report

March 19, 2025

1 Laboratorium 1 - A* vs Dijkstra / Tabu search

1.0.1 Lukasz Fabia 272724

1.0.2 Jak zacząć?

Python3, najlepiej wersja > 3.11.

```
python -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
```

1.1 Uwaga

Ze względu na to, że kompliowałem `ipynb` do `pdf` to wpyłnąć na wyświetlanie niektórych danych w szczególności tabel.

1.2 Teoria i cel

Celem tego zdania jest znalezienie najkrótszego (min. droga) przejazdu MPK lub przejechanie z punktu A do B w najkrótszym czasie (min. czasu).

Do tego problemu optymalizacji należy wykorzystać dwa popularne algorytmy do wyszukiwania najktrótszych ścieżek - A* i Dijkstra.

1.2.1 Dijkstra

Algorytm zachłanny, znajdujący najkrótszą ścieżkę od węzła startowego do wszystkich innych węzłów w grafie. Wykorzstuje wagi (u mnie i w zadaniu korzysta on z kosztu czasu).

1.2.2 A*

Jest sprowadzalny do **Dijkstry**. Można powiedzieć, że jest taką ulepszoną wersją i jest częściej używany ze względu na optymalność. A* korzysta z heurystyk, żeby oszacować, czy opłaca się wybierać taka a nie inną ścieżkę. Warto dodać, że działa to gdy wiemy czego szukamy, czyli znany jest *target*.

1.3 Wgląd do danych

Pierwsze 5 wierszy. Z racji tego, że jednym z czynników, którym będziemy się kierować przy wybieraniu trasy będzie czas no to warto różnicę, która będzie kosztem danej ścieżki.

[1]:
```
from parser import get_df
df = get_df()

df.head()
```

[1]:

	company	line	departure_time	arrival_time	start_stop	end_stop	start_stop_lat	start_stop_lon	end_stop_lat	end_stop_lon
0	MPK Autobusy	A	20:52:00	20:53:00	Zajezdnia Obornicka	Paprotna	51.148737	17.021069	51.147752	17.020539
1	MPK Autobusy	A	20:53:00	20:54:00	Paprotna	Obornicka (Wołowska)	51.147752	17.020539	51.144385	17.023735
2	MPK Autobusy	A	20:54:00	20:55:00	Obornicka (Wołowska)	Bezpieczna	51.144385	17.023735	51.141360	17.026376
3	MPK Autobusy	A	20:55:00	20:57:00	Bezpieczna	Bałtycka	51.141360	17.026376	51.136632	17.030617
4	MPK Autobusy	A	20:57:00	20:59:00	Bałtycka	Broniewskiego	51.136632	17.030617	51.135851	17.037383

1.4 Jak zbudowałem strukturę grafu?

Generalnie w `/models/graph.py` mamy całą strukturę składa się ona z `Node`, `Edge` i coś co agreguje w sobie węzły, czyli graf. W klasie grafu znajduje się tylko słownik [ulica: węzeł].

Kolejnym krokiem było parsowanie danych i wrzucenie ich do grafu. Iterowałem po wierszach i:

1. Wyciągałem dane z wiersza i robiłem z tego krawędź.
2. Dodawałem startowy i końcowy przystanek do słownika. Dodawanie działa w taki sposób, że jeśli ulicy nie ma w kluczach to dodaje dane tej ulicy pod tym kluczem.
3. Aktualizacja krawędzi, czyli do listy startowego węzła dodaje nowy edge z podpiętymi węzłami.

W ten sposób, wystarczy utworzyć obiekt i otrzymujemy `Graf skierowany ważony`. Gdzie **waga** to czas przejazdu w minutach z A do B, a **skierowany** dlatego, że dodaje nową krawędź do startowego node'a. Więcej na temat struktury w kodzie źródłowym `models/graph.py`.

Ostatnia uwaga, przyjąłem, że przy szukaniu połączenia (krawędzi) mogę czekać maksymalnie **15 minut** (czyli od teraz + 15 minut), aby ograniczyć liczbę bezsensownych połączeń. Przykład:

jestem na Grunwaldzie, a algorytm proponuje mi połączenie, które odjeżdża za godzinę, ale podróż trwa tylko **12 minut do celu**. Taka optymalizacja.

1.5 Strategie działania

Ten problem można fajnie rozwiązać za pomoca strategii. Logika leży tylko w wybieraniu najbardziej **optymalnej** ścieżki. Zatem zbudowałem sobie szkielet z częścią wspólną tych algorytmów i będę manipulować tylko w konkretnych implementacjach wyborem ścieżki. W ten sposób będzie można tworzyć łatwiej nowe odmiany tych algorytmów.

Odpowiedź na konieć to jest czas w formacie `HH:MM:SS`, żeby było czytelniej, *liczba odwiedzonych wierzchołków* i *czas wykonania*(w ms).

1.5.1 Zadanie 1a

Algorytm wyszukiwania najkrótszej ścieżki z A do B za pomocą algorytmu Dijkstry w oparciu o kryterium czasu.

1.5.2 Inicjalizacja grafu

[2]:
```
from parser import to_graph
from dijkstra import Dijkstra
from a import AStarMinTime, AStarMinTransfers, AStarModified
from datetime import time


g = to_graph()
```

[3]:
```
# Times
t1 = time(hour=8, minute=18, second=0)
t2 = time(hour=20, minute=50, second=0)
t3 = time(hour=7, minute=20, second=0)
```

[4]:
```
d_engine = Dijkstra(g)

_ = d_engine.search("Muchobór Wielki", "Mroźna", t1)
_ = d_engine.search("Zajezdnia Obornicka", "Bałtycka", t2)
_ = d_engine.search("Wyszyńskiego", "PL. GRUNWALDZKI", t3)
```

```
It took: 758.51 ms

======================================================================
Results for Dijkstra:
======================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
----------------------------------------------------------------------
119   08:21:00    Muchobór Wie   08:22:00    Stanisławows    4
119   08:22:00    Stanisławows   08:23:00    Trawowa         5
119   08:23:00    Trawowa        08:24:00    Krzemienieck    6
```

119	08:24:00	Krzemienieck	08:25:00	Końcowa	7
119	08:25:00	Końcowa	08:27:00	Ostrowskiego	9
119	08:27:00	Ostrowskiego	08:28:00	FAT	10
5	08:28:00	FAT	08:29:00	Hutmen	11
5	08:29:00	Hutmen	08:30:00	Bzowa (Centr	12
5	08:30:00	Bzowa (Centr	08:31:00	pl. Srebrny	13
5	08:31:00	pl. Srebrny	08:32:00	Stalowa	14
5	08:32:00	Stalowa	08:34:00	Pereca	16
5	08:34:00	Pereca	08:35:00	Grabiszyńska	17
5	08:35:00	Grabiszyńska	08:36:00	Kolejowa	18
5	08:36:00	Kolejowa	08:38:00	pl. Legionów	20
5	08:38:00	pl. Legionów	08:40:00	Arkady (Capi	22
5	08:40:00	Arkady (Capi	08:42:00	DWORZEC GŁÓW	24
K	08:42:00	DWORZEC GŁÓW	08:47:00	GALERIA DOMI	29
12	08:48:00	GALERIA DOMI	08:51:00	Urząd Wojewó	33
149	08:51:00	Urząd Wojewó	08:52:00	most Grunwal	34
149	08:52:00	most Grunwal	08:54:00	PL. GRUNWALD	36
111	08:55:00	PL. GRUNWALD	08:57:00	Bujwida	39
111	08:57:00	Bujwida	08:59:00	Kochanowskie	41
111	08:59:00	Kochanowskie	09:00:00	Śniadeckich	42
111	09:00:00	Śniadeckich	09:01:00	Zacisze	43
111	09:01:00	Zacisze	09:03:00	Kwidzyńska	45
111	09:03:00	Kwidzyńska	09:05:00	Brücknera	47
111	09:05:00	Brücknera	09:07:00	C.H. Korona	49
111	09:07:00	C.H. Korona	09:09:00	Zielna	51
111	09:09:00	Zielna	09:10:00	Psie Pole	52
111	09:10:00	Psie Pole	09:11:00	Psie Pole (R	53
131	09:13:00	Psie Pole (R	09:15:00	KIEŁCZOWSKA	57
131	09:15:00	KIEŁCZOWSKA	09:16:00	Poleska	58
131	09:16:00	Poleska	09:17:00	Kiełczowska	59
131	09:17:00	Kiełczowska	09:18:00	Żmudzka	60
121	09:19:00	Żmudzka	09:20:00	Kiełczowska	62
931	09:24:00	Kiełczowska	09:25:00	Kiełczów - S	67
931	09:25:00	Kiełczów - S	09:26:00	Kiełczów - B	68
931	09:26:00	Kiełczów - B	09:27:00	Kiełczów - W	69
931	09:27:00	Kiełczów - W	09:28:00	Kiełczów - p	70
931	09:28:00	Kiełczów - p	09:30:00	Kiełczów - W	72
911	09:36:00	Kiełczów - W	09:37:00	Kiełczów - o	79
911	09:37:00	Kiełczów - o	09:38:00	Wilczyce - S	80
911	09:38:00	Wilczyce - S	09:39:00	Wilczyce - D	81
911	09:39:00	Wilczyce - D	09:40:00	Wilczyce - D	82
911	09:40:00	Wilczyce - D	09:41:00	Wilczyce - B	83
911	09:41:00	Wilczyce - B	09:42:00	Wilczyce - W	84
911	09:42:00	Wilczyce - W	09:42:00	Wilczyce	84
911	09:42:00	Wilczyce	09:43:00	Mroźna	85

Total time (Dijkstra): 01:25:00

```
Total visited nodes: 35988
========================================================================

It took: 4.32 ms

========================================================================
Results for Dijkstra:
========================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
------------------------------------------------------------------------
A     20:52:00    Zajezdnia Ob   20:53:00    Paprotna        3
A     20:53:00    Paprotna       20:54:00    Obornicka (W    4
A     20:54:00    Obornicka (W   20:55:00    Bezpieczna      5
A     20:55:00    Bezpieczna     20:57:00    Bałtycka        7
------------------------------------------------------------------------

Total time (Dijkstra): 00:07:00
Total visited nodes: 214
========================================================================

It took: 17.40 ms

========================================================================
Results for Dijkstra:
========================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
------------------------------------------------------------------------
1     07:22:00    Wyszyńskiego   07:23:00    Prusa           3
1     07:23:00    Prusa          07:25:00    Piastowska      5
1     07:25:00    Piastowska     07:28:00    PL. GRUNWALD    8
------------------------------------------------------------------------

Total time (Dijkstra): 00:08:00
Total visited nodes: 870
========================================================================
```

Wnioski

1. Minimalny czas podróży
    - gwarantuje nam, że koszt faktycznie będzie najmniejszy w moim przypadku - czas podróży, ale gdy warunki będą idealne tj. brak opóźnień tramwajów.
    - można pomyśleć nad algorytmem, który obsłuży sytuacje losowe.
2. Liczba odwiedzonych wierzchołków
    - jest ona całkiem spora, ponieważ przeszukuje wszystko.

3. Performance
    - długi czas wykonania się dla tras gdzie mamy kilka przystanków to kilkanaście ms, ale liczba rośnie w przypadku trudniejszych odcinków. Jest to spowodowane przeszukiwaniem wszerz co też nijako wiąże się z dużą ilością odwiedzonych wierzchołków.
    - sam język nie jest demonem prędkości, można przepisać na coś szybszego jak (np. Rust, C/C++).
4. Optymalizacje
    - użycie algorytmu z heurystyką np. A*

1.5.3 Zadanie 1b

Algorytm wyszukiwania najkrótszej ścieżki z A do B za pomocą algorytmu A* w oparciu o kryterium czasu.

Jako, że A* to jest modyfikacja Dijkstry to po prostu wartość `priority` to będzie strategia obliczania kosztu dla mojej klasy w tym wypadku klasycznego A*. Poniżej snipped kodu z `a.py` z liczeniem priorytetu.

```
priority = self.cost_strategy(
                new_cost=new_cost,
                end_node=end_node,
                next_end_node=next_edge.end_node,
            )
```

A tutaj heurystyka, zastosowana w szukaniu obiecującej ścieżki. Akturat do policzenia odległość użyłem biblioteki `geopy` ze względu na współrzędne geograficzne. Wszystkie heurystyki znajdują się w `search.py`.

```
def cost_strategy(self, new_cost, **kwargs):
    return new_cost + self._geo_heuristic(
        a=kwargs["end_node"], b=kwargs["next_end_node"]
    )
```

[5]:
```
a_engine = AStarMinTime(g)
_ = a_engine.search("Muchobór Wielki", "Mroźna", t1)
_ = a_engine.search("Zajezdnia Obornicka", "Bałtycka", t2)
_ = a_engine.search("Wyszyńskiego", "PL. GRUNWALDZKI", t3)
```

```
It took: 59.86 ms

======================================================================
Results for AStarMinTime:
======================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
----------------------------------------------------------------------
119   08:21:00    Muchobór Wie   08:22:00    Stanisławows    4
119   08:22:00    Stanisławows   08:23:00    Trawowa         5
119   08:23:00    Trawowa        08:24:00    Krzemienieck    6
```

119	08:24:00	Krzemienieck	08:25:00	Końcowa	7
119	08:25:00	Końcowa	08:27:00	Ostrowskiego	9
119	08:27:00	Ostrowskiego	08:28:00	FAT	10
5	08:28:00	FAT	08:29:00	Hutmen	11
5	08:29:00	Hutmen	08:30:00	Bzowa (Centr	12
5	08:30:00	Bzowa (Centr	08:31:00	pl. Srebrny	13
5	08:31:00	pl. Srebrny	08:32:00	Stalowa	14
5	08:32:00	Stalowa	08:34:00	Pereca	16
5	08:34:00	Pereca	08:35:00	Grabiszyńska	17
5	08:35:00	Grabiszyńska	08:36:00	Kolejowa	18
5	08:36:00	Kolejowa	08:38:00	pl. Legionów	20
11	08:49:00	pl. Legionów	08:51:00	Narodowe For	33
6	08:53:00	Narodowe For	08:56:00	Renoma	38
148	08:59:00	Renoma	09:02:00	Dworzec Głów	44
114	09:05:00	Dworzec Głów	09:06:00	Kościuszki	48
114	09:06:00	Kościuszki	09:07:00	Świstackiego	49
114	09:07:00	Świstackiego	09:09:00	Na Niskich Ł	51
3	09:14:00	Na Niskich Ł	09:16:00	pl. Zgody (M	58
3	09:16:00	pl. Zgody (M	09:19:00	pl. Wróblews	61
4	09:21:00	pl. Wróblews	09:23:00	Urząd Wojewó	65
4	09:23:00	Urząd Wojewó	09:25:00	most Grunwal	67
4	09:25:00	most Grunwal	09:27:00	PL. GRUNWALD	69
116	09:28:00	PL. GRUNWALD	09:30:00	Kochanowskie	72
13	09:30:00	Kochanowskie	09:31:00	Chopina	73
13	09:31:00	Chopina	09:32:00	Karłowicza	74
13	09:32:00	Karłowicza	09:33:00	Stadion Olim	75
9	09:34:00	Stadion Olim	09:35:00	8 Maja	77
9	09:35:00	8 Maja	09:36:00	Godebskiego	78
9	09:36:00	Godebskiego	09:38:00	SĘPOLNO	80
115	09:38:00	SĘPOLNO	09:40:00	Monopolowa	82
115	09:40:00	Monopolowa	09:41:00	Kolumba	83
115	09:41:00	Kolumba	09:42:00	Magellana	84
115	09:42:00	Magellana	09:43:00	Swojczyce	85
118	09:43:00	Swojczyce	09:45:00	Miłoszycka	87
118	09:45:00	Miłoszycka	09:46:00	Gospodarska	88
118	09:46:00	Gospodarska	09:46:00	Ceglana	88
118	09:46:00	Ceglana	09:48:00	Kowalska	90
118	09:48:00	Kowalska	09:49:00	Tczewska	91
118	09:49:00	Tczewska	09:50:00	Działdowska	92
118	09:50:00	Działdowska	09:51:00	Kowale (Stac	93
118	09:51:00	Kowale (Stac	09:52:00	Kowalska 56	94
118	09:52:00	Kowalska 56	09:53:00	Olsztyńska	95
118	09:53:00	Olsztyńska	09:55:00	C.H. Korona	97
121	09:55:00	C.H. Korona	09:57:00	Zielna	99
121	09:57:00	Zielna	09:58:00	Psie Pole	100
121	09:58:00	Psie Pole	09:59:00	Psie Pole (R	101
N	10:05:00	Psie Pole (R	10:07:00	KIEŁCZOWSKA	109
N	10:07:00	KIEŁCZOWSKA	10:08:00	Poleska	110

```
150   10:21:00    Poleska         10:23:00    Szewczenki      125
150   10:23:00    Szewczenki      10:24:00    Gorlicka        126
911   10:30:00    Gorlicka        10:31:00    Palacha         133
911   10:31:00    Palacha         10:32:00    Zgorzelisko     134
911   10:32:00    Zgorzelisko     10:32:00    Kurlandzka      134
911   10:32:00    Kurlandzka      10:33:00    Mroźna          135
-----------------------------------------------------------------------

Total time (AStarMinTime): 02:15:00
Total visited nodes: 2481
=======================================================================

It took: 3.43 ms

=======================================================================
Results for AStarMinTime:
=======================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
-----------------------------------------------------------------------
A     20:52:00    Zajezdnia Ob   20:53:00    Paprotna        3
A     20:53:00    Paprotna       20:54:00    Obornicka (W    4
A     20:54:00    Obornicka (W   20:55:00    Bezpieczna      5
A     20:55:00    Bezpieczna     20:57:00    Bałtycka        7
-----------------------------------------------------------------------

Total time (AStarMinTime): 00:07:00
Total visited nodes: 96
=======================================================================

It took: 5.52 ms

=======================================================================
Results for AStarMinTime:
=======================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
-----------------------------------------------------------------------
N     07:24:00    Wyszyńskiego   07:26:00    Ogród Botani    6
9     07:27:00    Ogród Botani   07:28:00    Górnickiego     8
9     07:28:00    Górnickiego    07:29:00    Piastowska      9
16    07:31:00    Piastowska     07:35:00    PL. GRUNWALD    15
-----------------------------------------------------------------------

Total time (AStarMinTime): 00:15:00
Total visited nodes: 188
=======================================================================
```

Wnioski

1. Minimalny czas podróży
   - rozwiązanie jest gorsze jak można było się spodziewać, ale bardzo szybko je dostajemy
   - warto dodać, że jest to zależne od heurystyki
2. Liczba odwiedzonych wierzchołków
   - liczba jest o wiele mniejsza od rozwiązania problemu za pomocą **Dijkstry**
   - jest to spowodowane użyciem heurystyki, która przeszukuje rokujące węzły
   - całkiem mało odwiedzonych, ale wynika to, że szuka tylko tam gdzie jest obiecująco.
3. Performance
   - tu jest znaczenie lepiej bo nie przeszukujemy wszystkiego
4. Opytmalizacje
   - można dodać heurystyke, która by sprawdzała kierunek, którym ma się kierować na podstawie kątów obliczanych z koordynatów punktu A i B.

Zadanie 1c Minimalizacja liczby przesiadek. Intuicja podpowiada, że trzeba będzie wprowadzić coś w stylu kary, dodatkowego kosztu doliczanego do ścieżki, która może i jest najkrótszą, ale wymaga przesiadki właśnie. W ten sposób algorytm będzie brał pod uwagę ścieżki, które nie wymagają zmiany `line`. Tak naprawdę cała zabawa sprowadziła się do dodatania do nowego kosztu sprawdzienie ile kosztować przesiadka. Poniżej *snipped* z source kodu, który znajduje się `a.py`.

```
new_cost = (
            cost_so_far[current_node]
            + self.graph.compute_cost(next_edge, current_time)
            + self.graph.line_change_cost(
                edge=came_from[current_node.name], next_edge=next_edge
            )
        )
```

[6]:
```
a_mut_engine = AStarMinTransfers(g)

_ = a_mut_engine.search("Muchobór Wielki", "Mroźna", t1)
_ = a_mut_engine.search("Zajezdnia Obornicka", "Bałtycka", t2)
_ = a_mut_engine.search("Wyszyńskiego", "PL. GRUNWALDZKI", t3)
```

```
It took: 98.55 ms

======================================================================
Results for AStarMinTransfers:
======================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
----------------------------------------------------------------------
119   08:21:00    Muchobór Wie   08:22:00    Stanisławows    4
```

```
119   08:22:00    Stanisławows   08:23:00    Trawowa         5
119   08:23:00    Trawowa        08:24:00    Krzemienieck    6
119   08:24:00    Krzemienieck   08:25:00    Końcowa         7
119   08:25:00    Końcowa        08:27:00    Ostrowskiego    9
119   08:27:00    Ostrowskiego   08:28:00    FAT             10
134   08:37:00    FAT            08:38:00    Hutmen          520
134   08:38:00    Hutmen         08:40:00    Bzowa (Centr    522
134   08:40:00    Bzowa (Centr   08:41:00    pl. Srebrny     523
134   08:41:00    pl. Srebrny    08:43:00    Stalowa         525
134   08:43:00    Stalowa        08:44:00    Grochowa        526
134   08:44:00    Grochowa       08:45:00    Krucza          527
134   08:45:00    Krucza         08:48:00    Rondo           530
D     09:03:00    Rondo          09:09:00    Arkady (Capi    1051
D     09:17:00    Arkady (Capi   09:22:00    GALERIA DOMI    1064
D     09:22:00    GALERIA DOMI   09:25:00    Urząd Wojewó    1067
D     09:25:00    Urząd Wojewó   09:26:00    most Grunwal    1068
D     09:26:00    most Grunwal   09:28:00    PL. GRUNWALD    1070
911   09:43:00    PL. GRUNWALD   09:45:00    Kochanowskie    1587
911   09:45:00    Kochanowskie   09:47:00    Śniadeckich     1589
911   09:47:00    Śniadeckich    09:48:00    Zacisze         1590
911   09:48:00    Zacisze        09:50:00    Kwidzyńska      1592
911   09:50:00    Kwidzyńska     09:52:00    Brücknera       1594
N     10:06:00    Brücknera      10:10:00    Psie Pole       2112
N     10:10:00    Psie Pole      10:11:00    Psie Pole (R    2113
N     10:11:00    Psie Pole (R   10:13:00    KIEŁCZOWSKA     2115
N     10:13:00    KIEŁCZOWSKA    10:14:00    Poleska         2116
150   10:21:00    Poleska        10:23:00    Szewczenki      2625
150   10:23:00    Szewczenki     10:24:00    Gorlicka        2626
911   10:30:00    Gorlicka       10:31:00    Palacha         3133
911   10:31:00    Palacha        10:32:00    Zgorzelisko     3134
911   10:32:00    Zgorzelisko    10:32:00    Kurlandzka      3134
911   10:32:00    Kurlandzka     10:33:00    Mroźna          3135
-------------------------------------------------------------------

Total time (AStarMinTransfers): 02:15:00
Total visited nodes: 1926
===================================================================

It took: 3.66 ms

===================================================================
Results for AStarMinTransfers:
===================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
-------------------------------------------------------------------
A     20:52:00    Zajezdnia Ob   20:53:00    Paprotna        3
A     20:53:00    Paprotna       20:54:00    Obornicka (W    4
A     20:54:00    Obornicka (W   20:55:00    Bezpieczna      5
```

```
A     20:55:00    Bezpieczna     20:57:00    Bałtycka        7
----------------------------------------------------------------------

Total time (AStarMinTransfers): 00:07:00
Total visited nodes: 96
======================================================================

It took: 8.41 ms

======================================================================
Results for AStarMinTransfers:
======================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
----------------------------------------------------------------------
N     07:24:00    Wyszyńskiego   07:26:00    Ogród Botani    6
N     07:26:00    Ogród Botani   07:27:00    Katedra         7
111   07:28:00    Katedra        07:29:00    Reja            509
111   07:29:00    Reja           07:31:00    PL. GRUNWALD    511
----------------------------------------------------------------------

Total time (AStarMinTransfers): 00:11:00
Total visited nodes: 188
======================================================================
```

Wnioski Wynik jest podobny do zwykłego A*, gdzie minimalizowany był czas. Tutaj mieliśmy minimalizować przesiadki i powiedzmy, że się udało na przykładowych trasach. Mamy mniej zmian linii, czas nieznaczenie się wydłużył i widać doliczoną karę za przesiadkę. W niektórych testach liczba odwiedzonych wierzchołków jest mniejsza.

2 Podsumowanie pierwszej części listy

Generalnie zgodnie z założeniem Dijkstra zwraca najlepsze wyniki w nie najlepszym czasie. Udało się w miarę sensownie zaimplementować A* (minimalizacja czasu albo przystanków), który zwaraca dobrą odpowiedź w naprawdę fajnym czasie. Udało się napisać względnie bez duplikacji kodu przez co łatwo można pisać swoje implementacje, które jakoś optymalizują wybór ścieżki.

Największy problem jednak sprawił sam `Python`, jako osoba, która raczej jest przyzwyczajona do **Go**, **TypeScripta**, **Javy** no to było ciężko debugować, ale trzeba przyznać, że *JupyterNotebook* pomógł w procesie. Last but not least, ułatwieniem było, że doba nie trwała 24h tylko więcej to pomogło się skupiać na policzeniu kosztu w minutach między węzłami (przystankami).

2.1 Tabu Search

Metaheurystyka polegająca na interacyjnym przeszukiwaniu sąsiedztwa, uwzględnia zestaw ruchów niedozwolonych. Wymaga dodakowej pamięci w porównaniu do **przeszukiwania lokalnego**.

W zadaniu należy przechejać przez wszystkie przystanki i wrócić się do punktu początkowego po przez minimalizację:

- czasu
- przesiadek

Podobnie jak w poprzednim zadaniu tutaj też postarałem się aby nie duplikować kodu i poprostu parametry w zadaniu będę przyjmować opcjonalnie.

W implementacji `wycieczki` po mieście użyłem A*, ponieważ Dijkstra wykonywałby się znacznie dłużej.

2.2 Przykład użycia

[7]:
```
from tabu import Tabu
max_iter = 111
points = ["PL. GRUNWALDZKI", "Dubois", "DWORZEC GŁÓWNY"]
src = "Wyszyńskiego"
t = Tabu(g=g, t=t2, points=points, src=src, max_iter=max_iter)
```

[8]:
```
best_sln = t.search()
t.set_points(best_sln)
t.go_on_a_trip()
```

```
It took: 1683.33 ms
It took: 7.80 ms

======================================================================
Results for AStarMinTime:
======================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
----------------------------------------------------------------------
A     20:50:00    Wyszyńskiego   20:52:00    Ogród Botani    2
A     20:52:00    Ogród Botani   20:53:00    Katedra         3
A     20:53:00    Katedra        20:54:00    Urząd Wojewó    4
10    20:57:00    Urząd Wojewó   21:00:00    GALERIA DOMI    10
N     21:11:00    GALERIA DOMI   21:15:00    DWORZEC GŁÓW    25
----------------------------------------------------------------------

Total time (AStarMinTime): 00:25:00
Total visited nodes: 228
======================================================================

It took: 10.94 ms

======================================================================
Results for AStarMinTime:
======================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
```

```
------------------------------------------------------------------------
9     21:19:00    DWORZEC GŁÓW   21:21:00    Wzgórze Part    6
9     21:21:00    Wzgórze Part   21:24:00    GALERIA DOMI    9
D     21:29:00    GALERIA DOMI   21:31:00    Urząd Wojewó    16
D     21:31:00    Urząd Wojewó   21:32:00    most Grunwal    17
D     21:32:00    most Grunwal   21:34:00    PL. GRUNWALD    19
------------------------------------------------------------------------

Total time (AStarMinTime): 00:19:00
Total visited nodes: 334
========================================================================

It took: 11.95 ms

========================================================================
Results for AStarMinTime:
========================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
------------------------------------------------------------------------
13    21:36:00    PL. GRUNWALD   21:37:00    most Grunwal    3
146   21:45:00    most Grunwal   21:47:00    Poczta Główn    13
D     21:58:00    Poczta Główn   22:00:00    GALERIA DOMI    26
11    22:02:00    GALERIA DOMI   22:03:00    pl. Nowy Tar    29
11    22:03:00    pl. Nowy Tar   22:04:00    Hala Targowa    30
11    22:04:00    Hala Targowa   22:06:00    pl. Bema        32
6     22:11:00    pl. Bema       22:13:00    Dubois          39
------------------------------------------------------------------------

Total time (AStarMinTime): 00:39:00
Total visited nodes: 372
========================================================================

It took: 5.72 ms

========================================================================
Results for AStarMinTime:
========================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
------------------------------------------------------------------------
19    22:14:00    Dubois         22:16:00    pl. Bema        3
19    22:16:00    pl. Bema       22:18:00    Ogród Botani    5
N     22:18:00    Ogród Botani   22:20:00    Wyszyńskiego    7
------------------------------------------------------------------------

Total time (AStarMinTime): 00:07:00
Total visited nodes: 158
========================================================================
```

```
It took: 0.00 ms
```

2.2.1 Tabu Search dobór długości tablicy T (b)

Dostosowuje długość listy tabu w trakcie działania, w zależności od postępu algorytmu. Długość T to 10, ale można zmienić.

[9]:
```
best_sln = t.dynamic_search()
t.set_points(best_sln)
t.go_on_a_trip()
```

```
It took: 2678.78 ms
It took: 5.78 ms

======================================================================
Results for AStarMinTime:
======================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
----------------------------------------------------------------------
1     20:53:00    Wyszyńskiego   20:55:00    Nowowiejska     5
8     20:56:00    Nowowiejska    20:57:00    Jedności Nar    7
8     20:57:00    Jedności Nar   20:58:00    Na Szańcach     8
8     20:58:00    Na Szańcach    20:59:00    pl. Bema        9
19    21:03:00    pl. Bema       21:05:00    Dubois          15
----------------------------------------------------------------------

Total time (AStarMinTime): 00:15:00
Total visited nodes: 186
======================================================================

It took: 5.95 ms

======================================================================
Results for AStarMinTime:
======================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
----------------------------------------------------------------------
13    21:13:00    Dubois         21:15:00    pl. Bema        10
19    21:16:00    pl. Bema       21:18:00    Ogród Botani    13
N     21:18:00    Ogród Botani   21:20:00    Wyszyńskiego    15
----------------------------------------------------------------------

Total time (AStarMinTime): 00:15:00
Total visited nodes: 158
======================================================================

It took: 0.01 ms
It took: 8.22 ms
```

```
======================================================================
Results for AStarMinTime:
======================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
----------------------------------------------------------------------
A     21:20:00    Wyszyńskiego   21:22:00    Ogród Botani    2
A     21:22:00    Ogród Botani   21:23:00    Katedra         3
A     21:23:00    Katedra        21:24:00    Urząd Wojewó    4
2     21:28:00    Urząd Wojewó   21:31:00    GALERIA DOMI    11
N     21:41:00    GALERIA DOMI   21:45:00    DWORZEC GŁÓW    25
----------------------------------------------------------------------

Total time (AStarMinTime): 00:25:00
Total visited nodes: 222
======================================================================

It took: 12.08 ms

======================================================================
Results for AStarMinTime:
======================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
----------------------------------------------------------------------
23    21:45:00    DWORZEC GŁÓW   21:47:00    Wzgórze Part    2
23    21:47:00    Wzgórze Part   21:50:00    GALERIA DOMI    5
12    21:52:00    GALERIA DOMI   21:55:00    Urząd Wojewó    10
12    21:55:00    Urząd Wojewó   21:57:00    most Grunwal    12
12    21:57:00    most Grunwal   21:59:00    PL. GRUNWALD    14
----------------------------------------------------------------------

Total time (AStarMinTime): 00:14:00
Total visited nodes: 324
======================================================================

It took: 4.36 ms

======================================================================
Results for AStarMinTime:
======================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
----------------------------------------------------------------------
16    21:59:00    PL. GRUNWALD   22:01:00    Piastowska      2
16    22:01:00    Piastowska     22:03:00    Prusa           4
16    22:03:00    Prusa          22:04:00    Wyszyńskiego    5
----------------------------------------------------------------------

Total time (AStarMinTime): 00:05:00
```

```
Total visited nodes: 128
======================================================================
```

2.2.2 Tabu Search aspiracja (c)

Pozwala zaakceptować rozwiązania z listy tabu, jeśli są lepsze niż poprzednie najlepsze rozwiązanie.

[10]:
```
best_sln = t.aspiration_search()
t.set_points(best_sln)
t.go_on_a_trip()
```

```
It took: 5103.68 ms
It took: 0.01 ms
It took: 0.00 ms
It took: 0.00 ms
It took: 5.95 ms

======================================================================
Results for AStarMinTime:
======================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
----------------------------------------------------------------------
A     20:50:00    Wyszyńskiego   20:52:00    Ogród Botani    2
19    20:58:00    Ogród Botani   20:59:00    Górnickiego     9
19    20:59:00    Górnickiego    21:00:00    Piastowska      10
19    21:00:00    Piastowska     21:04:00    PL. GRUNWALD    14
----------------------------------------------------------------------

Total time (AStarMinTime): 00:14:00
Total visited nodes: 112
======================================================================

It took: 6.16 ms

======================================================================
Results for AStarMinTime:
======================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
----------------------------------------------------------------------
12    21:05:00    PL. GRUNWALD   21:06:00    most Grunwal    2
146   21:15:00    most Grunwal   21:17:00    Poczta Główn    13
146   21:17:00    Poczta Główn   21:19:00    skwer Krasiń    15
146   21:19:00    skwer Krasiń   21:21:00    DWORZEC GŁÓW    17
----------------------------------------------------------------------

Total time (AStarMinTime): 00:17:00
Total visited nodes: 184
```

```
======================================================================

It took: 14.32 ms

======================================================================
Results for AStarMinTime:
======================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
----------------------------------------------------------------------
18    21:21:00    DWORZEC GŁÓW   21:23:00    Wzgórze Part    2
114   21:25:00    Wzgórze Part   21:28:00    GALERIA DOMI    7
23    21:29:00    GALERIA DOMI   21:30:00    pl. Nowy Tar    9
23    21:30:00    pl. Nowy Tar   21:31:00    Hala Targowa    10
23    21:31:00    Hala Targowa   21:33:00    pl. Bema        12
19    21:43:00    pl. Bema       21:45:00    Dubois          24
----------------------------------------------------------------------

Total time (AStarMinTime): 00:24:00
Total visited nodes: 408
======================================================================

It took: 5.20 ms

======================================================================
Results for AStarMinTime:
======================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
----------------------------------------------------------------------
19    21:54:00    Dubois         21:56:00    pl. Bema        11
19    21:56:00    pl. Bema       21:58:00    Ogród Botani    13
128   22:06:00    Ogród Botani   22:08:00    Wyszyńskiego    23
----------------------------------------------------------------------

Total time (AStarMinTime): 00:23:00
Total visited nodes: 166
======================================================================
```

2.2.3 Tabu Search sampling (d)

Wybieram losowo k list z sąsiadami, w celu zwiększenia różnorodności poszukiwań

[11]:
```
best_sln = t.sampling_search()
t.set_points(best_sln)
t.go_on_a_trip()
```

```
It took: 9212.04 ms
It took: 0.01 ms
```

```
It took: 0.00 ms
It took: 0.00 ms
It took: 4.34 ms

========================================================================
Results for AStarMinTime:
========================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
------------------------------------------------------------------------
A     20:50:00    Wyszyńskiego   20:52:00    Ogród Botani    2
19    20:58:00    Ogród Botani   20:59:00    Górnickiego     9
19    20:59:00    Górnickiego    21:00:00    Piastowska      10
19    21:00:00    Piastowska     21:04:00    PL. GRUNWALD    14
------------------------------------------------------------------------

Total time (AStarMinTime): 00:14:00
Total visited nodes: 112
========================================================================

It took: 4.58 ms

========================================================================
Results for AStarMinTime:
========================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
------------------------------------------------------------------------
1     21:08:00    PL. GRUNWALD   21:10:00    Piastowska      6
1     21:10:00    Piastowska     21:12:00    Prusa           8
1     21:12:00    Prusa          21:13:00    Wyszyńskiego    9
------------------------------------------------------------------------

Total time (AStarMinTime): 00:09:00
Total visited nodes: 142
========================================================================

It took: 9.48 ms

========================================================================
Results for AStarMinTime:
========================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
------------------------------------------------------------------------
128   21:16:00    Wyszyńskiego   21:18:00    Ogród Botani    5
914   21:19:00    Ogród Botani   21:20:00    Katedra         7
914   21:20:00    Katedra        21:21:00    Urząd Wojewó    8
914   21:21:00    Urząd Wojewó   21:24:00    GALERIA DOMI    11
8     21:24:00    GALERIA DOMI   21:26:00    Wzgórze Part    13
8     21:26:00    Wzgórze Part   21:28:00    DWORZEC GŁÓW    15
```

```
------------------------------------------------------------------------

Total time (AStarMinTime): 00:15:00
Total visited nodes: 280
========================================================================

It took: 10.49 ms

========================================================================
Results for AStarMinTime:
========================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
------------------------------------------------------------------------
K     21:37:00    DWORZEC GŁÓW   21:41:00    GALERIA DOMI    13
11    21:42:00    GALERIA DOMI   21:43:00    pl. Nowy Tar    15
11    21:43:00    pl. Nowy Tar   21:44:00    Hala Targowa    16
11    21:44:00    Hala Targowa   21:46:00    pl. Bema        18
6     21:51:00    pl. Bema       21:53:00    Dubois          25
------------------------------------------------------------------------

Total time (AStarMinTime): 00:25:00
Total visited nodes: 316
========================================================================

It took: 4.88 ms

========================================================================
Results for AStarMinTime:
========================================================================
Line  Departure   Start Stop     Arrival     End Stop       Cost
------------------------------------------------------------------------
19    21:54:00    Dubois         21:56:00    pl. Bema        3
19    21:56:00    pl. Bema       21:58:00    Ogród Botani    5
128   22:06:00    Ogród Botani   22:08:00    Wyszyńskiego    15
------------------------------------------------------------------------

Total time (AStarMinTime): 00:15:00
Total visited nodes: 160
========================================================================
```

2.3 Podsumowanie Tabu

Ogólnie myślę, że udało się rozwiązać ten problem, **różne warianty tabu** dają różne wyniki, czyli modyfikacje w głównym bloku działają.

Wyniki przedstawiają się następująco (dla mojej kompilacji, niektóre mogą się różnić ze względu na użycie liczb pseudolosowych):

Nazwa	Wyniki [min]
Basic Tabu	**90**
Dynamic Tabu	**74**
Aspiracja	**78**
Sampling	**78**

Wynik, który zwrócił standardowy tabu, jest troche słabszy niz pozostałe.

Wyniki zależą także od dobranych parametrów. Prawdopodobnie w łatwiejszych warunkach algorytmy mogłyby zadziałać lepiej. Warto dodać, że testowałem je w trudnych warunkach, ponieważ około godziny 21 zaczyna być coraz mniej połączeń.