Wydział Informatyki i Telekomunikacji

Sztuczna Inteligencja i Inżynieria Wiedzy

Lista nr 1

Jakub Krupiński
255356

1. Wprowadzenie

a. Algorytm Dijkstry

Funkcja kosztu, która została tutaj użyta to czas pomiędzy czasem przyjazdu do stacji końcowej a czasem obecnym (w sekundach).

b. Algorytm A* - kryterium czasu

Funkcja kosztu, która została została tu użyta to czas pomiędzy czasem przyjazdu do stacji końcowej a czasem obecnym (w sekundach), do którego doliczona została wartość odległości Manhattan pomiędzy przystankiem początkowym a końcowym pomnożona o 1000 w celu odpowiedniego wyskalowania wyniku.

c. Algorytm A* - kryterium przesiadek

Funkcja kosztu, która została tu użyta to czas pomiędzy czasem przyjazdu do stacji końcowej a czasem obecnym (w sekundach), do którego doliczana jest wartość '50', jeśli wymagana będzie przesiadka.

d. Algorytm A* - optymalizacje

- Zaimplementowanie kolejki priorytetowej w oparciu o stertę
- Zaimplementowanie grafu w oparciu o słowniki, w celu uzyskania stałego czasu wyszukiwania wartości
- Odrzucanie tych krawędzi, których czas odjazdu jest wcześniejszy niż obecny czas
- Odrzucenie tych krawędzi, których czas odjazdu jest później niż 2h od czasu obecnego

2. Klasy użyte w implementacji

a. Node – wierzchołek grafu, który zawiera w sobie nazwę przystanku, oraz jego długość i szerokość geograficzną.

b. Edge – krawędź grafu, określa połączenie między dwoma wierzchołkami. Zawiera w sobie indeks, nazwę 'firmy' (np. MPK Wrocław), wierzchołki: startowy i końcowy, czas odjazdu oraz czas przyjazdu.

c. Graph – klasa reprezentująca pełen graf, zawiera w sobie słowniki, które przypisują wierzchołki oraz krawędzie do odpowiadającym im nazwom przystanków.

d. DataReader – klasa, która służy mi do odczytania i odpowiedniego wypakowania zawartości pliku 'connection_graph.csv' do obiektów klas 'Node', 'Edge' oraz 'Graph'.

3. Założenia i problemy

Dla ułatwienia założyłem, że wszystkie przystanki o tej samej nazwie można połączyć w jeden przystanek.

Ze względu na brak czasu nie udało mi się zaimplementować przeszukiwania Tabu.

4. Analiza wyników

W ramach analizy otrzymanych przeze mnie wyników stworzyłem funkcję, która zapisuje wyniki przeprowadzonych przeze mnie symulacji do plików o rozszerzeniu .csv. Dla każdego algorytmy wyniki zapisywane są oddzielnie w dwóch miejscach, w dwóch postaciach – pliki „{rodzaj_algorytmu}_output.csv" zapisują pełną trasę wraz z godzinami odjazdów i przyjazdów z i do poszczególnych przystanków, natomiast pliki „{rodzaj_algorytmu}_runtimes.csv" zawierają postaci

uproszczone, gdzie widoczne są jedynie: rodzaj algorytmu, czas szukania rozwiązania oraz parametry wejściowe (przystanek początkowy, końcowy i czas początkowy). Dane które zostają zapisane w plikach nie są pełnymi zestawami – na koniec są one skracane o 5% najszybszych i najwolniejszych czasów wykonania, żeby wyeliminować przypadki skrajne.

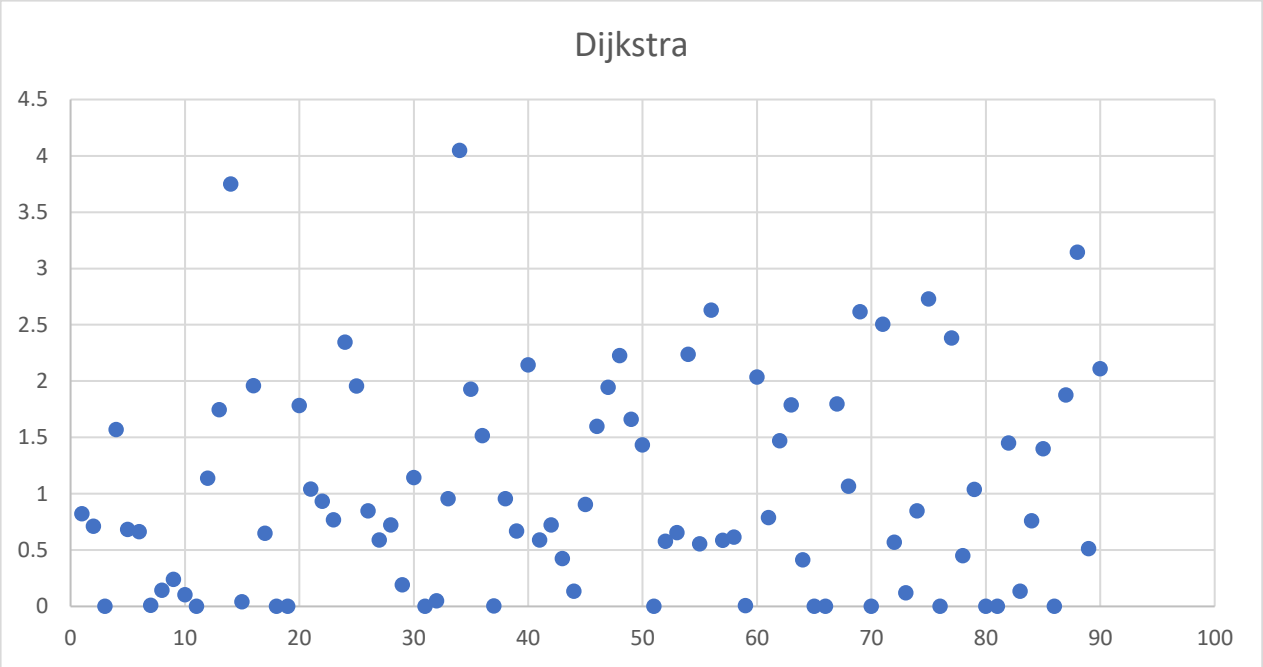
Przykładowa zawartość pliku „dijkstra_outputs.csv”:

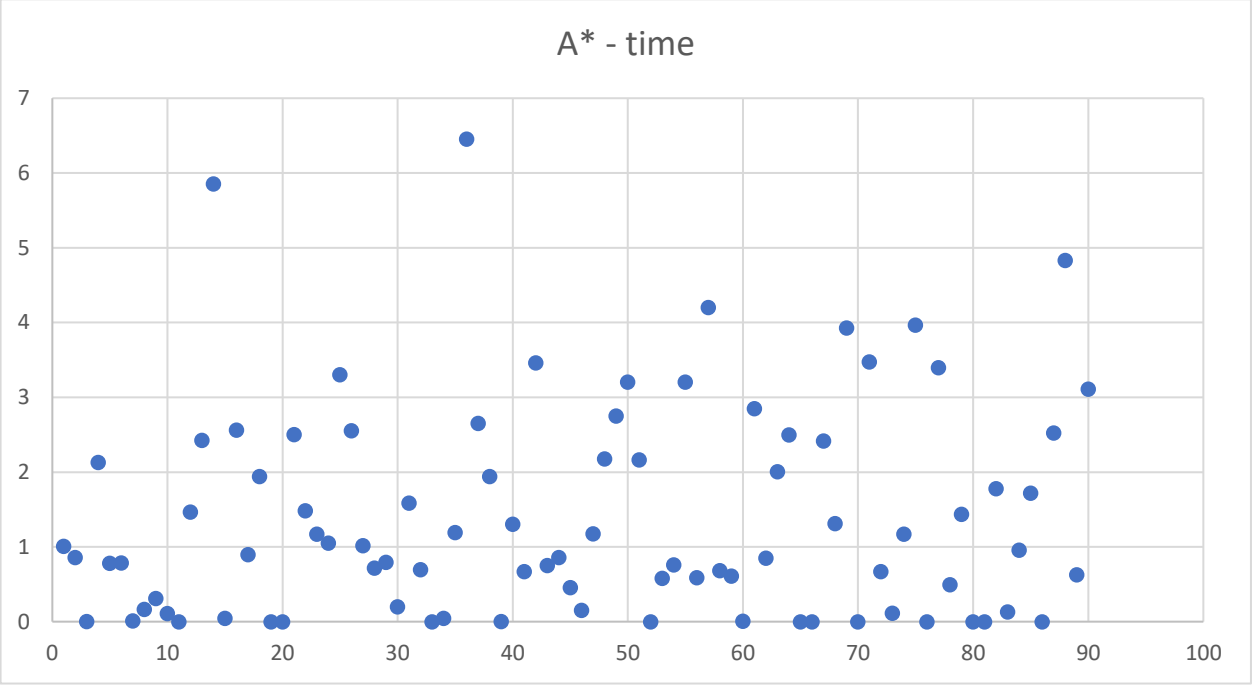
Stop	Runtime	Line	Departure time	Arrival time
DIJKSTRA	0.8219			
From:	Mirkow - Sportowa	To:	Gospodarska	
Mirkow - Sportowa	>			20:50:00
BIERUTOWSKA (Wiadukt)	>	914	21:11:00	21:13:00
Bierutowska 75	>	914	21:13:00	21:14:00
Bierutowska	>	914	21:14:00	21:15:00
Bierutowska 65	>	914	21:15:00	21:15:00
Dobroszycka	>	914	21:15:00	21:16:00
PSIE POLE (Stacja kolejowa)	>	914	21:16:00	21:17:00
Psie Pole (Rondo Lotnikow Polskich)	>	914	21:17:00	21:18:00
Psie Pole	>	914	21:18:00	21:19:00
C.H. Korona	>	914	21:19:00	21:22:00
Brucknera	>	914	21:22:00	21:24:00
Kwidzynska	>	921	21:30:00	21:32:00
Sniadeckich	>	921	21:32:00	21:34:00
Kochanowskiego	>	921	21:34:00	21:36:00
Chopina	>	17	21:42:00	21:43:00
Karlowicza	>	17	21:43:00	21:44:00
Stadion Olimpijski	>	17	21:44:00	21:45:00
8 Maja	>	17	21:45:00	21:46:00
Godebskiego (AWF Wroclaw)	>	17	21:46:00	21:47:00
SEPOLNO	>	17	21:47:00	21:49:00
Monopolowa	>	115	21:56:00	21:57:00
Kolumba	>	115	21:57:00	21:58:00
Magellana	>	115	21:58:00	21:59:00
Swojczyce	>	115	21:59:00	22:00:00
Miloszycka	>	118	22:09:00	22:11:00
Gospodarska	>	118	22:11:00	22:12:00

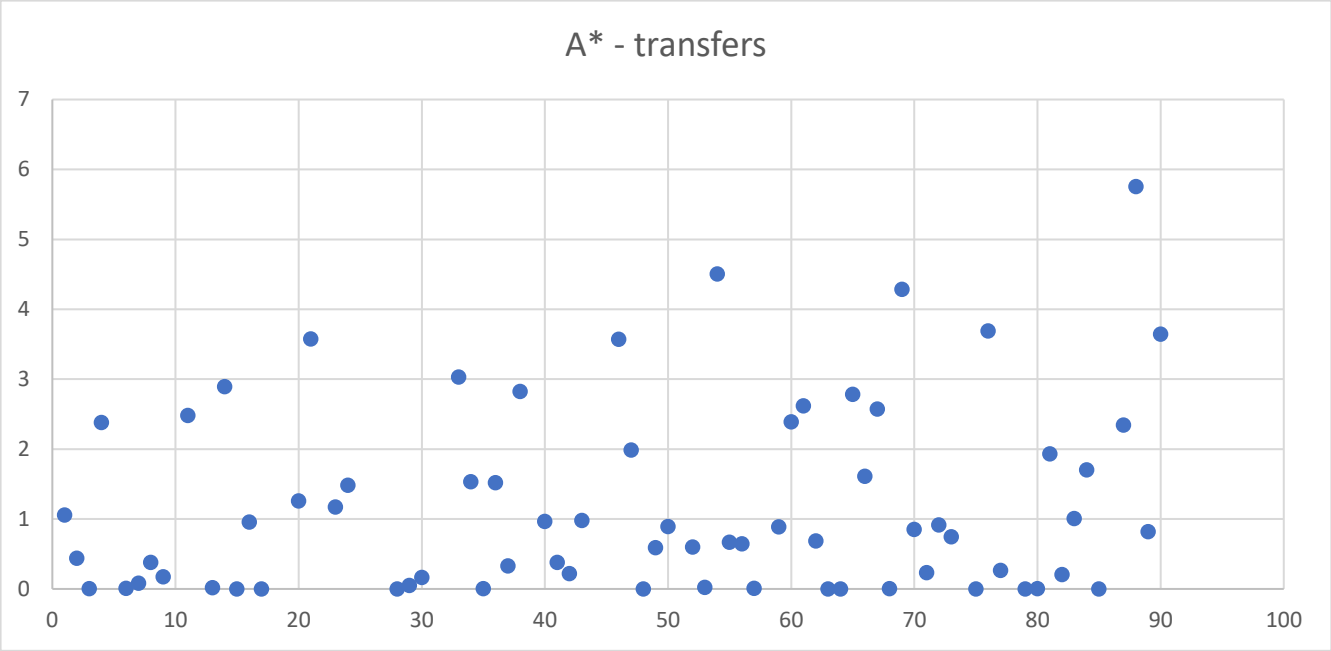
Przykładowa zawartość pliku „dijkstra_runtimes.csv”:

DIJKSTRA	0.8219	Mirkow - Sportowa	--->	Gospodarska	20:50:00
DIJKSTRA	0.711625	Krakowska	--->	ZWYCIESKA	18:50:00
DIJKSTRA	0.000997	Os. Przyjazni	--->	IWINY - petla	00:25:00
DIJKSTRA	1.568806	Wiaduktowa	--->	Krolewiecka	09:16:00
DIJKSTRA	0.68359	KSIEZE MALE	--->	Biestrzykow - Wroclawska	06:03:00
DIJKSTRA	0.662929	Tunelowa	--->	Modra	17:44:00
DIJKSTRA	0.009976	DH Astra	--->	Tarczynski Arena (Lotnicza)	11:59:00
DIJKSTRA	0.143939	Dlugoleka - Kasztanowa	--->	Kielczow - petla/Wroclawska	20:47:00
DIJKSTRA	0.239312	Szczodre - Trzebnicka (na wys. nr	--->	Dubois	06:39:00
DIJKSTRA	0.101768	ROD Oswiata	--->	GIELDOWA (Centrum Hurtu)	15:29:00
DIJKSTRA	0	Zabrodzie - petla	--->	Lenartowice przy posesji nr 11a	00:01:00
DIJKSTRA	1.138236	Waniliowa	--->	Borowska (Szpital)	20:46:00
DIJKSTRA	1.744226	Hala Orbita	--->	Pisarzowice - Kolejowa	19:30:00
DIJKSTRA	3.749194	STABLOWICKA (Osrodek zdrowia	--->	Kepa (na wys. nr 36)	03:36:00
DIJKSTRA	0.039971	Kurlandzka	--->	Sniadeckich	06:41:00
DIJKSTRA	1.95919	Mickiewicza	--->	Pustecka	06:48:00
DIJKSTRA	0.649674	Wilczyce - Wilczycka (Clarena)	--->	FAT	05:58:00
DIJKSTRA	0	IWINY - petla	--->	Galczynskiego	08:47:00
DIJKSTRA	0	SWINIARY	--->	Mosty Warszawskie	00:12:00
DIJKSTRA	1.783252	Monte Cassino	--->	REDZIN	19:56:00
DIJKSTRA	1.039581	Hippiczna	--->	Awicenny (Stacja kolejowa)	13:21:00
DIJKSTRA	0.931713	OPOROW	--->	Kozanowska	05:40:00
DIJKSTRA	0.769028	Kepinska	--->	Kopanskiego	17:26:00
DIJKSTRA	2.343732	Pisarzowice - Kolejowa	--->	Siedlec - skrzy. Osiedlowa	07:51:00
DIJKSTRA	1.954814	Wyscigowa	--->	Goslawice - przejazd PKP	20:30:00
DIJKSTRA	0.847984	Gorlicka	--->	Ksieska	18:11:00
DIJKSTRA	0.587508	GRABISZYNSKA (Cmentarz II)	--->	Wislanska	16:55:00
DIJKSTRA	0.721156	ZACHODNIA (Stacja kolejowa)	--->	Kasprowicza	21:39:00
DIJKSTRA	0.191473	Marcepanowa	--->	Trawowa	12:29:00
DIJKSTRA	1.142097	Trawowa	--->	RATYN	15:13:00
DIJKSTRA	0	Budziwojowice	--->	Pietrzykowice - zaklad	02:40:00
DIJKSTRA	0.049946	Jerzmanowska nr 9	--->	Wilkszyn - Polna	23:29:00
DIJKSTRA	0.956624	Muchobor Wielki	--->	JARNOLTOW	21:36:00
DIJKSTRA	4.046067	Kielczow - petla/Wroclawska	--->	Szymanow	12:19:00
DIJKSTRA	1.926813	Kielczow - petla (Plac Jana Gdaka	--->	Rakow - osiedle	18:43:00
DIJKSTRA	1.515299	Zagrodnicza	--->	Kosmonautow (Szpital)	04:59:00

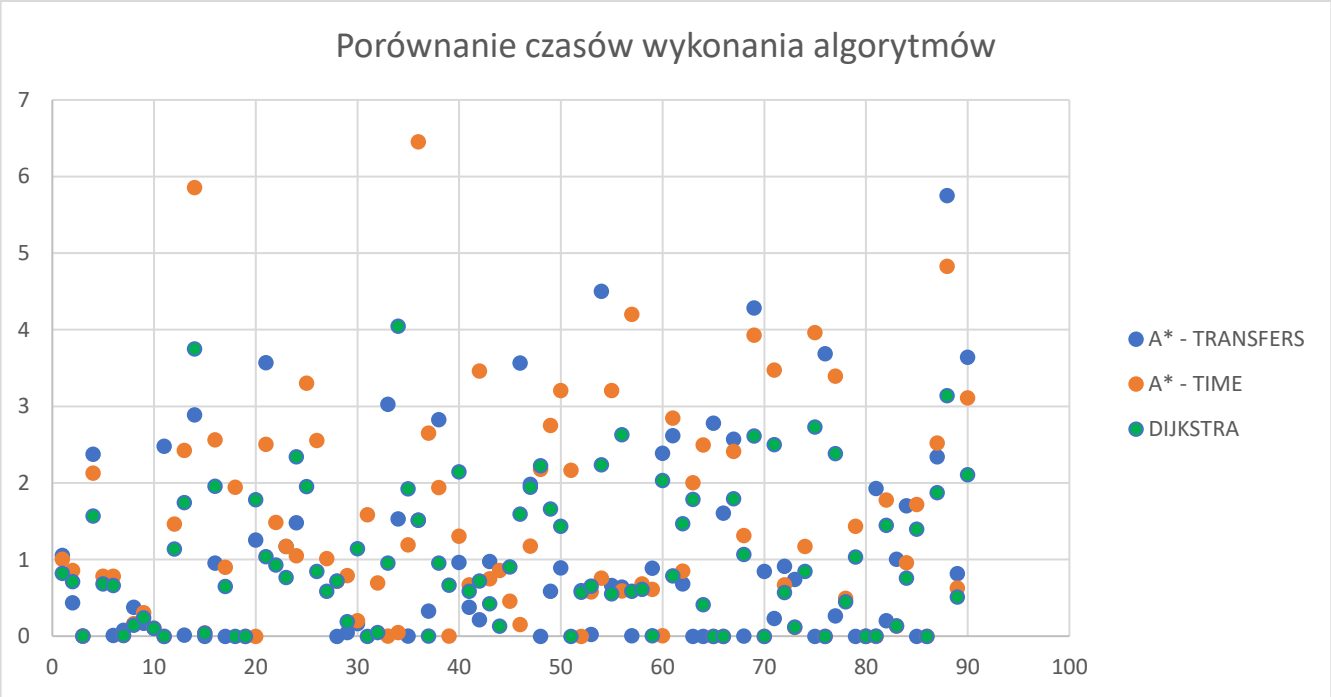
Na podstawie zawartości plików „...runtimes.csv” wygenerowałem następujące wykresy, na których widać jak rozkładały się czasy szukania ścieżki pomiędzy iteracjami:


Dijkstra
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0
10
20
30
40
50
60
70
80
90
100


A* - time
7
6
5
4
3
2
1
0
0
10
20
30
40
50
60
70
80
90
100



Połączenie ze sobą danych z trzech tych plików zaowocowało tak wyglądającym wykresem punktowym:



Nie jest to idealna reprezentacja tych danych, która nie jest może najbarzdziej przejrzysta, widać jednak na niej wciąż parę zastanawiających rzeczy:

Po pierwsze, wyniki algorytmu A* na podstawie przesiadek radzi sobie najgorzej ze wszystkich, jako że najczęściej zdarzają się w jego przypadku wartości 0.0 oraz float('inf') (które – na potrzeby wygenerowania wykresu – zostały przeze mnie wyfiltrowane, przez co niebieskich kropek jest na tym wykresie mniej), jako że algorytm ten w mojej implementacji nie zawsze z sukcesem znajdywał

odpowiednią ścieżkę. Algorytm Dijkstra radzi sobie zaskakująco dobrze, co również sugeruje, że być może zaproponowane przeze mnie rozwiązanie A* w kryterium czasu nie należy do optymalnych, jako że zaimplementowanie heurystyki powinno było spowodować znacznie krótsze czasy przeszukiwania.

Ponizej pokazane sa przykladowe wyniki tej samej trasy dla algorytmow A* z kryterium przesiadkowym oraz czasu:

A* - TRANSFERS	1.25533			
From:	Kepinska	To:	Kopanskiego	
Kepinska	>			17:26:00
Kamienskiego	>	74	17:27:00	17:28:00
Broniewskiego	>	74	17:28:00	17:30:00
Pola	>	132	17:36:00	17:37:00
Syrokomli	>	132	17:37:00	17:38:00
Kasprowicza	>	132	17:38:00	17:39:00
pl. Danilowskiego	>	132	17:39:00	17:41:00
Berenta	>	132	17:41:00	17:42:00
KROMERA	>	132	17:42:00	17:47:00
Kromera (Czajkowskiego)	>	N	17:57:00	17:59:00
Grudziadzka	>	N	17:59:00	18:00:00
Brucknera	>	N	18:00:00	18:01:00
Psie Pole	>	N	18:01:00	18:05:00
Psie Pole (Rondo Lotnikow Pol	>	N	18:05:00	18:07:00
Zakrzowska	>	151	18:08:00	18:09:00
Kopanskiego	>	151	18:09:00	18:10:00

599	A* - TIME	1.051844			
600	From:	Kepinska	To:	Kopanskiego	
601	Kepinska	>			17:26:00
602	Kamienskiego	>	74	17:27:00	17:28:00
603	Broniewskiego	>	74	17:28:00	17:30:00
604	Pola	>	132	17:36:00	17:37:00
605	Syrokomli	>	132	17:37:00	17:38:00
606	Kasprowicza	>	132	17:38:00	17:39:00
607	pl. Danilowskiego	>	132	17:39:00	17:41:00
608	Berenta	>	132	17:41:00	17:42:00
609	KROMERA	>	118	17:44:00	17:47:00
610	Kwidzynska	>	118	17:47:00	17:51:00
611	Brucknera	>	118	17:51:00	17:53:00
612	C.H. Korona	>	121	17:55:00	17:57:00
613	Zielna	>	121	17:57:00	17:59:00
614	Psie Pole	>	121	17:59:00	18:00:00
615	Psie Pole (Rondo Lotnikow Polskich)	>	121	18:00:00	18:02:00
616	Kopanskiego	>	D	18:06:00	18:08:00

Jak widac na zdjeciu, kryterium przesiadkowe pozwala na dotarcie na miejsce nieco pozniej, jednak pozwala tez na zmniejszenie ilosci przesiadek – cztery przesiadki zamiast pieciu.