Ma trận A[100][100]

1.

```
int hasCycle = 0, color[MAXN];
void dfs(Graph *pG, int u) {
	color[u] = GRAY;
	int i;
	for (i = 1; i <= pG->n; i++)
		if (pG->A[u][i] && color[i] == WHITE)
			dfs(pG, i);
		else if (pG->A[u][i] && color[i] == GRAY)

	hasCycle = 1;
	color[u] = BLACK;
}
int main() {
	int n, m, u, v, e;
	scanf("%d%d", &n, &m);
	Graph G;
	initGraph(&G, n);
	for (e = 1; e <= m; e++)
		scanf("%d%d", &u, &v), addEdge(&G, u, v);
	for (u = 1; u <= n; u++)
		if (color[u] == WHITE)

	dfs(&G, u);
	if (!hasCycle)
		printf("HOP LE");
	else printf("KHONG");
	return 0;
}
```

2.

```
void check(Graph *pG, int u, int v) {
	int i, has = 0;
	for (i = 1; i <= pG->n; i++)
		if (pG->A[u][i] && pG->A[v][i])

	printf("%d ", i), has = 1;
	if (!has)
		printf("KHONG CHUNG DOI THU");
}
int main() {
	int n, u, v, w;
	Graph G;
	scanf("%d", &n);
	initGraph(&G, n);
	for (u = 1; u <= n; u++)
		for (v = 1; v <= n; v++) {

	scanf("%d", &w);
			if (w)addEdge(&G, u, v);
		}
	scanf("%d%d", &u, &v);
	check(&G, u, v);
	return 0;
}
```

3.

```
int outDegree(Graph *pG, int u) {
	int d = 0, v;
	for (v = 1; v <= pG->n; v++)
		d += (pG->A[v][u]);
	return d;
}
int main() {
	int n, m, u, v, e;
	scanf("%d%d", &n, &m);
	Graph G;
	initGraph(&G, n);
	for (e = 1; e <= m; e++)
		scanf("%d%d", &u, &v), addEdge(&G, u, v);
	scanf("%d", &u);.
	printf("%d:%d", u, outDegree(&G, u));
	return 0;
}
```

4.

```
int outDegree(Graph *pG, int u) {
	int d = 0, v;
	for (v = 1; v <= pG->n; v++)
		d += (pG->A[v][u]);
	return d;
}
int main() {
	int n, u, w, i, j;
	scanf("%d", &n);
	Graph G;
	initGraph(&G, n);
	for (i = 1; i <= n; i++)
		for (j = 1; j <= n; j++) {

	scanf("%d", &w);
			if (w)addEdge(&G, i, j);
		}
	scanf("%d", &u);
	printf("%d:%d", u, outDegree(&G, u));
	return 0;
}
```

Ma trận A[100][100]

6.

```
int color[MAXN], conflict = 0;

void colorize(Graph *pG, int u, int c) {
	color[u] = c;
	int i;
	for (i = 1; i <= pG->n; i++) {
		if (pG->A[u][i]) {
			if (color[i] == NO_COLOR)
	colorize(pG, i, BLACK + WHITE - c);
			else if (color[i] == color[u])
	conflict = 1;
		}
	}
}

int main() {
	int n, m, u, v, e;
	scanf("%d%d", &n, &m);
	Graph G;
	initGraph(&G, n);
	for (e = 1; e <= m; e++)
		scanf("%d%d", &u, &v), addEdge(&G, u, v);
	for (u = 1; u <= n; u++)
		if (color[u] == NO_COLOR)
	colorize(&G, u, BLACK);
	if (conflict)
		printf("-1 -1");
	else {
		int count = 0;
		for (u = 1; u <= n; u++)
			if (color[u] == BLACK)
	count++;
		printf("%d %d", count, n - count);
	}
	return 0;
}
```

9.

```
void check(Graph *pG, int u, int v) {
	int i, count = 0;
	for (i = 1; i <= pG->n; i++)
		if (pG->A[u][i] && pG->A[v][i])
	count++;
	printf("%d", count);
}

int main() {
	int n, u, v, w;
	Graph G;
	scanf("%d", &n);
	initGraph(&G, n);
	for (u = 1; u <= n; u++)
		for (v = 1; v <= n; v++)
{
	scanf("%d", &w);
			if (w)
	addEdge(&G, u, v);
		}
	scanf("%d%d", &u, &v);
	check(&G, u, v);
	return 0;
}
```

10.

```
int degree(Graph *pG, int u) {
	int d = 0, v;
	for (v = 1; v <= pG->n; v++)
		d += (pG->A[v][u]);
	return d;
}
int main() {
	int n, m, u, v, e;
	scanf("%d%d", &n, &m);
	Graph G;
	initGraph(&G, n);
	for (e = 1; e <= m; e++)
		scanf("%d%d", &u, &v), addEdge(&G, u, v);
	scanf("%d", &u);
	printf("%d:%d", u, degree(&G, u));
	return 0;
}
```

11.c

```
int degree(Graph *pG, int u) {
	int d = 0, v;
	for (v = 1; v <= pG->n; v++)
		d += (pG->A[v][u]);
	return d;
}
int main() {
	int n, u, i, j;
	scanf("%d", &n);
	Graph G;
	initGraph(&G, n);
	for (i = 1; i <= n; i++)
		for (j = 1; j <= n; j++)
	scanf("%d", &G.A[i][j]);
	scanf("%d", &u);
	printf("%d:%d", u, degree(&G, u));
	return 0;
}
```

Ma trận A[100][100]

12.c

```
void dijkstra(Graph *pG, int s, int pi[], int p[]) {
	int i, mark[MAXN];
	for (i = 1; i <= pG->n; i++)
		pi[i] = INF, p[i] = -1, mark[i] = 0;
	pi[s] = 0;
	int _;
	for (_ = 0; _ < pG->n - 1; _++) {
		int minPi = INF, u = -1, v;
		for (v = 1; v <= pG->n; v++)
			if (!mark[v] && pi[v] < minPi) {
	minPi = pi[v];
	u = v;
			}
		if (u == -1)
			break;
		mark[u] = 1;
		for (v = 1; v <= pG->n; v++)
			if (!mark[v] && pG->A[u][v] != NO_EDGE && pi[v] > pi[u] + pG->A[u][v])
	pi[v] = pi[u] + pG->A[u][v], p[v] = u;
	}
}
int main() {
	int n, m, u, v, w, e;
	scanf("%d%d", &n, &m);
	Graph G;
	initGraph(&G, n);
	for (e = 1; e <= m; e++)
		scanf("%d%d%d", &u, &v, &w), addEdge(&G, u, v, w);
	scanf("%d%d", &u, &v);
	int pi[MAXN], p[MAXN];
	dijkstra(&G, u, pi, p);

	printf("%d\n", pi[v]);
	int path[MAXN], len = 0;
	while (v != -1) {
		path[len++] = v;
		v = p[v];
	}
	for (e = len - 1; e >= 0; e--)
		printf("%d ", path[e]);
	return 0;
}
```

15.c

```
int numOfEdges(Graph *pG, int u, int v) {
	return pG->A[u][v];
}
int main() {
	int n, m, u, v, e;
	scanf("%d%d", &n, &m);
	Graph G;
	initGraph(&G, n);
	for (e = 1; e <= m; e++) {
		scanf("%d%d", &u, &v), addEdge(&G, u, v);
	}
	scanf("%d%d", &u, &v);
	if (!numOfEdges(&G, u, v))
		printf("HAY CHAO NHAU DI");
	else
		printf("%d", numOfEdges(&G, u, v));
	return 0;
}
```

```
#include<stdio.h>

#define MAXN 101
#define BLACK 1
#define WHITE 2
#define NO_COLOR 0
int color[MAXN], conflict = 0;

void colorize(Graph *pG, int u, int c) {
	color[u] = c;

	int i;
	for (i = 1; i <= pG->n; i++) {
		if (pG->A[u][i]) {
			if (color[i] == NO_COLOR)
	colorize(pG, i, BLACK + WHITE - c);
			else if (color[i] == color[u])
	conflict = 1;
		}
	}
}
int main() {
	int n, m, u, v, e;
	scanf("%d%d", &n, &m);
	Graph G;
	initGraph(&G, n);
	for (e = 1; e <= m; e++)
		scanf("%d%d", &u, &v), addEdge(&G, u, v);
	for (u = 1; u <= n; u++)
		if (color[u] == NO_COLOR)
```

Ma trận A[100][100]

```
		colorize(&G, u, BLACK);
		if (conflict)
			printf("-1 -1");
		else {
			int count = 0;
			for (u = 1; u <= n; u++)
				if (color[u] == BLACK)
		count++;
			printf("%d %d", count, n - count);
		}
		return 0;
}
```

18.c

```
int mark[MAXN];

void dfs(Graph *pG, int u) {
		mark[u] = 1;
		int i;
		for (i = 1; i <= pG->n; i++)
			if (!mark[i] && pG->A[u][i])
				dfs(pG, i);
}
int main() {
		int n, m, u, v, e;
		scanf("%d%d", &n, &m);
		Graph G;
		initGraph(&G, n);
		for (e = 1; e <= m; e++)
			scanf("%d%d", &u, &v), addEdge(&G, u, v);
		scanf("%d", &u);
		dfs(&G, u);
		for (v = 1; v <= n; v++)
			if (mark[v]) printf("%d ", v);
		return 0;
}
```

20.c

```
int main() {
		int n, u, v;
		scanf("%d", &n);
		Graph G;
		initGraph(&G, n);
		for (u = 1; u <= n; u++)
			for (v = 1; v <= n; v++)
		scanf("%d", &G.A[u][v]);
		for (u = 1; u <= n; u++)
			for (v = 1; v <= n; v++)
				if (G.A[u][v])
		printf("%d has sent %d email(s) to %d\n", u, G.A[u][v], v);
		return 0;
}
```

21.c

```
int degree(Graph *pG, int u) {
		int d = 0, v;
		for (v = 1; v <= pG->n; v++)
			d += pG->A[u][v];
		return d;
}
int main() {
		int n, m, u, v, e;
		scanf("%d%d", &n, &m);
		Graph G;
		initGraph(&G, n);
		for (e = 1; e <= m; e++)
			scanf("%d%d", &u, &v), addEdge(&G, u, v);
		for (u = 1; u <= n; u++)
			printf("%d\n", degree(&G, u));
		return 0;
}
```

23.c

```
int inDegree(Graph *pG, int u) {
		int d = 0, v;
		for (v = 1; v <= pG->n; v++)
			d += (pG->A[v][u]);
		return d;
}
int main() {
		int n, m, u, v, e;
		scanf("%d%d", &n, &m);
		Graph G;
		initGraph(&G, n);
		for (e = 1; e <= m; e++)
			scanf("%d%d", &u, &v), addEdge(&G, u, v);
		u = 1;
		for (v = 2; v <= n; v++)
			u = (inDegree(&G, u) >= inDegree(&G, v) ? u : v);
		printf("%d has received %d email(s).", u, inDegree(&G, u));
		return 0;
}
```

Ma trận A[100][100]

24.c

```
int inDegree(Graph *pG, int u) {
	int d = 0, v;
	for (v = 1; v <= pG->n; v++)
		d += (pG->A[v][u]);
	return d;
}
int outDegree(Graph *pG, int u) {
	int d = 0, v;
	for (v = 1; v <= pG->n; v++)
		d += (pG->A[u][v]);
	return d;
}
int main() {
	int n, m, u, v, e;
	scanf("%d%d", &n, &m);
	Graph G;
	initGraph(&G, n);
	for (e = 1; e <= m; e++)
		scanf("%d%d", &u, &v), addEdge(&G, u, v);
	scanf("%d", &u);
	printf("%d %d", outDegree(&G, u), inDegree(&G, u));
	return 0;
}
```

25.c int inDegree

```
int main() {
	int n, u, i, j;
	scanf("%d", &n);
	Graph G;
	initGraph(&G, n);
	for (i = 1; i <= n; i++)
		for (j = 1; j <= n; j++)
	scanf("%d", &G.A[i][j]);
	scanf("%d", &u);
	printf("%d:%d", u, inDegree(&G, u));
	return 0;
}
```

26.c int degree

```
int main() {
	int n, u, i, j;
	scanf("%d", &n);
	Graph G;
	initGraph(&G, n);
	for (i = 1; i <= n; i++)
		for (j = 1; j <= n; j++)
	scanf("%d", &G.A[i][j]);
	scanf("%d", &u);
	printf("%d:%d", u, degree(&G, u));
	return 0;
}
```

27.c

```
int color[MAXN], conflict = 0;
void colorize(Graph *pG, int u, int c) {
	color[u] = c;
	int i;
	for (i = 1; i <= pG->n; i++) {
		if (pG->A[u][i]) {
	if (color[i] == NO_COLOR)
	colorize(pG, i, BLACK + WHITE - c);
		else if (color[i] == color[u])
	conflict = 1;
		}
	}
}
int main() {
	int n, m, u, v, e;
	scanf("%d%d", &n, &m);
	Graph G;
	initGraph(&G, n);
	for (e = 1; e <= m; e++)
		scanf("%d%d", &u, &v), addEdge(&G, u, v);
	for (u = 1; u <= n; u++)
		if (color[u] == NO_COLOR)
	colorize(&G, u, BLACK);
	if (conflict)
	printf("KHONG DUOC");
	else printf("DUOC");
	return 0;
}
```

Ma trận A[100][100]

28.c

```
void dijkstra(Graph *pG, int s, int pi[], int p[]) {
int i, mark[MAXN];
for (i = 1; i <= pG->n; i++)
pi[i] = INF, p[i] = -1, mark[i] = 0;
pi[s] = 0;
int _;
for (_ = 0; _ < pG->n - 1; _++) {
int minPi = INF, u = -1, v;
for (v = 1; v <= pG->n; v++)
if (!mark[v] && pi[v] < minPi) {
minPi = pi[v];
u = v;
}
if (u == -1)
break;
mark[u] = 1;
for (v = 1; v <= pG->n; v++)
if (!mark[v] && pG->A[u][v] != NO_EDGE && pi[v] > pi[u] + pG->A[u][v])
pi[v] = pi[u] + pG->A[u][v], p[v] = u;
                    }
}
int main() {
int n, m, u, v, w, e;
scanf("%d%d", &n, &m);
Graph G;
initGraph(&G, n);

for (e = 1; e <= m; e++)
scanf("%d%d%d", &u, &v, &w), addEdge(&G, u, v, w);
int pi[MAXN], p[MAXN];
dijkstra(&G, 1, pi, p);
for (u = 2; u <= n; u++)
printf("%d %d %d\n", 1, v, pi[v]);
return 0;
}
```

29,c

```
void dijkstra(Graph *pG, int s, int pi[], int p[]) {
int i, mark[MAXN];
for (i = 1; i <= pG->n; i++)
pi[i] = INF, p[i] = -1, mark[i] = 0;
pi[s] = 0;
int _;
for (_ = 0; _ < pG->n - 1; _++) {
int minPi = INF, u = -1, v;
for (v = 1; v <= pG->n; v++)
if (!mark[v] && pi[v] < minPi) {
minPi = pi[v];
u = v;
}
if (u == -1)break;      mark[u] = 1;
for (v = 1; v <= pG->n; v++)
if (!mark[v] && pG->A[u][v] != NO_EDGE && pi[v] > pi[u] + pG->A[u][v])
pi[v] = pi[u] + pG->A[u][v], p[v] = u;
}   }
int main() {
int n, m, u, v, w, e;
scanf("%d%d", &n, &m);
Graph G; initGraph(&G, n);
for (e = 1; e <= m; e++)
scanf("%d%d%d", &u, &v, &w), addEdge(&G, u, v, w);
scanf("%d%d", &u, &v);
int pi[MAXN], p[MAXN];
dijkstra(&G, u, pi, p);
printf("%d\n", pi[v]);
int path[MAXN], len = 0;
while (v != -1) {
path[len++] = v;
v = p[v];
}
for (e = len - 1; e >= 0; e--)
printf("%d ", path[e]);   return 0;  }
```

30.c

```
void dijkstra(Graph *pG, int s, int pi[], int p[]) {
int i, mark[MAXN];
for (i = 1; i <= pG->n; i++)
pi[i] = INF, p[i] = -1, mark[i] = 0;
pi[s] = 0;
int _;
for (_ = 0; _ < pG->n - 1; _++) {
int minPi = INF, u = -1, v;
for (v = 1; v <= pG->n; v++)
if (!mark[v] && pi[v] < minPi) {
minPi = pi[v];
u = v;
}
if (u == -1) break;
mark[u] = 1;
for (v = 1; v <= pG->n; v++)
if (!mark[v] && pG->A[u][v] != NO_EDGE && pi[v] > pi[u] + pG->A[u][v])
pi[v] = pi[u] + pG->A[u][v], p[v] = u;
} }
int main() {
int n, m, u, v, w, e;
scanf("%d%d", &n, &m);
Graph G; initGraph(&G, n);
for (e = 1; e <= m; e++)
scanf("%d%d%d", &u, &v, &w), addEdge(&G, u, v, w);
scanf("%d%d", &u, &v);
```

Ma trận A[100][100]

```
int pi[MAXN], p[MAXN];
dijkstra(&G, u, pi, p);
printf("%d\n", pi[v]);
int path[MAXN], len = 0;
while (v != -1) {
path[len++] = v;
v = p[v];
}

for (e = len - 1; e >= 0; e--)
printf("%d ", path[e]);
return 0;
}
```

31.c

```
void dijkstra(Graph *pG, int s, int pi[], int p[]) {
int i, mark[MAXN];
for (i = 1; i <= pG->n; i++)
pi[i] = INF, p[i] = -1, mark[i] = 0;
pi[s] = 0;
int _;
for (_ = 0; _ < pG->n - 1; _++) {
int minPi = INF, u = -1, v;
for (v = 1; v <= pG->n; v++)
if (!mark[v] && pi[v] < minPi) {
minPi = pi[v];
u = v;
}
if (u == -1) break;
mark[u] = 1;
for (v = 1; v <= pG->n; v++)
if (!mark[v] && pG->A[u][v] != NO_EDGE && pi[v] > pi[u] + pG->A[u][v])
pi[v] = pi[u] + pG->A[u][v], p[v] = u;
                }
}
int main() {
int n, m, u, v, w, e;
scanf("%d%d", &n, &m);
Graph G; initGraph(&G, n);
for (e = 1; e <= m; e++)
scanf("%d%d%d", &u, &v, &w), addEdge(&G, u, v, w);
scanf("%d%d%d", &u, &v, &w);
int pi[MAXN], p[MAXN];
dijkstra(&G, u, pi, p);
if (pi[v] <= w)
printf("%d", w - pi[v]);
else printf("KHONG DI DUOC");
return 0;
}
```

35.c

```
int color[MAXN], conflict = 0;
void colorize(Graph *pG, int u, int c) {
color[u] = c;
int i;
for (i = 1; i <= pG->n; i++) {
if (pG->A[u][i]) {
if (color[i] == NO_COLOR)
colorize(pG, i, BLACK + WHITE - c);
else if (color[i] == color[u])
conflict = 1;
                }
            }
}
int main() {
int n, m, u, v, e;
scanf("%d%d", &n, &m);
Graph G;
initGraph(&G, n);
for (e = 1; e <= m; e++)
scanf("%d%d", &u, &v), addEdge(&G, u, v);
for (u = 1; u <= n; u++)
if (color[u] == NO_COLOR)
colorize(&G, u, BLACK);
if (conflict)
printf("IMPOSSIBLE");
else printf("OK");

                return 0;
}
```

41.c

```
int mark[MAXN];
void dfs(Graph *pG, int u) {
mark[u] = 1;
int i;
for (i = 1; i <= pG->n; i++)
if (!mark[i] && pG->A[u][i])
dfs(pG, i);
}
int main() {
int n, m, u, v, e;
scanf("%d%d", &n, &m);
Graph G;
initGraph(&G, n);
for (e = 1; e <= m; e++)
scanf("%d%d", &u, &v), addEdge(&G, u, v);
dfs(&G, 1);
for (u = 1; u <= n; u++)
if (!mark[u])
return !printf("NOT OK");
return !printf("OK");
}
```

Ma trận A[100][100]

4.2

```
int mark[MAXN];


void dfs(Graph *pG, int u) {
mark[u] = 1;
int i;
for (i = 1; i <= pG->n; i++)
if (!mark[i] && pG->A[u][i])
dfs(pG, i);
}
int main() {
int n, m, u, v, e;
scanf("%d%d", &n, &m);
Graph G;
initGraph(&G, n);
for (e = 1; e <= m; e++)
scanf("%d%d", &u, &v),
addEdge(&G, u, v);
dfs(&G, 1);
for (u = 1; u <= n; u++)
if (!mark[u])
return !printf("KHONG");
return !printf("DUOC");
}
```

445.c

```
int mark[MAXN];
void dfs(Graph *pG, int u) {
mark[u] = 1;
int i;
for (i = 1; i <= pG->n; i++)
if (!mark[i] && pG->A[u][i])
dfs(pG, i);
}
int main() {
int n, m, u, v, e;
scanf("%d%d", &n, &m);
Graph G; initGraph(&G, n);
for (e = 1; e <= m; e++)
scanf("%d%d", &u, &v),
addEdge(&G, u, v);
scanf("%d%d", &u, &v);
dfs(&G, u);
if (!mark[v])
return !printf("KHONG");
return !printf("DUOC");
}
```

46.c

```
int mark[MAXN];


void dfs(Graph *pG, int u) {
mark[u] = 1;
int i;
for (i = 1; i <= pG->n; i++)
if (!mark[i] && pG->A[u][i])
dfs(pG, i);
}
int main() {
int n, m, u, v, e;
scanf("%d%d", &n, &m);
Graph G;
initGraph(&G, n);
for (e = 1; e <= m; e++)
scanf("%d%d", &u, &v),
addEdge(&G, u, v);
scanf("%d%d", &u, &v);
dfs(&G, u);
if (!mark[v])
return !printf("KHONG");
return !printf("DUOC");
}
```

47.c

```
int degree(Graph *pG, int u) {
int d = 0, v;
for (v = 1; v <= pG->n; v++)
d += (pG->A[v][u]);
return d;
}
int main() {
int n, m, u, v, e;
scanf("%d%d", &n, &m);
Graph G;
initGraph(&G, n);


for (e = 1; e <= m; e++)
scanf("%d%d", &u, &v),
addEdge(&G, u, v);
u = 1;
for (v = 1; v <= n; v++)
u = (degree(&G, u) >  degree(&G, v)
? u : v);
printf("%d", degree(&G, u));
return 0;
}
```

Ma trận A[100][100]