Национальный Исследовательский Университет ИТМО
Кафедра ВТ

Лабораторная работа №2

Системы искусственного интеллекта

Алгоритмы поиска

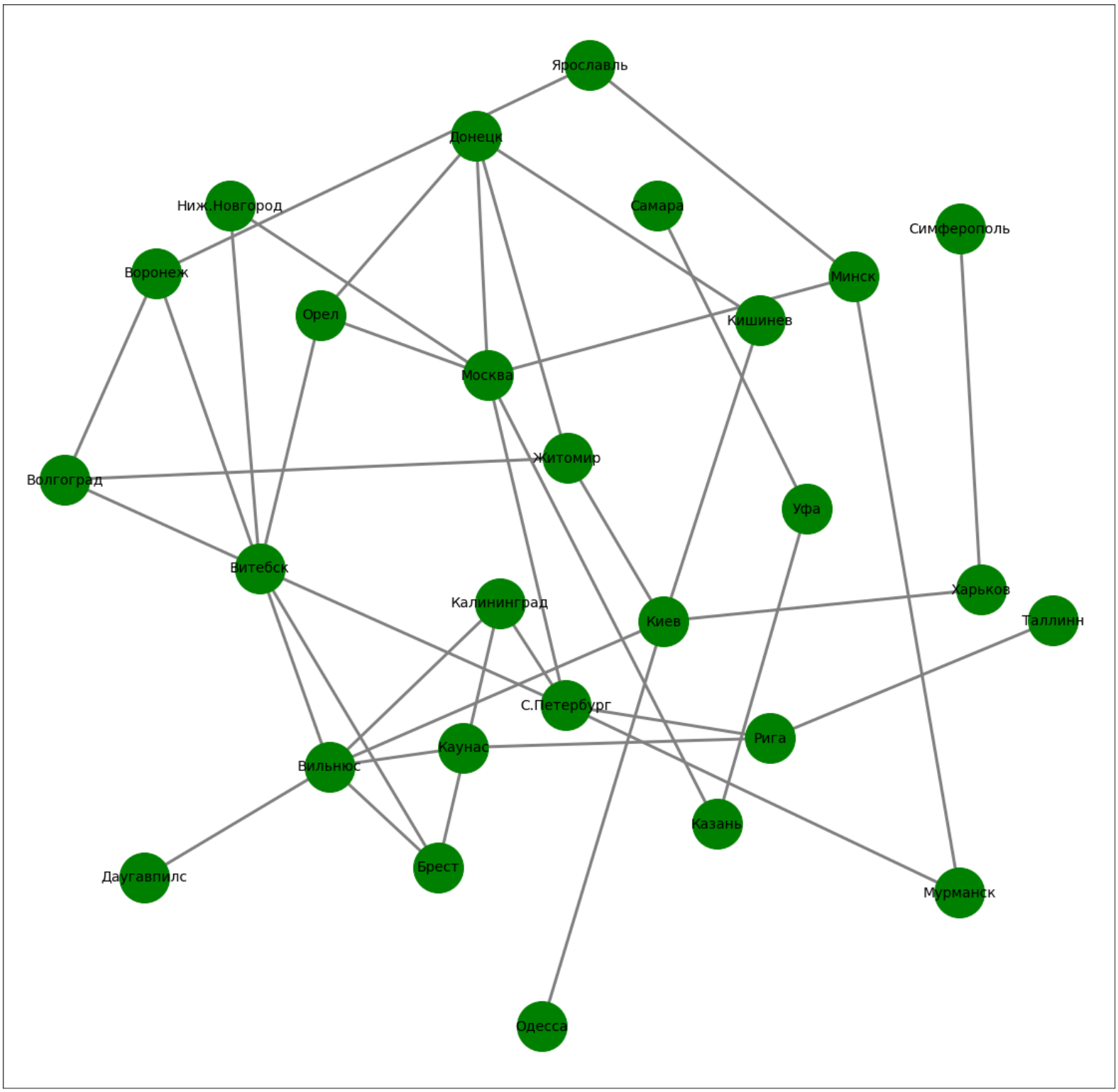
Выполнил: Федоров Сергей
Группа: P33113
Преподаватель: Болдырева Елена Александровна

Санкт-Петербург
2020 г.

Задание: По заданным вариантом города найти путь из одного в другой, путем применения различных алгоритмов информированного и неинформированного поиска.

Вариант: (19 + 02) % 10 +1 = #2 = Санкт-Петербург -> Житомир

Данные:



Выполнение:

Несмотря на то что было разрешено выполнение “голыми руками”, я предпочел так же реализовать данные алгоритмы поиска в программном виде.

Дабы не тратить место в отчете и не изводить бумагу, прилагаю ссылку и QR-code на исходный код лабораторной работы: https://github.com/Punctuality/Artificial_Intelligence_Systems_ITMO_2020/blob/master/lab2/Lab2.html

Далее перечислены алгоритмы поиска и схематический вывод программы при подставноке значений заданых вариантом:

Неинформированные алгоритмы поиска

- Breadth-First Search

```
bfs(adjancency_list, 'С.Петербург', 'Житомир')
```

```
 1 | С.Петербург -> Витебск Калининград Рига Москва Мурманск
       2 | Витебск -> Брест Вильнюс Воронеж Волгоград Ниж.Новгород
С.Петербург Орел
       2 | Калининград -> Брест Вильнюс С.Петербург
       2 | Рига -> С.Петербург Каунас Таллинн
       2 | Москва -> Казань Ниж.Новгород Минск Донецк С.Петербург Орел
       2 | Мурманск -> С.Петербург Минск
             3 | Брест -> Вильнюс Витебск Калининград
             3 | Вильнюс -> Брест Витебск Даугавпилс Калининград Каунас Киев
             3 | Воронеж -> Витебск Волгоград Ярославль
             3 | Волгоград -> Воронеж Витебск Житомир !!!
```

- Depth-First Search

```
dfs(adjancency_list, 'С.Петербург', 'Житомир')
```

```
 1 | С.Петербург -> Витебск
   2 | Витебск -> Брест
     3 | Брест -> Вильнюс
       4 | Вильнюс -> Даугавпилс
       4 | Вильнюс -> Киев
         5 | Киев -> Харьков
           6 | Харьков -> Симферополь
         5 | Киев -> Житомир !!!
```

- Depth-Limited Search

```
limited_dfs(adjancency_list, 'С.Петербург', 'Житомир', limit = 5)
```

```
 1 | С.Петербург -> Витебск
   2 | Витебск -> Брест
     3 | Брест -> Вильнюс
       4 | Вильнюс -> Даугавпилс
       4 | Вильнюс -> Киев
         5 | Киев -> Харьков
         5 | Киев -> Житомир !!!
```

- Iterative-Deepening Depth-First Search

```
iterative_dfs(adjancency_list, 'С.Петербург', ‘Житомир')
```

```
Limit = 1
 1 | С.Петербург -> Витебск
 1 | С.Петербург -> Мурманск
 1 | С.Петербург -> Калининград
 1 | С.Петербург -> Рига
 1 | С.Петербург -> Москва
Limit = 2
 1 | С.Петербург -> Витебск
   2 | Витебск -> Брест
   2 | Витебск -> Ниж.Новгород
   2 | Витебск -> Вильнюс
   2 | Витебск -> Воронеж
   2 | Витебск -> Орел
   2 | Витебск -> Волгоград
 1 | С.Петербург -> Мурманск
   2 | Мурманск -> Минск
 1 | С.Петербург -> Калининград
 1 | С.Петербург -> Рига
   2 | Рига -> Каунас
   2 | Рига -> Таллинн
 1 | С.Петербург -> Москва
   2 | Москва -> Донецк
   2 | Москва -> Казань
```

```
Limit = 3
 1 | С.Петербург -> Витебск
   2 | Витебск -> Брест
     3 | Брест -> Вильнюс
     3 | Брест -> Калининград
   2 | Витебск -> Ниж.Новгород
     3 | Ниж.Новгород -> Москва
   2 | Витебск -> Вильнюс
     3 | Вильнюс -> Даугавпилс
     3 | Вильнюс -> Киев
     3 | Вильнюс -> Каунас
   2 | Витебск -> Воронеж
     3 | Воронеж -> Ярославль
     3 | Воронеж -> Волгоград
   2 | Витебск -> Орел
     3 | Орел -> Донецк
   2 | Витебск -> Волгоград
     3 | Волгоград -> Житомир !!!
```

• Bidirectional Search

```
bds(adjancency_list, 'С.Петербург', 'Житомир')
```

```
1 Forward | С.Петербург -> Витебск Калининград Рига Москва Мурманск
1 Back    | Житомир -> Киев Донецк Волгоград

       2 Forward | Витебск -> Брест Вильнюс Воронеж Волгоград Ниж.Новгород
С.Петербург Орел
       2 Back    | Киев -> Вильнюс Житомир Кишинев Одесса Харьков

Вильнюс OR Волгоград !!!
```

Информированные алгоритмы поиска

• Best-First Search

```
best_fs(adjancency_list, 'Москва', 'Житомир', zhitomir_data)
```

```
1     | Москва -> Казань:1496 Ниж.Новгород:1218 Минск:411 Донецк:709
С.Петербург:1081 Орел:592
2     | Минск -> Москва:854 Мурманск:2094 Ярославль:1097
3     | Ярославль -> Воронеж:754 Минск:411
4     | Воронеж -> Витебск:553 Волгоград:1155 Ярославль:1097
5     | Витебск -> Брест:395 Вильнюс:543 Воронеж:754 Волгоград:1155
Ниж.Новгород:1218 С.Петербург:1081 Орел:592
6     | Брест -> Вильнюс:543 Витебск:553 Калининград:740
7     | Вильнюс -> Брест:395 Витебск:553 Даугавпилс:641 Калининград:740
Каунас:607 Киев:131
8     | Киев -> Вильнюс:543 Житомир:0 Кишинев:362 Одесса:446 Харьков:542
!!!
```

- A* Search

```
astar_s(adjancency_list, 'С.Петербург', 'Житомир', zhitomir_data)
```

```
1 | С.Петербург -> Витебск:1155 Калининград:1479 Рига:1442 Москва:1518
Мурманск:3506
      2 | Витебск -> Брест:1635 Вильнюс:1505 Воронеж:2225 Волгоград:3212
Ниж.Новгород:2731 Орел:1716
      3 | Рига -> Каунас:1515 Таллинн:1999
      4 | Калининград -> Брест:1833 Вильнюс:1615
            5 | Вильнюс -> Брест:1888 Даугавпилс:1814 Каунас:1671 Киев:1827
            6 | Каунас ->
      7 | Москва -> Казань:2975 Ниж.Новгород:2293 Минск:1765 Донецк:2457
Орел:1624
            8 | Орел -> Донецк:2450
            9 | Брест ->
            10 | Орел -> Донецк:2542
            11 | Минск -> Мурманск:5686 Ярославль:3391
                  12 | Даугавпилс ->
                  13 | Киев -> Житомир:1827 !!! Кишинев:2525 Одесса:2629
Харьков:2709
```

Вывод:

Помимо стандартных “сделал, понял” после лабораторной работы осталось несколько мыслей/выводов:

1. Неинформированные алгоритмы поиска не включают во внимание расстояние до цели, а лишь только находят первый попавшийся путь.
2. Из-за этого, в зависимости от состояния исходных данных, может радикально поменяться кол-во пройденных вершин, например такое возможно с поиском в глубину и измененным порядком в списке смежности.
3. Поиск с итеративным углублением выглядит как очень глупый brute-force по лимиту глубины в limited_dfs, хотя возможно если как-нибудь оптимизировать алгоритм сохраненными состояниями с прошлых итераций, то это даст прирост в производительности.
4. Информированные поиски в условиях данной задачи производят сильно больше действий чем другие, однако выглядит так что в реальном применении, где кол-во городов будет выходить на тысячи и десятки тысяч эти алгоритмы будут работать эффективнее. Собственно это можно вывести из асимптотической сложности алгоритма.