```
// Constantes e variáveis globais
#define MAX 6
#define INF 9999

int grafo[MAX][MAX];
char vertices[MAX] = {'A', 'B', 'C', 'D', 'E', 'F'};
bool visitado[MAX];

// Reset do vetor de visitados
void resetVisitado() {
    for (int i = 0; i < MAX; i++) {
        visitado[i] = false;
    }
}

// Leitura manual do grafo
void lerGrafoManual() {
    printf("Digite a matriz de adjacência (%dx%d):\n", MAX, MAX);
    for (int i = 0; i < MAX; i++) {
        for (int j = 0; j < MAX; j++) {
            scanf("%d", &grafo[i][j]);
        }
    }
}

// Leitura de grafo por arquivo
void lerGrafoArquivo() {
    char nomeArquivo[100];
    printf("Digite o nome do arquivo (ex: grafo.txt): ");
    scanf("%s", nomeArquivo);

    FILE *arquivo = fopen(nomeArquivo, "r");
    if (arquivo == NULL) {
        printf("Erro ao abrir o arquivo.\n");
        return;
    }

    for (int i = 0; i < MAX; i++) {
        for (int j = 0; j < MAX; j++) {
            fscanf(arquivo, "%d", &grafo[i][j]);
        }
    }
    fclose(arquivo);
    printf("Grafo carregado com sucesso!\n");
}

// DFS
void dfs(int v) {
    visitado[v] = true;
    printf("%c ", vertices[v]);
    for (int i = 0; i < MAX; i++) {
        if (grafo[v][i] && !visitado[i]) {
            dfs(i);
        }
```

```
    }
}

// BFS
void bfs(int inicio) {
    bool visitadoLocal[MAX] = {false};
    int fila[MAX], frente = 0, tras = 0;
    int nivel[MAX] = {0};

    visitadoLocal[inicio] = true;
    fila[tras++] = inicio;

    printf("Ordem de visita da BFS:\n");
    while (frente < tras) {
        int v = fila[frente++];
        printf("%c - nível = %d\n", vertices[v], nivel[v]);
        for (int i = 0; i < MAX; i++) {
            if (grafo[v][i] && !visitadoLocal[i]) {
                visitadoLocal[i] = true;
                fila[tras++] = i;
                nivel[i] = nivel[v] + 1;
            }
        }
    }
}

// Dijkstra
void dijkstra(int inicio) {
    int dist[MAX];
    bool visitadoLocal[MAX] = {false};
    for (int i = 0; i < MAX; i++) dist[i] = INF;
    dist[inicio] = 0;
    for (int c = 0; c < MAX - 1; c++) {
        int min = INF, u = -1;
        for (int i = 0; i < MAX; i++) {
            if (!visitadoLocal[i] && dist[i] < min) {
                min = dist[i];
                u = i;
            }
        }
        if (u == -1) break;
        visitadoLocal[u] = true;
        for (int v = 0; v < MAX; v++) {
            if (!visitadoLocal[v] && grafo[u][v] && dist[u] + grafo[u][v] < dist[v]) {
                dist[v] = dist[u] + grafo[u][v];
            }
        }
    }
    printf("Menor distância do vértice %c para os outros vértices:\n", vertices[inicio]);
    for (int i = 0; i < MAX; i++) {
        if (dist[i] == INF)
            printf("%c -> %c = Infinito (sem caminho)\n", vertices[inicio], vertices[i]);
        else
            printf("%c -> %c = %d\n", vertices[inicio], vertices[i], dist[i]);
    }
```

```
}

// Verificar se é conexo
bool ehConexo(int grafo[MAX][MAX], int n) {
    int visitado[MAX] = {0};
    dfs(grafo, n, visitado, 0);
    for (int i = 0; i < n; i++) {
        if (!visitado[i]) return false;
    }
    return true;
}

// Verificar se é bipartido
bool ehBipartido(int grafo[MAX][MAX], int n) {
    int cor[MAX];
    for (int i = 0; i < n; i++) cor[i] = -1;
    cor[0] = 1;
    int fila[MAX], frente = 0, tras = 0;
    fila[tras++] = 0;
    while (frente < tras) {
        int u = fila[frente++];
        for (int v = 0; v < n; v++) {
            if (grafo[u][v] && cor[v] == -1) {
                cor[v] = 1 - cor[u];
                fila[tras++] = v;
            } else if (grafo[u][v] && cor[v] == cor[u]) {
                return false;
            }
        }
    }
    return true;
}

// Função principal
int main() {
    setlocale(LC_ALL, "");
    int opcao;
    do {
        printf("\n------------------ MENU PRINCIPAL ------------------\n");
        printf("1 - Busca em Profundidade (DFS)\n");
        printf("2 - Busca em Largura (BFS)\n");
        printf("3 - Dijkstra (Menor Caminho)\n");
        printf("0 - Sair\n");
        printf("----------------------------------------------------\n");
        printf("Escolha uma opção: ");
        scanf("%d", &opcao);
        if (opcao == 0) {
            printf("Encerrando o programa...\n");
            break;
        }
        escolherEntrada();
        switch (opcao) {
            case 1:
                resetVisitado();
                printf("Visita DFS a partir do vértice A:\n");
```

```
                dfs(0);
                printf("\n");
                break;
            case 2:
                bfs(0);
                break;
            case 3:
                dijkstra(0);
                break;
            default:
                printf("Opção inválida!\n");
        }
    } while (opcao != 0);
    return 0;
}
```