```
int i, has = 0;
                                                                                                                    scanf("%d%d", &n, &m);
                                                               for (i = 1; i <= pG->n; i++)
                                                                                                                    Graph G;
int hasCycle = 0, color[MAXN];
                                                                          if (pG->A[u][i] &&
                                                                                                                    initGraph(&G, n);
void dfs(Graph *pG, int u) {
                                                    pG->A[v][i])
                                                                                                                    for (e = 1; e <= m; e++)
           color[u] = GRAY;
                                                                                                                               scanf("%d%d",
                                                               printf("%d ", i), has = 1;
           int i;
                                                                                                         &u, &v), addEdge(&G, u, v);
                                                               if (!has)
           for (i = 1; i <= pG->n; i++)
                                                                                                                    scanf("%d", &u);.
                                                                           printf("KHONG
                      if (pG->A[u][i] \&\&
                                                                                                                    printf("%d:%d",
                                                                                                                                                u,
                                                    CHUNG DOI THU");
color[i] == WHITE)
                                                                                                         outDegree(&G, u));
                                                    }
                                 dfs(pG,
                                                                                                                    return 0;
i);
                                                    int main() {
                                                                                                        }
                      else if (pG->A[u][i]
                                                               int n, u, v, w;
&& color[i] == GRAY)
                                                                                                         4.
                                                               Graph G;
                                                                                                        int outDegree(Graph *pG, int u) {
           hasCycle = 1;
                                                               scanf("%d", &n);
                                                                                                                    int d = 0, v;
           color[u] = BLACK;
                                                               initGraph(&G, n);
                                                                                                                    for (v = 1; v <= pG->n; v++)
}
                                                               for (u = 1; u <= n; u++)
int main() {
                                                                                                                               d += (pG->A[v][u]);
                                                                          for (v = 1; v \le n;
                                                    v++) {
                                                                                                                    return d;
           int n, m, u, v, e;
                                                                                                         }
           scanf("%d%d", &n, &m);
                                                               scanf("%d", &w);
                                                                                                        int main() {
           Graph G;
                                                                                      if
                                                    (w)addEdge(&G, u, v);
           initGraph(&G, n);
                                                                                                                    int n, u, w, i, j;
                                                                                                                    scanf("%d", &n);
           for (e = 1; e <= m; e++)
                                                               scanf("%d%d", &u, &v);
                                                                                                                    Graph G;
                      scanf("%d%d",
&u, &v), addEdge(&G, u, v);
                                                               check(&G, u, v);
                                                                                                                    initGraph(&G, n);
           for (u = 1; u \le n; u++)
                                                               return 0;
                                                                                                                    for (i = 1; i <= n; i++)
                      if (color[u] ==
                                                                                                                               for (j = 1; j \le n;
WHITE)
                                                                                                        j++) {
           dfs(&G, u);
                                                                                                                    scanf("%d", &w);
                                                    int outDegree(Graph *pG, int u) {
           if (!hasCycle)
                                                                                                                                          if
                                                               int d = 0, v;
                                                                                                         (w)addEdge(&G, i, j);
                      printf("HOP LE");
                                                               for (v = 1; v <= pG->n; v++)
                                                                                                                               }
           else printf("KHONG");
                                                                          d += (pG->A[v][u]);
                                                                                                                    scanf("%d", &u);
           return 0;
                                                               return d;
                                                                                                                    printf("%d:%d",
}
                                                                                                         outDegree(&G, u));
                                                    }
2.
                                                                                                                    return 0;
                                                    int main() {
void check(Graph *pG, int u, int v) {
                                                                                                         }
                                                               int n, m, u, v, e;
```

```
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 \le n;
u++)
(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 \le n; j++)
            scanf("%d", &G.A[i][j]);
            scanf("%d", &u);
            printf("%d:%d", u, degree(&G, u));
            return 0;
```

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 \le pG -> n;
v++)
(!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 \le 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++)
                             (color[u]
NO COLOR)
```

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;
}
```

```
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 \le 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 \le 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 \le 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;
}
```

```
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,
                              &٧,
                                     &w),
addEdge(&G, u, v, w);
int pi[MAXN], p[MAXN];
dijkstra(&G, 1, pi, p);
```

for $(u = 2; u \le n; u++)$

```
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 \le 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;
```

```
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 \le 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 \le 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);
```

while (v != -1) {

```
int pi[MAXN], p[MAXN];
                                                     }
                                                                                                           initGraph(&G, n);
dijkstra(&G, u, pi, p);
                                                     int main() {
                                                                                                           for (e = 1; e <= m; e++)
                                                                                                           scanf("%d%d", &u, &v), addEdge(&G, u,
printf("%d\n", pi[v]);
                                                     int n, m, u, v, w, e;
int path[MAXN], len = 0;
                                                     scanf("%d%d", &n, &m);
                                                                                                           for (u = 1; u <= n; u++)
while (v != -1) {
                                                     Graph G; initGraph(&G, n);
                                                                                                           if (color[u] == NO COLOR)
path[len++] = v;
                                                     for (e = 1; e <= m; e++)
                                                                                                           colorize(&G, u, BLACK);
v = p[v];
                                                     scanf("%d%d%d", &u,
                                                                                   &ν,
                                                                                          &w),
                                                     addEdge(&G, u, v, w);
                                                                                                           if (conflict)
                                                     scanf("%d%d%d", &u, &v, &w);
                                                                                                           printf("IMPOSSIBLE");
                                                     int pi[MAXN], p[MAXN];
                                                                                                           else printf("OK");
for (e = len - 1; e >= 0; e--)
                                                     dijkstra(&G, u, pi, p);
printf("%d ", path[e]);
                                                     if (pi[v] \le w)
                                                                                                                             return 0;
return 0;
                                                     printf("%d", w - pi[v]);
                                                                                                           }
                                                     else printf("KHONG DI DUOC");
                                                                                                           41.c
31.c
                                                     return 0;
                                                                                                           int mark[MAXN];
void dijkstra(Graph *pG, int s, int pi[], int
                                                     }
                                                                                                           void dfs(Graph *pG, int u) {
p[]) {
                                                     35.c
                                                                                                           mark[u] = 1;
int i, mark[MAXN];
                                                                                                           int i;
for (i = 1; i <= pG->n; i++)
                                                                                                           for (i = 1; i <= pG->n; i++)
                                                     int color[MAXN], conflict = 0;
pi[i] = INF, p[i] = -1, mark[i] = 0;
                                                                                                           if (!mark[i] && pG->A[u][i])
                                                     void colorize(Graph *pG, int u, int c) {
pi[s] = 0;
                                                                                                           dfs(pG, i);
                                                     color[u] = c;
int _;
                                                                                                           }
                                                     int i;
for (_ = 0; _ < pG->n - 1; _ ++) {
                                                                                                           int main() {
                                                     for (i = 1; i \le pG -> n; i++) \{
int minPi = INF, u = -1, v;
                                                                                                           int n, m, u, v, e;
                                                     if (pG->A[u][i]) {
for (v = 1; v \le pG -> n; v++)
                                                                                                           scanf("%d%d", &n, &m);
                                                     if (color[i] == NO_COLOR)
if (!mark[v] && pi[v] < minPi) {
                                                                                                           Graph G;
                                                     colorize(pG, i, BLACK + WHITE - c);
minPi = pi[v];
                                                                                                           initGraph(&G, n);
                                                     else if (color[i] == color[u])
u = v;
                                                                                                           for (e = 1; e <= m; e++)
                                                     conflict = 1;
                                                                                                           scanf("%d%d", &u, &v), addEdge(&G, u,
if (u == -1) break;
                                                                                                           v);
mark[u] = 1;
                                                                                                           dfs(&G, 1);
for (v = 1; v <= pG->n; v++)
                                                                                                           for (u = 1; u <= n; u++)
                                                     int main() {
if (!mark[v] \&\& pG->A[u][v] != NO_EDGE
                                                                                                           if (!mark[u])
&& pi[v] > pi[u] + pG -> A[u][v]
                                                     int n, m, u, v, e;
                                                                                                           return !printf("NOT OK");
pi[v] = pi[u] + pG->A[u][v], p[v] = u;
                                                     scanf("%d%d", &n, &m);
                                                                                                           return !printf("OK");
                 }
                                                     Graph G;
```

for $(i = 1; i \le pG -> n; i++)$

if (!mark[i] && pG->A[u][i]) for (e = 1; e <= m; e++) 4.2 dfs(pG, i); scanf("%d%d", &u, &v), int mark[MAXN]; addEdge(&G, u, v); } scanf("%d%d", &u, &v); int main() { void dfs(Graph *pG, int u) { dfs(&G, u); int n, m, u, v, e; mark[u] = 1;if (!mark[v]) scanf("%d%d", &n, &m); int i; return !printf("KHONG"); Graph G; initGraph(&G, n); return !printf("DUOC"); for $(i = 1; i \le pG -> n; i++)$ for (e = 1; e <= m; e++) if (!mark[i] && pG->A[u][i]) scanf("%d%d", &u, &v), addEdge(&G, u, v); 47.c dfs(pG, i); scanf("%d%d", &u, &v); } int degree(Graph *pG, int u) { dfs(&G, u); int main() { int d = 0, v; if (!mark[v]) int n, m, u, v, e; for (v = 1; v <= pG->n; v++) return !printf("KHONG"); scanf("%d%d", &n, &m); d += (pG->A[v][u]);return !printf("DUOC"); return d; Graph G; } initGraph(&G, n); 46.c int main() { for (e = 1; e <= m; e++) int mark[MAXN]; int n, m, u, v, e; scanf("%d%d", &u, &v), addEdge(&G, u, v); scanf("%d%d", &n, &m); void dfs(Graph *pG, int u) { Graph G; dfs(&G, 1); mark[u] = 1;initGraph(&G, n); for $(u = 1; u \le n; u++)$ int i; if (!mark[u]) for $(i = 1; i \le pG -> n; i++)$ for (e = 1; e <= m; e++) return !printf("KHONG"); if (!mark[i] && pG->A[u][i]) scanf("%d%d", &u, &v), return !printf("DUOC"); addEdge(&G, u, v); dfs(pG, i); } u = 1;} 445.c for (v = 1; v <= n; v++) int main() { u = (degree(&G, u) > degree(&G, v)int mark[MAXN]; int n, m, u, v, e; ? u : v); void dfs(Graph *pG, int u) { scanf("%d%d", &n, &m); printf("%d", degree(&G, u)); mark[u] = 1;Graph G; return 0; int i; initGraph(&G, n); }

<mark>Ma trận A[100][100]</mark>