

МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. ІВАНА ФРАНКА
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ ТА ІНФОРМАТИКИ
КАФЕДРА ПРОГРАМУВАННЯ

ЗВІТ

до завдання № 1

із штучного інтелекту

BIDIRECTIONAL SEARCH

Виконав:

студент ПМІ 41

Крупич Андрій

Львів – 2012

1. Алгоритм

Bidirectional search в абсолютній більшості випадків – найкращий серед алгоритмів неінформованого пошуку. По суті він представляє uniform-cost search, але проводить обчислення одночасно з обох кінців шуканого шляху, що дозволяє значно скоротити час та ресурси для пошуку. Працює алгоритм зі зваженими орієнтованими графами.

У порівнянні з пошуком вглиб, bidirectional search виконує більше операцій, але завжди знаходить оптимальний шлях. Організовуючи bidirectional search через пошук вглиб, ми б наткнулись на проблему «проходження повз» і нестиковки шляхів із заданих вершин, оскільки пошук вглиб вибирає довільну вершину кожного наступного рівня.

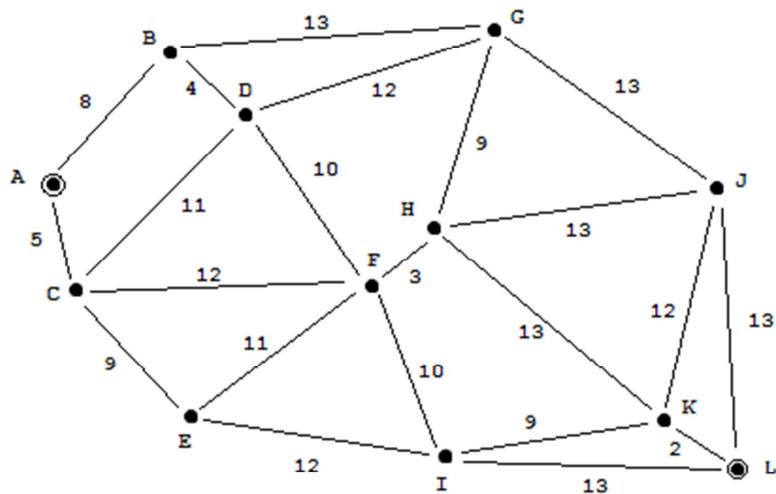
Порівнюючи з пошуком вшир, bidirectional search здійснює в рази менше операцій (складності цих алгоритмів відрізняються половинним степеневим показником), а при густій сітці пошук вглиб взагалі не варто розглядати як альтернативу bidirectional search.

Алгоритми, що базуються на цих типах пошуків, теж в більшості випадків програють bidirectional search як у часі на пошук, так і у витрачених ресурсах.

Формально алгоритм можна описати так:

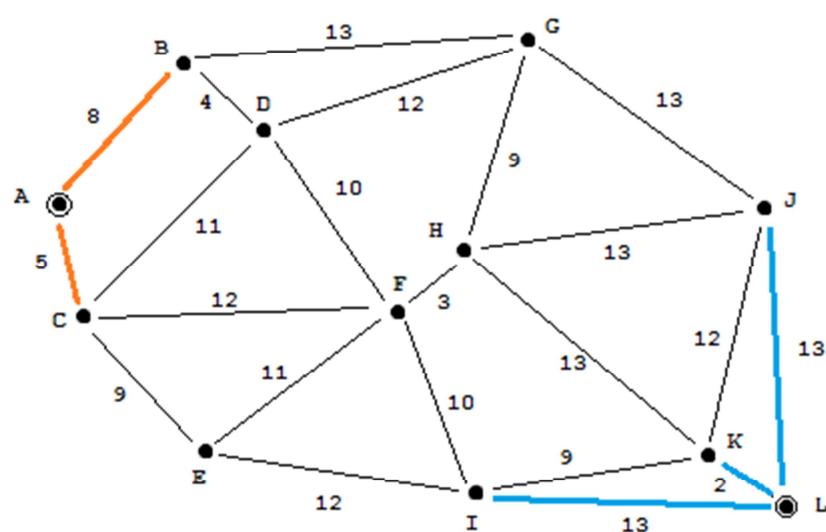
1. Вибираємо всі шляхи первого рівня, що починаються в початковій вершині або закінчуються в кінцевій і заносимо в список шляхів.
2. Вибираємо зі списку шляхів шлях найменшої довжини, вилучаємо його і заносимо у список оптимальних шляхів.
3. Якщо список оптимальних шляхів містить шляхи з початкової та кінцевої вершини, що замикаються на певній вершині, то складений з них шлях є результатом; в цьому разі робота алгоритму завершена.
4. Формуємо список із шляхів наступного рівня, що продовжують вибраний оптимальний шлях у прямому або оберненому порядку (залежно від напрямленості обраного шляху) і вносимо до загального списку шляхів.
5. Фільтруємо список, видаляючи неоптимальні шляхи до однакових вершин і переходимо до 2.

Наглядніше дію алгоритму можна побачити на прикладі:



Знайдемо шлях від вершини A до вершини L.

:



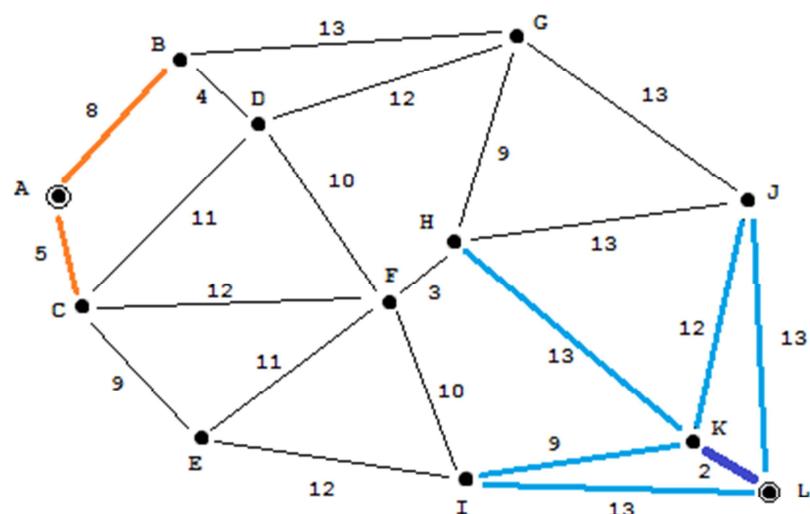
$$AB=8$$

$$AC=5$$

$$LJ=13$$

$$\underline{LK}=2$$

$$LI=13$$



$$AB=8$$

$$AC=5$$

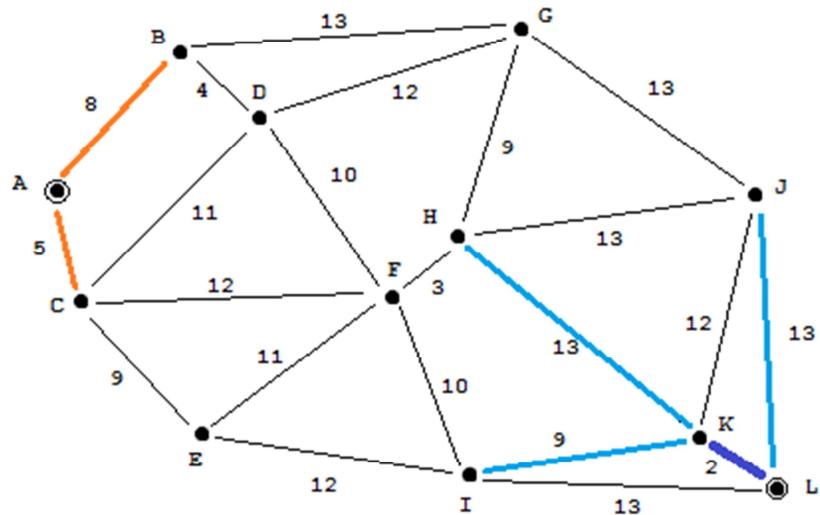
$$LJ=13$$

$$*LI=13$$

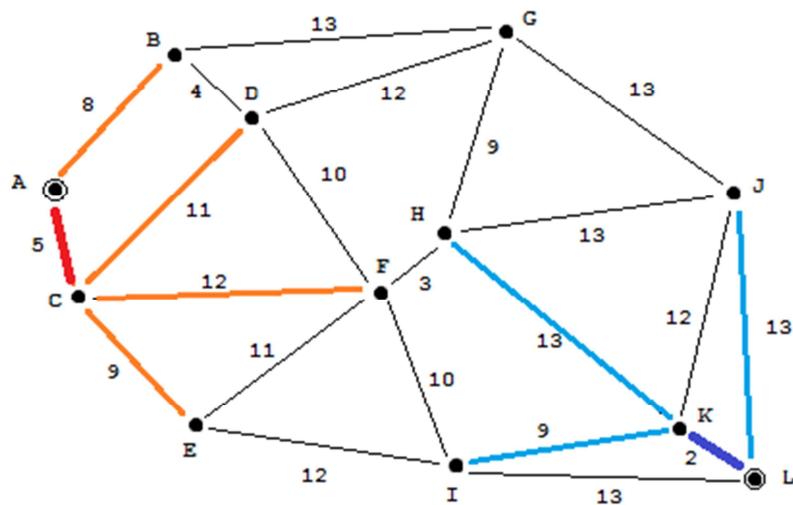
$$*LKJ=14$$

$$LKH=15$$

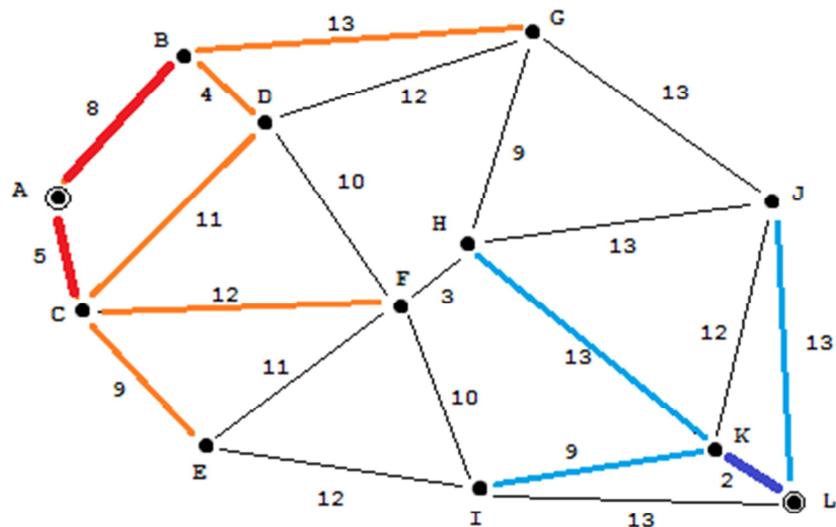
$$LKI=11$$



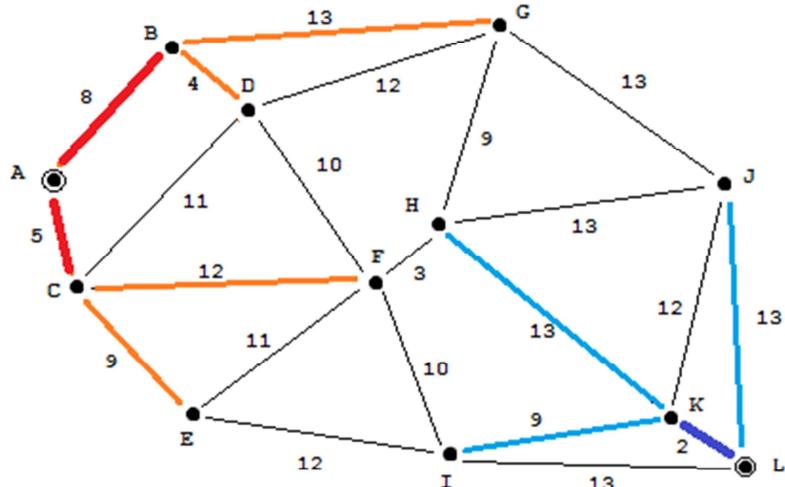
AB=8
AC=5
LJ=13
LKH=15
LKI=11



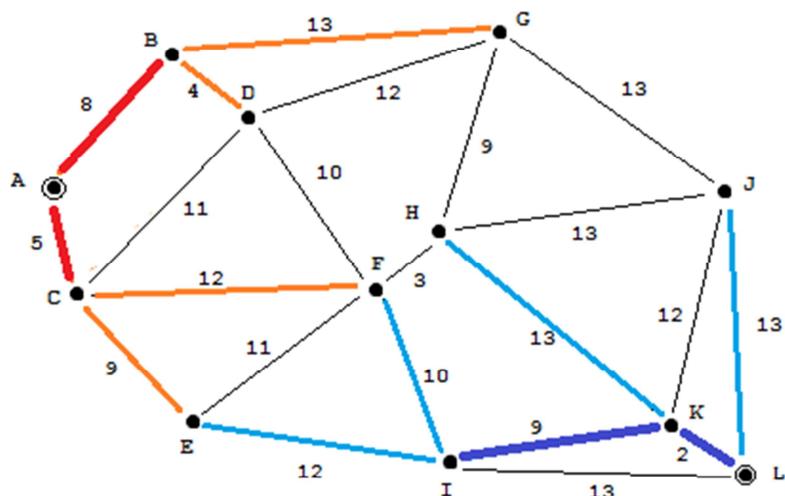
AB=8
LJ=13
LKH=15
LKI=11
ACD=16
ACF=17
ACE=14



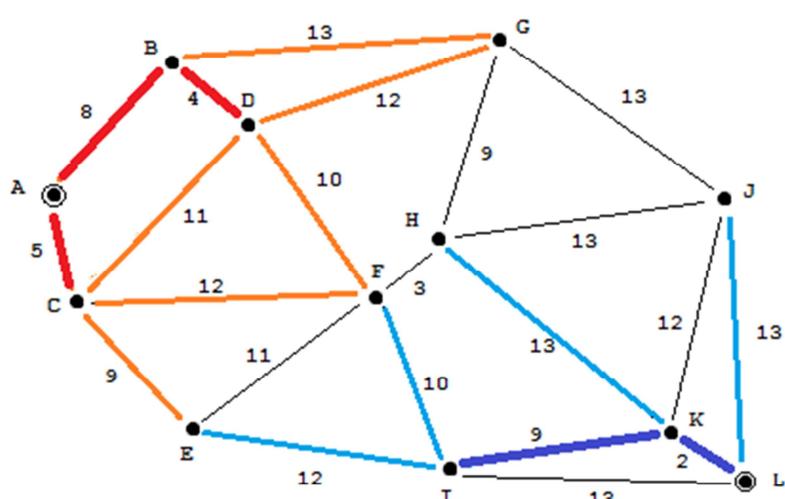
LJ=13
LKH=15
LKI=11
*ACD=16
ACF=17
ACE=14
ABG=21
ABD=12



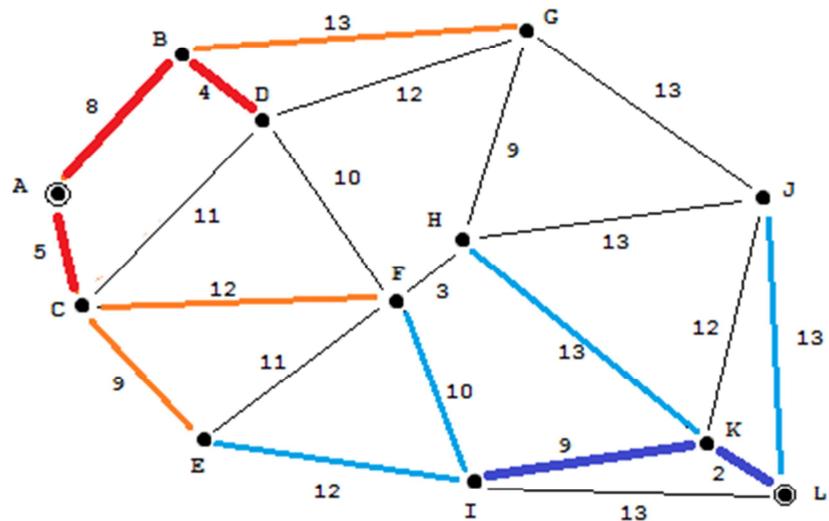
LJ=13
LKH=15
LKI=11
ACF=17
ACE=14
ABG=21
ABD=12



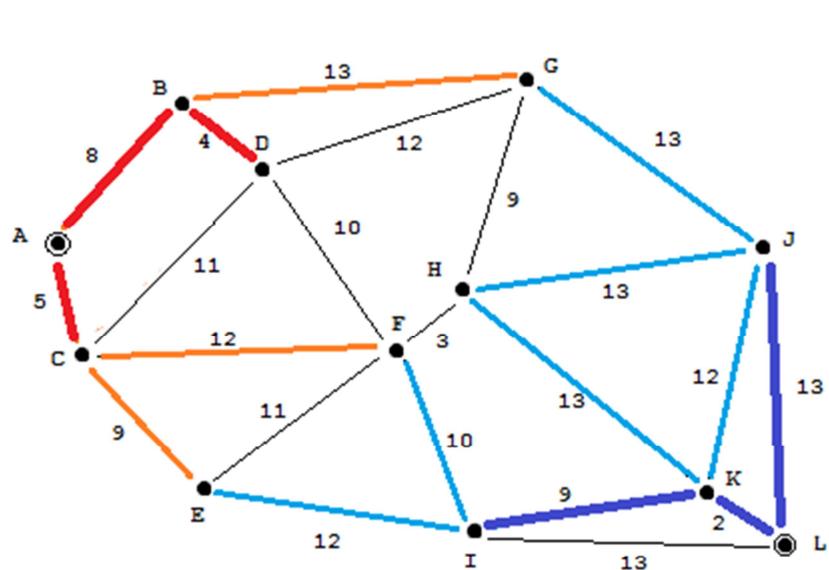
LJ=13
LKH=15
ACF=17
ACE=14
ABG=21
ABD=12
LKIF=21
LKIE=23



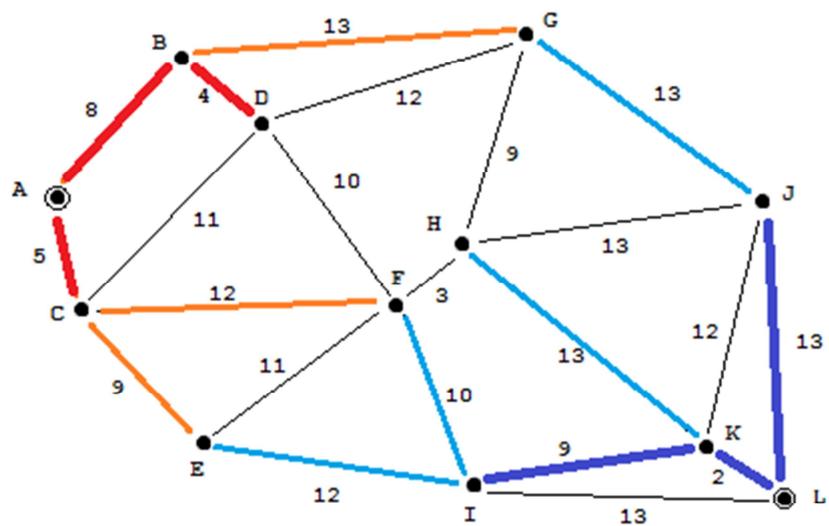
LJ=13
LKH=15
ACF=17
ACE=14
ABG=21
LKIF=21
LKIE=23
*ABDG=24
*ABDF=22
*ABDC=23



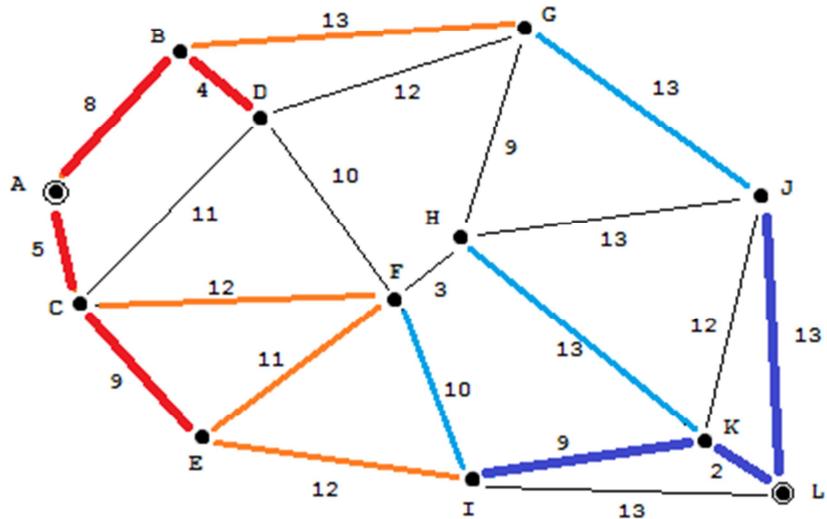
LJ=13
 LKH=15
 ACF=17
 ACE=14
 ABG=21
 LKIF=21
 LKIE=23



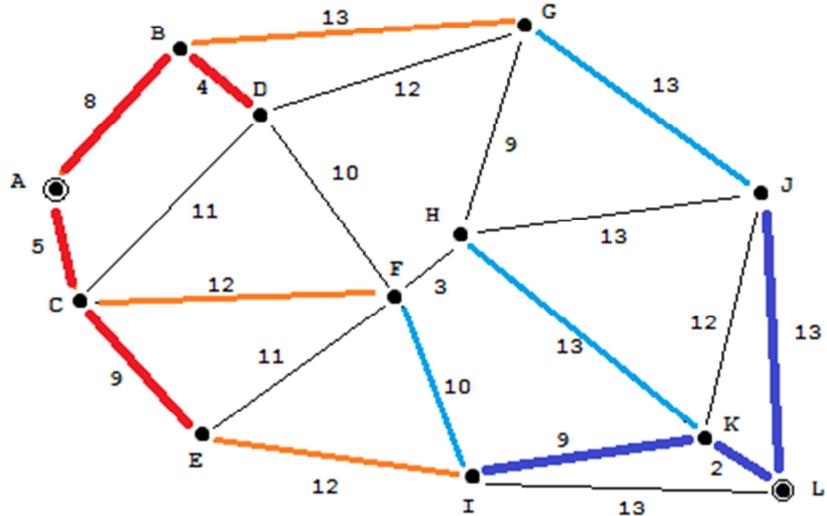
LKH=15
 ACF=17
 ACE=14
 ABG=21
 LKIF=21
 LKIE=23
 LJG=26
 *LJH=26
 *LJK=25



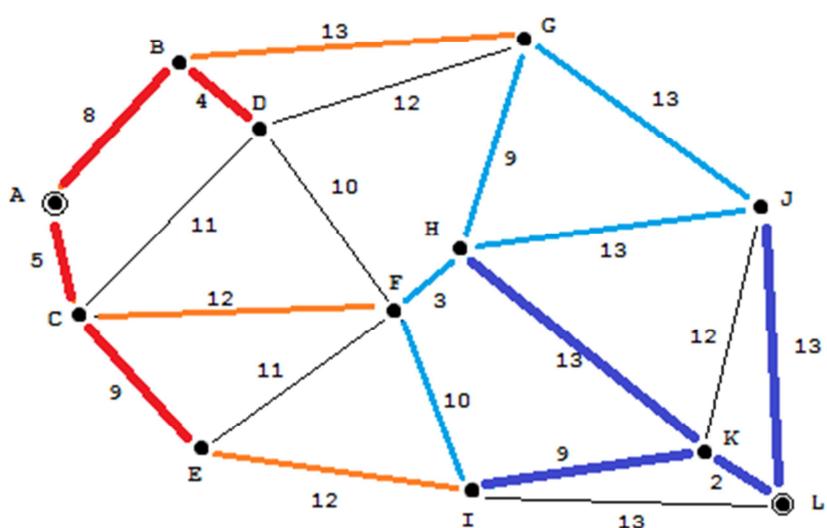
LKH=15
ACF=17
ACE=14
 ABG=21
 LKIF=21
 LKIE=23
 LJG=26



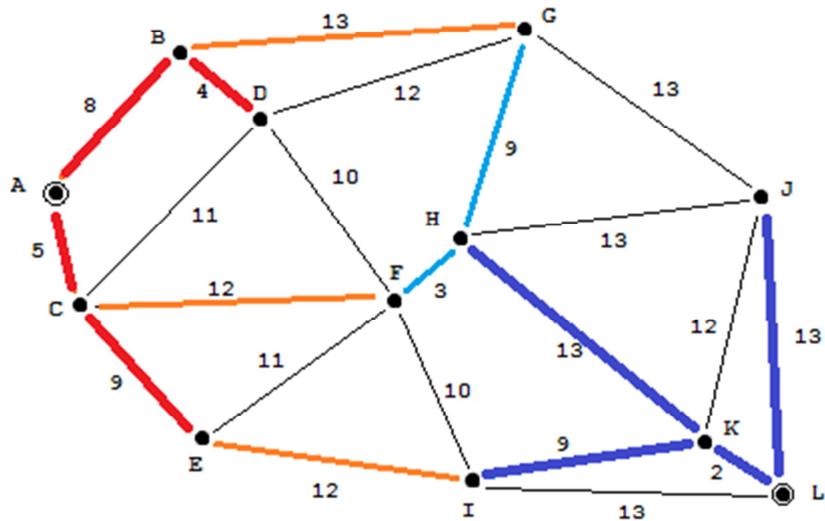
LKH=15
 ACF=17
 ABG=21
 LKIF=21
 LKIE=23
 LJG=26
 *ACEF=25
 ACEI=26



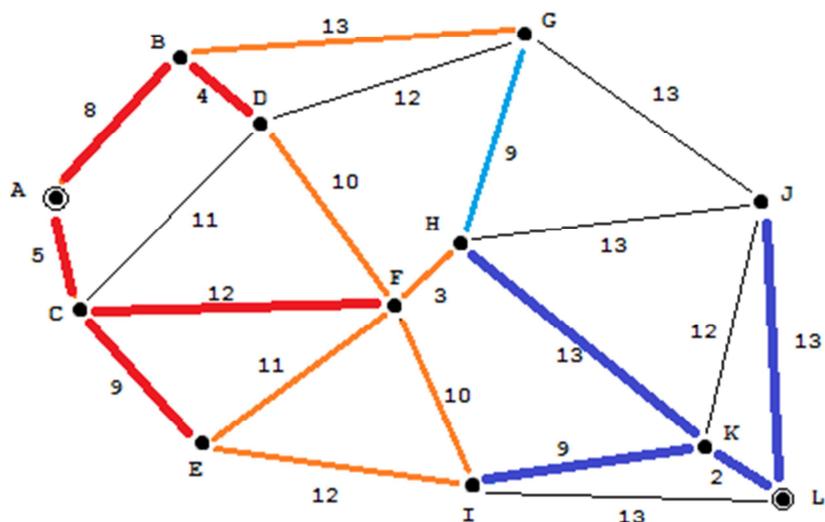
LKH=15
 ACF=17
 ABG=21
 LKIF=21
 LKIE=23
 LJG=26
 ACEI=26



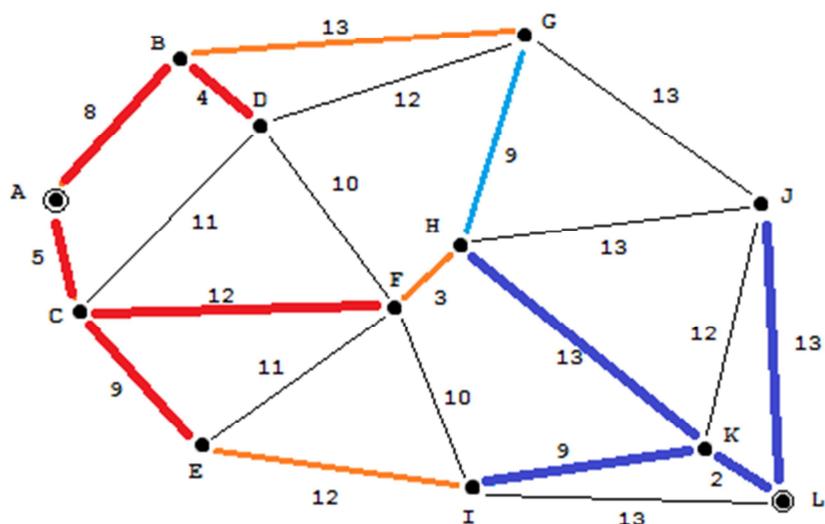
ACF=17
 ABG=21
 *LKIF=21
 LKIE=23
 *LJG=26
 ACEI=26
 *LKHJ=28
 LKHG=24
 LKHF=18



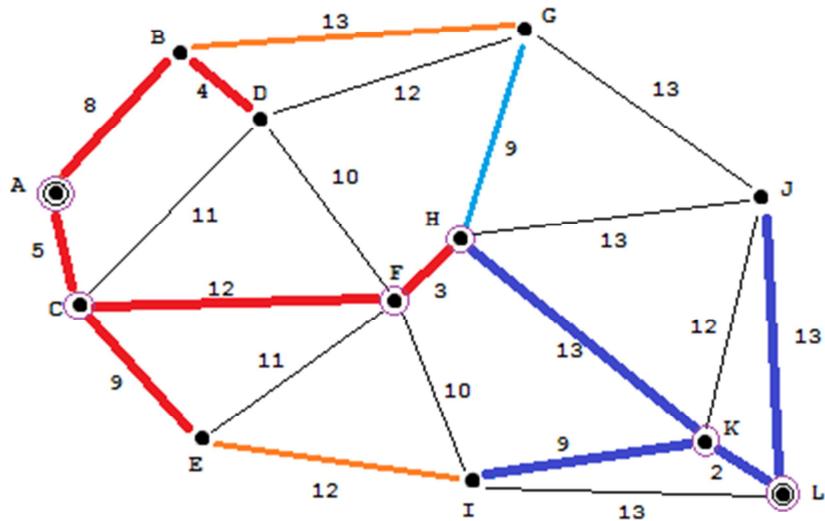
ACF=17
~~ABG=21~~
 LKIE=23
~~ACEI=26~~
 LKHG=24
 LKHF=18



~~ABG=21~~
 LKIE=23
~~ACEI=26~~
 LKHG=24
~~*LKHF=18~~
~~*ACFD=27~~
 ACFH=20
~~*ACFI=27~~
~~*ACFE=28~~



~~ABG=21~~
 LKIE=23
~~ACEI=26~~
 LKHG=24
ACFH=20

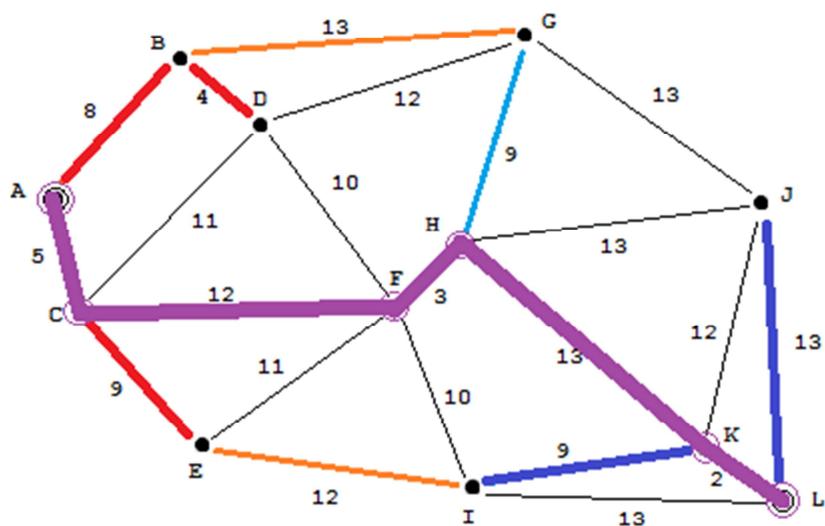


ABG=21

LKIE=23

ACEI=26

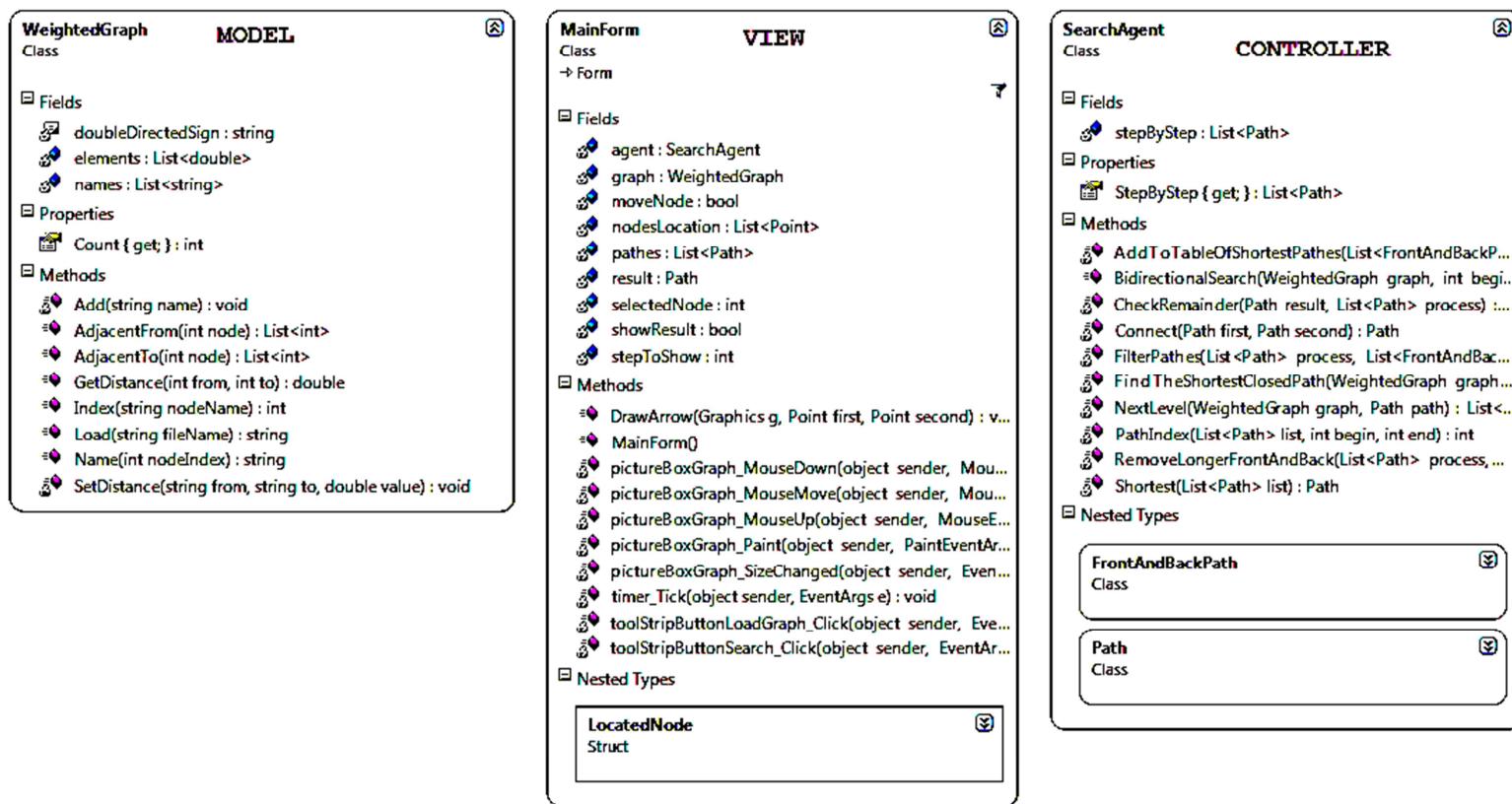
LKHG=24



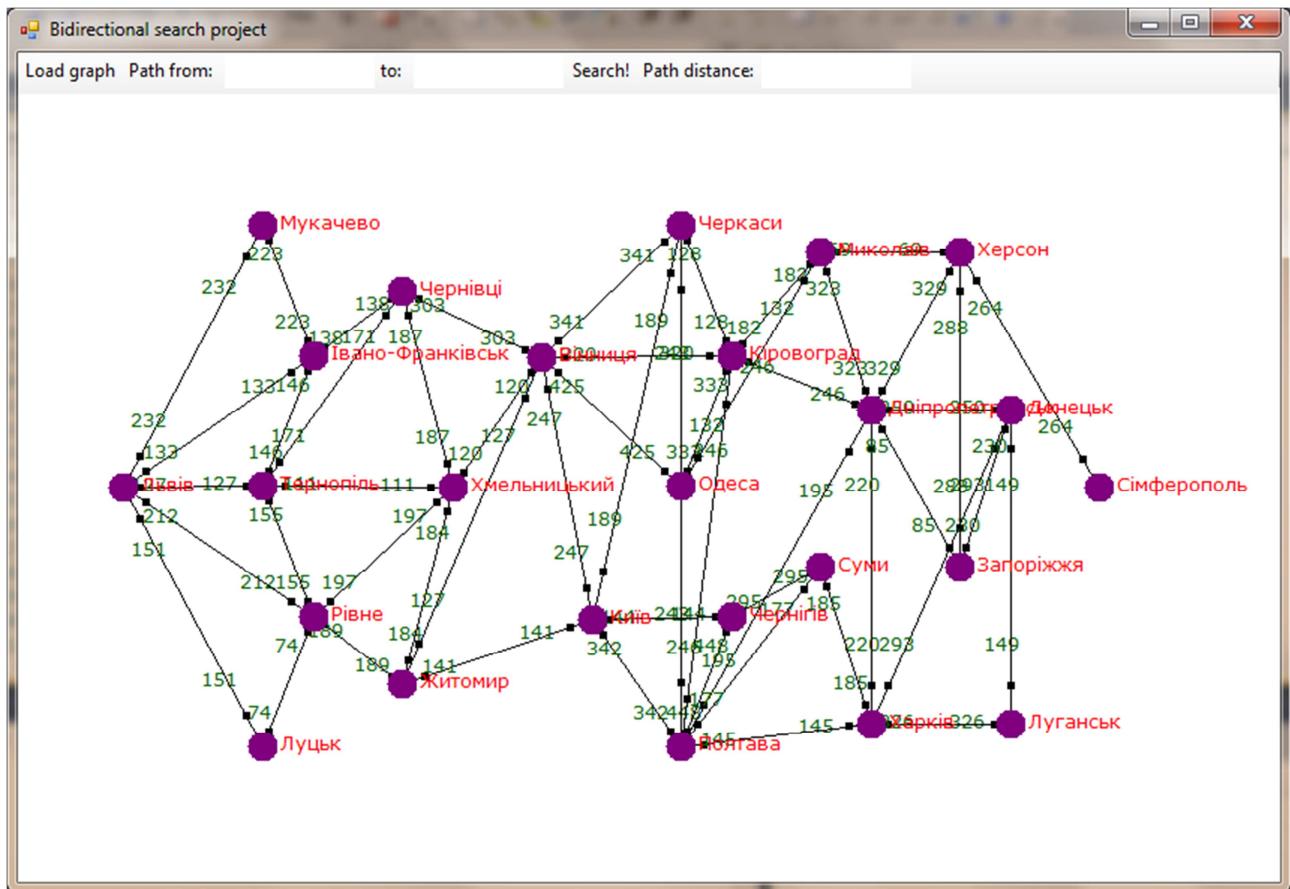
Таким чином, за 18 кроків алгоритм знайшов оптимальний шлях з мінімальним використанням пам'яті.

2. Реалізація

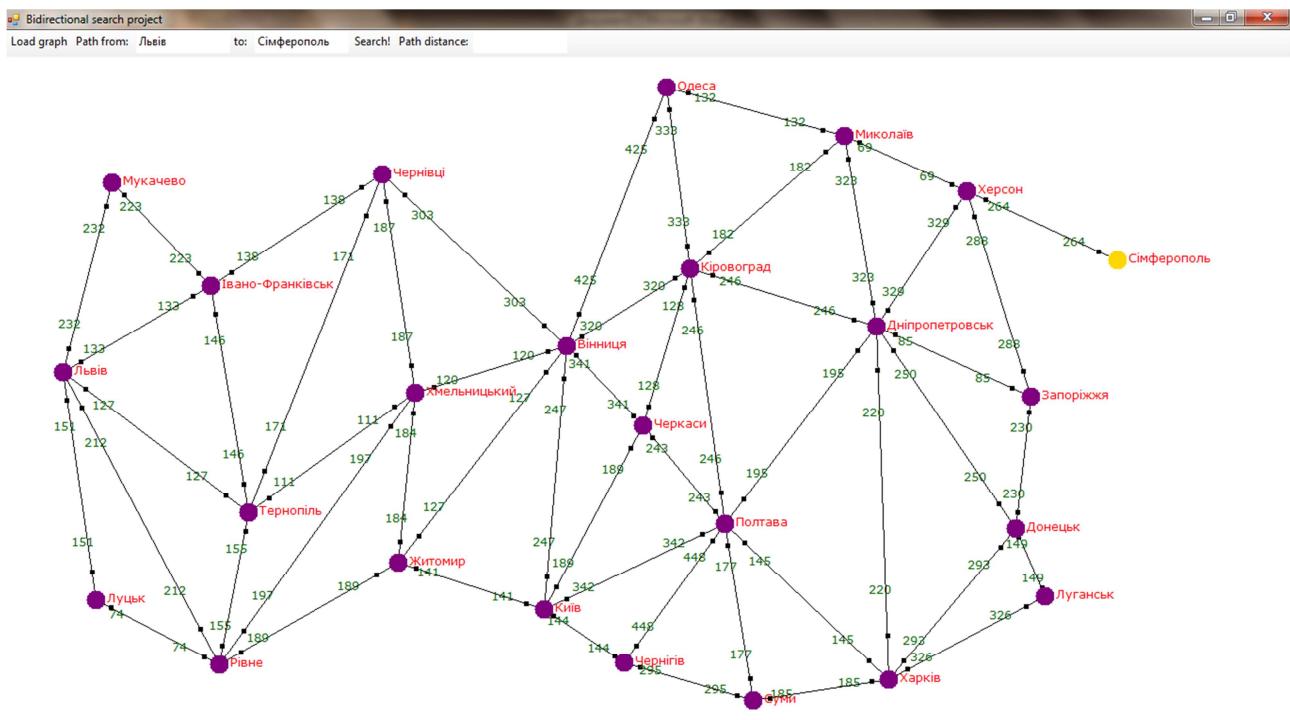
Програма дозволяє відкривання файла із заданим графом, завантаження його у пам'ять, візуалізацію та керування відображенням, динамічний вибір початкової та кінцевої вершин, та, звісно ж, bidirectional search.



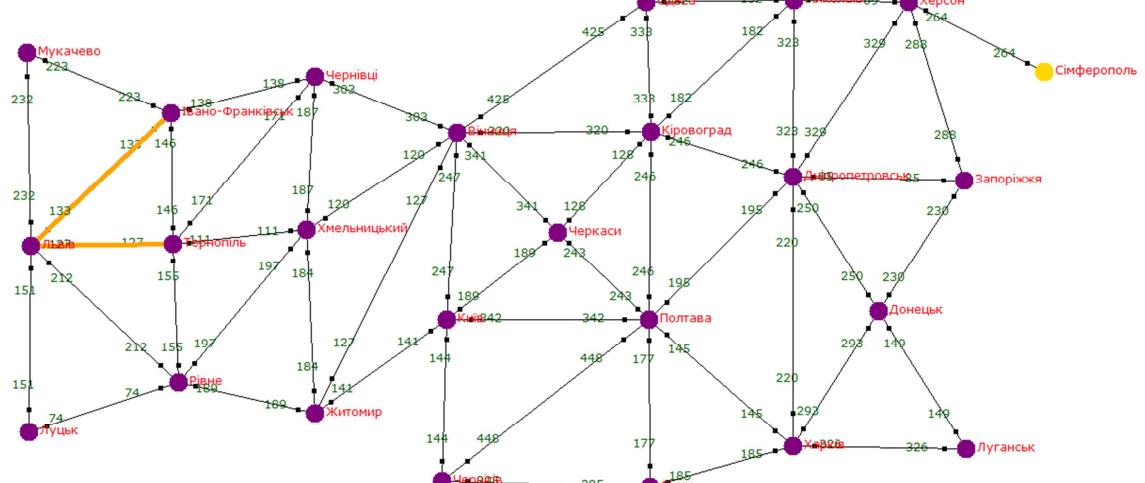
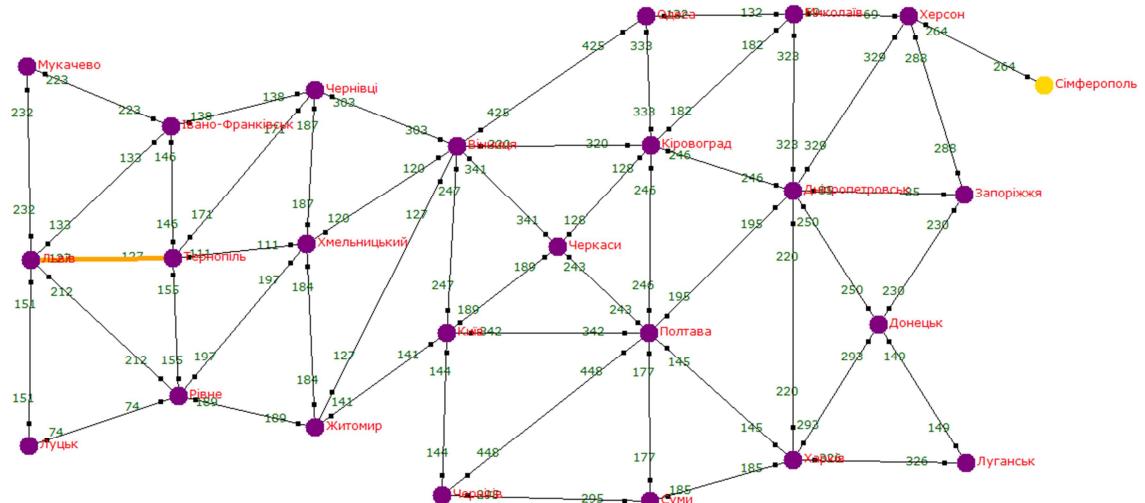
Структурно програма побудована за моделлю MVC. За зберігання та доступ до даних графа відповідає клас `WeightedGraph`, за візуалізацію та інтеракцію з користувачем – `MainForm`, за власне алгоритм пошуку – `SearchAgent`.



Після завантаження даних з файла view автоматично розставляє вершини порівнево, вписуючи в область вікна. При цьому можна переставити вершини у зручному порядку для покращення видимості. Відображення графа масштабується при будь-якій зміні розмірів вікна.

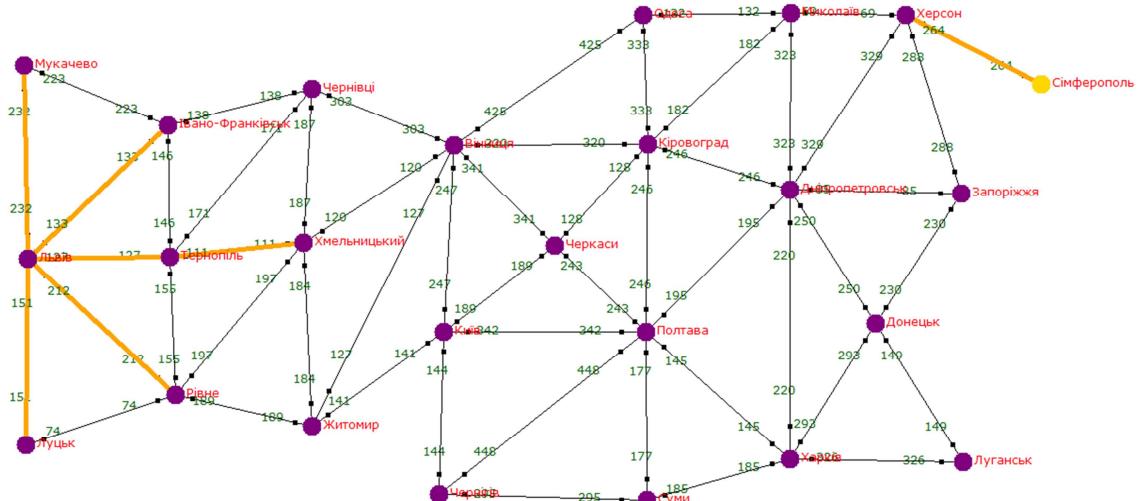


Встановивши початкову та кінцеву вершини шуканого шляху, можемо запустити пошук. Довжина шляху відображається сразу у верхній частині вікна, а процес побудови шляху «розгортається» покроково з інтервалом в одну секунду.

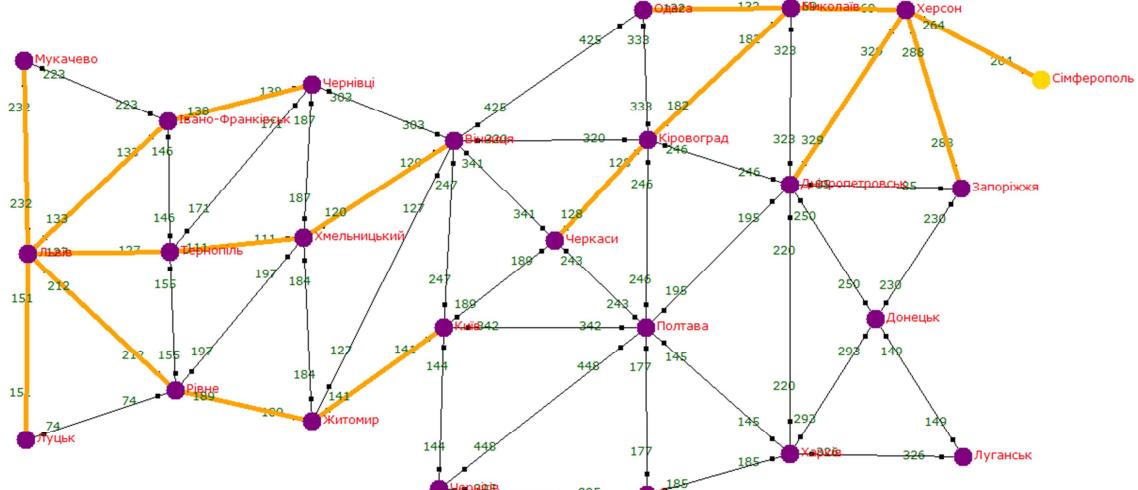


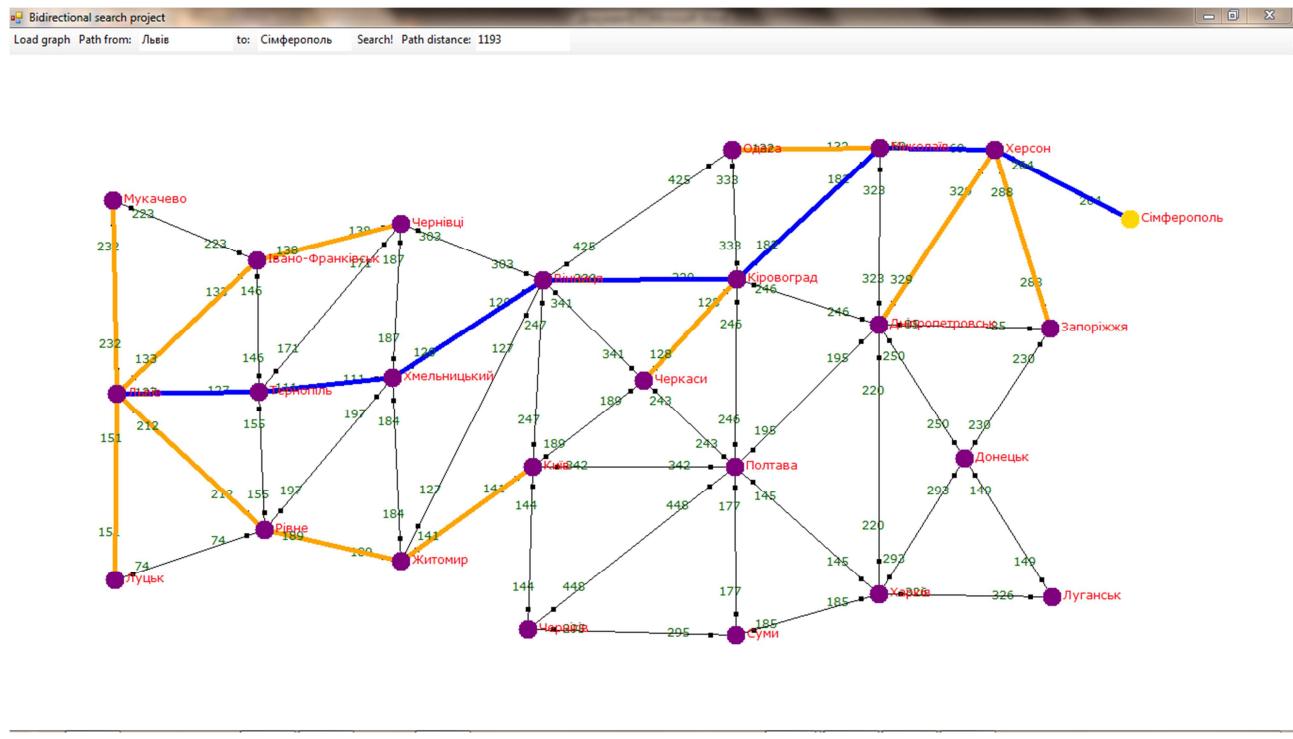
...

Bidirectional search project
Load graph Path from: Львів to: Сімферополь Search! Path distance: 1193



...
Bidirectional search project
Load graph Path from: Львів to: Сімферополь Search! Path distance: 1193





В результаті отримали найкоротший маршрут від Львова до Сімферополя через обласні центри України. Отримана магістральна довжина займає не мало, не багато – 1193 км. Перевіривши маршрут на Google maps, можемо підтвердити його валідність.

3. Висновки:

- bidirectional search у більшості випадків – найефективніший алгоритм неінформованого пошуку у зважених графах.
- Програмування алгоритму потребує більше зусиль.
- Алгоритм має найбільшу швидкість, тому може використовуватись у real-time системах для визначення маршрутів (див. всеможливі мережі).

4. Використана література:

- Russell S.J., Norvig P. - Artificial Intelligence - A Modern Approach (Prentice Hall, 3rd edition) – 2009
- Люгер Дж. Ф. Искусственный интеллект - 4изд – 2003
- http://en.wikipedia.org/wiki/Bidirectional_search