Оваа програма имплементира три основни алгоритми за решавање на проблемот со максимален проток (Max Flow) во граф:

1. Edmonds-Karp алгоритам : Оваа верзија на Ford-Fulkerson алгоритмот користи BFS (Breadth-First Search) за да пронајде аугментирачки патеки и да го максимизира протокот.

2. Ford-Fulkerson алгоритам : Користи DFS (Depth-First Search) за да најде патеки за зголемување на протокот.

3. Dinic's алгоритам : Унапреден алгоритам кој користи комбинација од BFS и DFS за да се справи со блокирачките текови и да го оптимизира пресметувањето на протокот.

Програмата нуди две претстави на графовите:

- Листа на соседство (Adjacency List) : Каде секој јазол има листа на поврзани јазли.

- Матрица на соседство (Adjacency Matrix) : Каде графот е претставен како матрица, со директно наведување на поврзаноста меѓу јазлите.

Целта на програмата е да се најде максималниот можен проток од одреден извор (source) до одреден краен јазол (sink) во даден граф, користејќи ги овие алгоритми.

1. Класата за граф (Graph Class)

Основната класа Graph е дизајнирана за да ја претстави основната структура на графот. Таа се проширува преку две подкласи:

- GraphAdjList : Претставува граф со листа на соседство.

- GraphAdjMatrix : Претставува граф со матрица на соседство.

* Атрибути:

- n : Број на јазли во графот.

- Capacity : Матрица која ги чува капацитетите на секој раб во графот.

- capacity : Матрица која се користи за пресметување на преостанатиот капацитет на рабовите во текот на извршувањето.

- adj : Структура која го претставува графот, преку листа или матрица на соседство

* Методи:

- Конструктор/Деструктор : Овие методи го подготвуваат и ослободуваат динамички доделената меморија за графот.

- build\_graph : Методот ги внесува рабовите и нивните капацитети за да го изгради графот.

- solveEdmondKarp : Користи BFS за пронаоѓање на патеки и пресметување на максималниот проток.

- solveFordFulkerson : Користи DFS за пронаоѓање на патеки за максимизирање на протокот.

- solveDinic : Имплементира Dinic’s алгоритам, кој ја комбинира ефикасноста на BFS и DFS.

- solve : Оваа функција ги извршува сите три алгоритми (Edmonds-Karp, Ford-Fulkerson и Dinic) и ги прикажува резултатите.

**2. Претставување на графови**

Програмата овозможува графовите да бидат претставени на два различни начини, преку две подкласи:

* GraphAdjList (Листа на соседство) :

- add\_edge : Додава нов раб помеѓу јазли.

- bfs : Пребарува аугментирачки патеки користејќи BFS за Edmonds-Karp алгоритмот.

- dfs : Пребарува аугментирачки патеки користејќи DFS за Ford-Fulkerson.

* GraphAdjMatrix (Матрица на соседство) :

- add\_edge : Додава нови рабови во матрицата.

- bfs и dfs : Слични на методите во GraphAdjList, но користат индекси за пристап до соседните јазли преку матрицата.

**3. Алгоритми за максимален проток**

* Edmonds-Karp Алгоритам :

- Овој алгоритам користи BFS за да пронајде патеки за зголемување на протокот во графот.

- Преку секоја пронајдена патека, се одредува минималниот и се зголемува протокот за таа вредност.

- Временска сложеност: O(VE²) (V е бројот на јазли, E е бројот на рабови).

* Ford-Fulkerson Алгоритам :

- Алгоритамот користи DFS за да бара патеки за зголемување.

- Секоја патека се користи за да се зголеми протокот, а капацитетите се ажурираат по секоја итерација.

- Временска сложеност: O(max\_flow \* E) .

* Dinic’s Алгоритам :

- Прво, се гради график на нивоа со BFS, каде секој јазол се поврзува на друг според нивото на оддалеченост од изворот.

- Потоа DFS ги наоѓа блокирачките текови , каде што се заситуваат рабовите низ графот.

- Временска сложеност: O(V²E) , значително побрз за некои типови на графови.

**4. Пресметка на Min-Cut**

Минималниот пресек (Min-Cut) на графот се одредува по пресметка на максималниот проток. Следувајќи ги чекорите:

1. Пресметка на Max Flow : Програмата го користи Edmonds-Karp алгоритмот за пресметување на максималниот проток.

2. BFS за достапност : Се извршува BFS од изворниот јазол за да се откријат сите јазли што сè уште може да се достигнат од изворот.

3. Идентификување на Min-Cut рабови : Рабовите кои поврзуваат достапни јазли со недостапни јазли се идентификуваат како пресек. Овие рабови претставуваат тесно грло.

4. Печатење на Min-Cut : Функцијата `print\_MinCut` ги печати овие критични рабови.

5. Кориснички внес и извршување

* Главната програма ги повикува следниве чекори:

1. Корисникот го внесува бројот на графови.

2. За секој граф, се внесува бројот на јазли, бројот на рабови и типот на претставување (листа или матрица).

3. Корисникот ги внесува рабовите и капацитетите.

4. Програмата го извршува секој алгоритам за максимален проток и ги прикажува резултатите.

5. Програмата исто така го идентификува и прикажува минималниот пресек (Min-Cut).

* Заклучок

Оваа програма нуди комплетно решение за проблемите на Max Flow и Min-Cut во графови, користејќи три различни алгоритми: Edmonds-Karp, Ford-Fulkerson и Dinic’s. Програмата овозможува корисникот да избере помеѓу два различни начина за претставување на графот (листа на соседство или матрица на соседство), што овозможува флексибилност во користењето на различни типови графови.

Секој од алгоритмите го решава проблемот со максимален проток, додека функцијата Min-Cut помага да се идентификуваат критичните рабови кои го ограничуваат протокот.