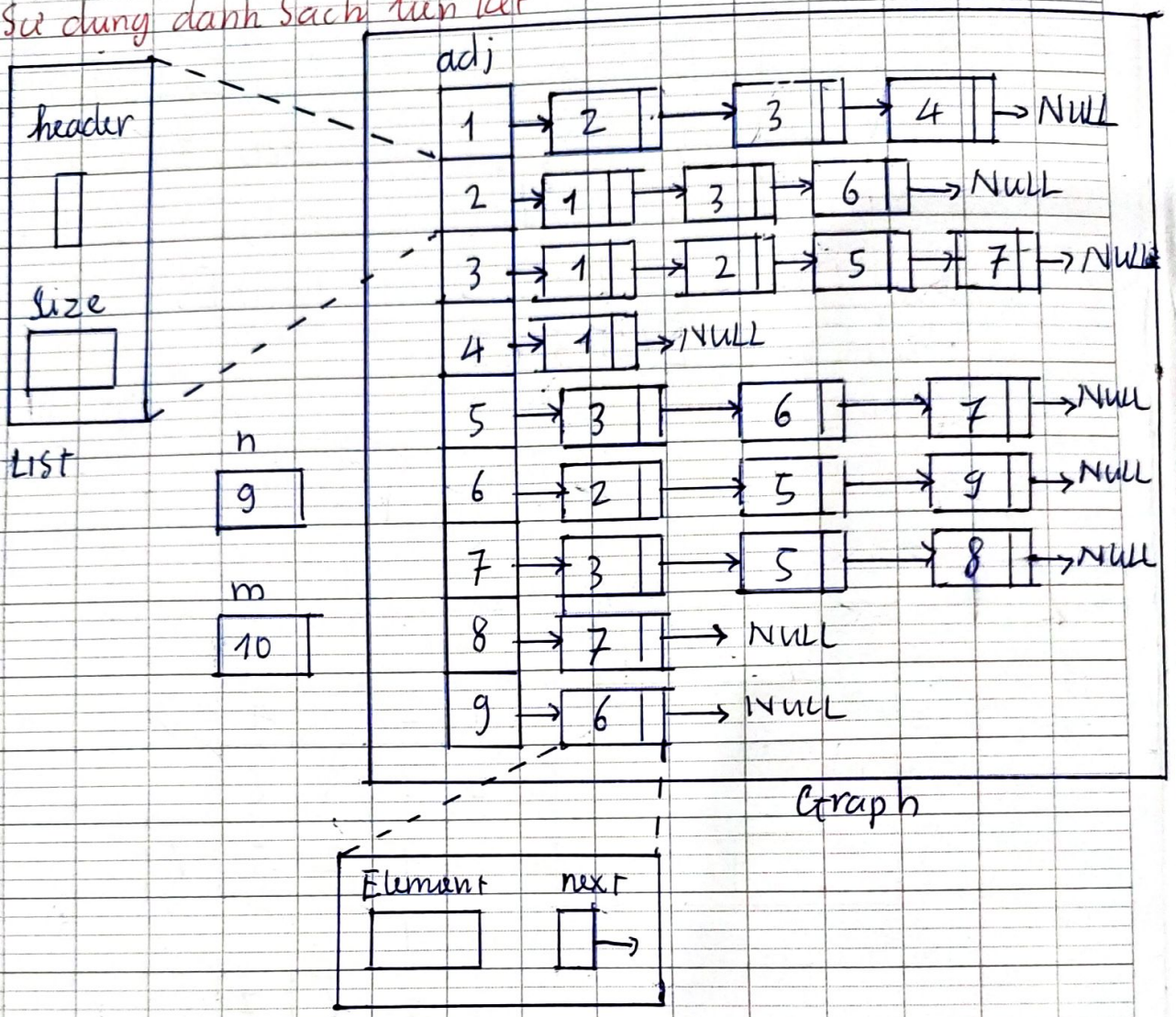
adj[1] = [3, 5, 6]
adj[2] = [5, 6]
adj[3] = [1, 4, 5]
adj[4] = [3]
adj[5] = [1, 2, 3, 6]
adj[6] = [1, 2, 5]
adj[7] = [8]
adj[8] = [7]
adj[9] =

23

- Sử dụng danh sách cạnh



- Sử dụng danh sách liên kết



24)

```
void init_Graph ( Graph *G, int n, int m ){
    int u;
    G->n = n;
    for ( u=1; u<=n; u++ )
        make_null (&G->adj[u] );
}
```

Date

25) void add_edge(Graph, *G, int x, int y) {
push_back (&G->adj[x], y);
push_back (&G-> adj[y], x);
}

26) int degree (Graph *G , int x) {
int i, deg = 0;
for (i= 1, i<= G-> n; i++)
if(G-> adj[x].size == 1)
deg ++;
}

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

28 void neighbours (Graph *G , int x) {
int i;
for (i=1; i<= G-> adj; i++)
if (adjacent (G, x, i) != 0)
printf ("%d ", i);
}