Języki Skryptowe

Dokumentacja projektu Plan Metra z Olimpiady Informatycznej XXV

Michał Osiewicz, grupa 1B

Wydział Matematyki Stosowanej

Kierunek Informatyka

Semestr III

Studia Stacjonarne

17 stycznia 2021

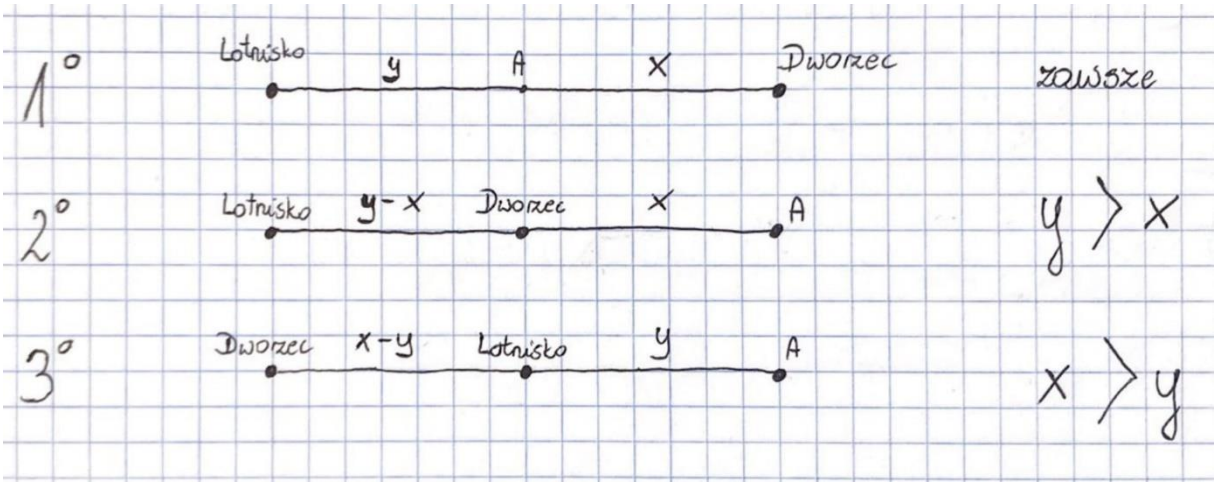
Część I

Opis programu

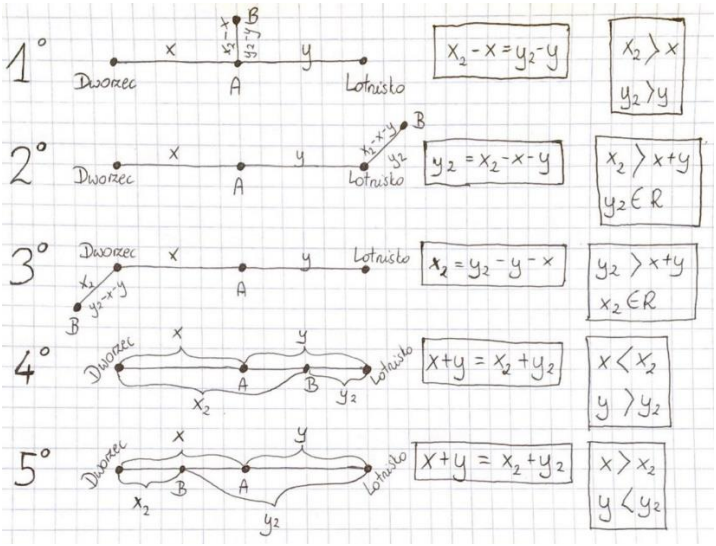
Bajtazar udał się na wycieczkę do Bajtogrodu i postanowił, że podczas całego pobytu w mieście będzie poruszał się metrem. Wysiadłszy na dworcu kolejowym (przy którym znajduje się jedna ze stacji metra), poszedł zakupić bilety w automacie. Z cennika wynikało, że przejazd na trasie prowadzącej od stacji „Dworzec kolejowy” do stacji „Lotnisko” jest darmowy, natomiast dla wszystkich pozostałych tras cena biletu jest równa odległości pomiędzy stacją początkową a stacją docelową. Dla wygody pasażerów przyjeżdżających do Bajtogrodu, przy automacie była wypisana lista cen biletów dla tras od stacji „Dworzec kolejowy” do wszystkich pozostałych stacji oraz dla tras od stacji „Lotnisko” do wszystkich pozostałych stacji. Bajtazar dowiedział się też, że metro posiada n stacji, połączonych oszczędną siecią n − 1 tuneli dodatniej długości (wystarczających jednak do przejechania z dowolnej stacji do dowolnej innej). Na podstawie tych wszystkich informacji nasz bohater chciałby wyznaczyć połączenia pomiędzy stacjami lub stwierdzić, że posiadane przez niego dane są błędne.

Opis działania

Program rozwiązuje zadanie wybierając pierwszą stację(A)(oprócz Dworca i Lotniska) i łączy ją ze stacją Dworzec i Lotnisko, na jeden lub dwa sposoby w zależności od odległości Dworca od stacji A(x) oraz Lotniska od stacji A(y). Algorytm jest przeprowadzany dla każdej stacji jako pierwszej(A). Poniżej prezentuje możliwe umiejscowienie stacji ze względu na x i y.

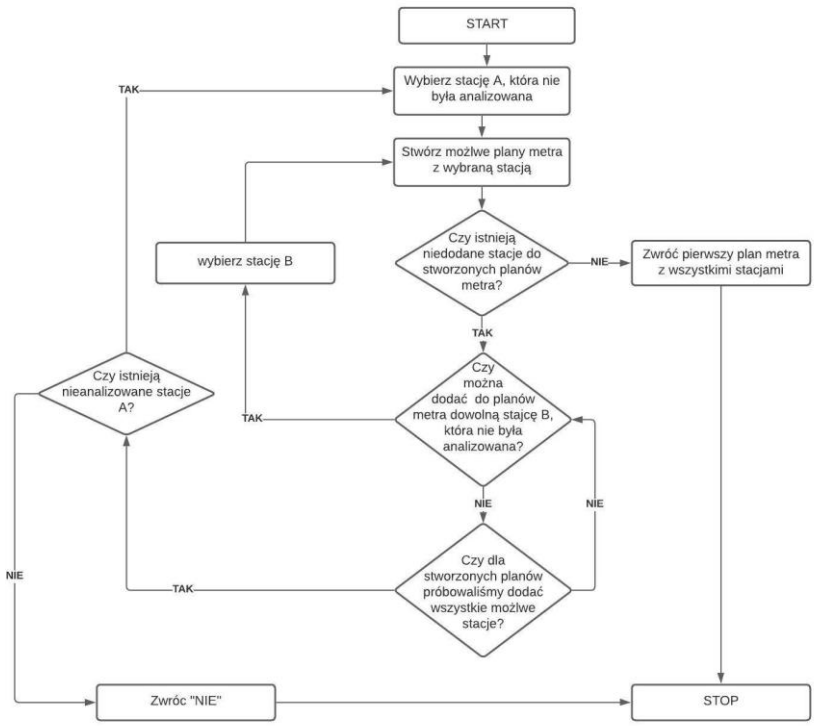


Algorytm dla utworzonego planu metra próbuje dołączyć kolejna stację(B), badając zależność między odległością Dworca od stacji B(x2) oraz Lotniska od stacji B(y2) a x i y z pierwszego kroku algorytmu. Poniżej prezentuje możliwe umiejscowienia stacji B ze względu na x,y,x2,y2.

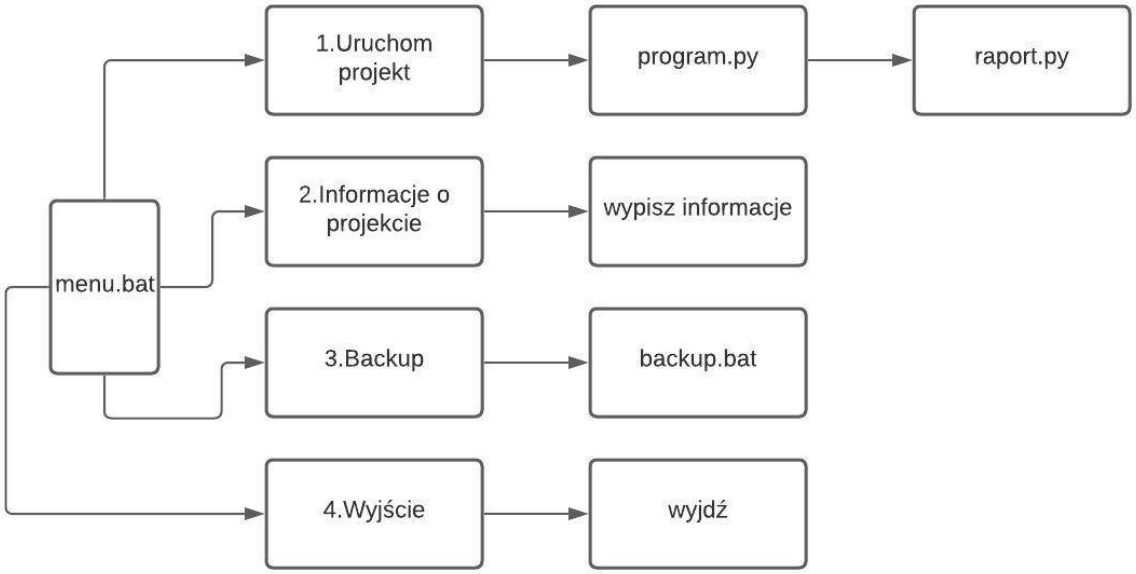


Algorytm po udanym dodaniu stacji próbuje dołączyć kolejną stacje w taki sam sposób jak stację B, algorytm próbuje dołączyć każdą niedodaną stację na każdym kroku. Po udanym dołączeniu ostatniej stacji algorytm zwraca plan metra i kończy swoją pracę. W przypadku przeanalizowania wszystkich możliwych przypadków i braku planu metra ze wszystkimi stacjami algorytm zwraca informację o braku możliwości utworzenia planu metra z podanych danych wejściowych.

Algorytm

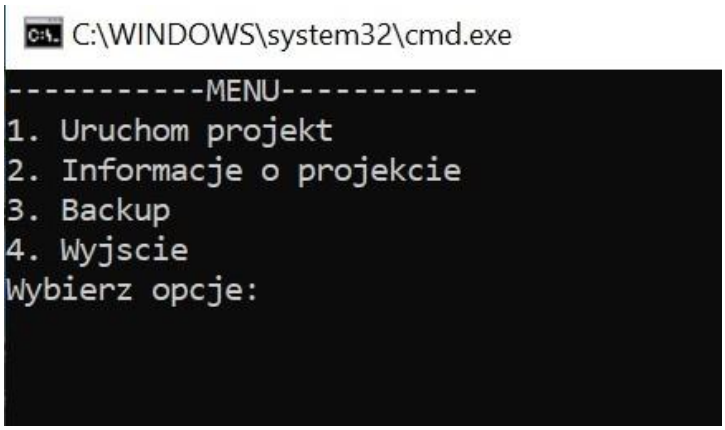


Schemat działania



Instrukcja obsługi

Program uruchamiamy włączając plik menu.bat. Po uruchomieniu pliku ukazuje nam się menu programu. Wybieramy jedną z czterech opcji wpisując odpowiadający im numer(1,2,3,4).



Do poprawnego działania programu należy utworzyć w katalogu „in” pliki tekstowe, nazywając je „wejsciei.txt”, gdzie i to cyfra. Dla każdego takiego pliku zostanie utworzony plik „wyjsciei.txt” w katalogu „out” z rozwiązaniem danego przykładu. Poniżej zostały opisane wytyczne jak powinien wyglądać plik wejścia oraz jak będzie wyglądał plik wyjścia.

Wejście

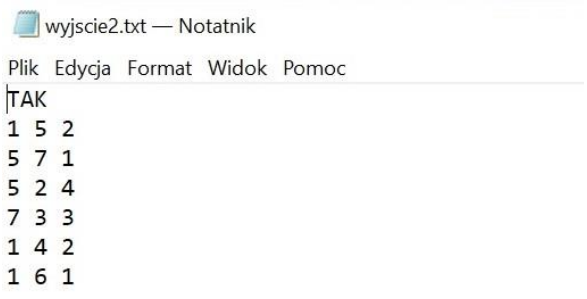
Pierwszy wiersz standardowego wejścia zawiera jedną dodatnią liczbę całkowitą n oznaczającą liczbę stacji metra w Bajtogrodzie. Stacje są numerowane liczbami od 1 do n, przy czym stacja „Dworzec kolejowy” ma numer 1, a stacja „Lotnisko” ma numer n. W drugim wierszu znajduje się ciąg n − 2 liczb całkowitych d2, d3, . . . , dn−1 z przedziału [1, 1 000 000], pooddzielanych pojedynczymi odstępami; i-ta liczba w ciągu oznacza cenę biletu na trasie od stacji „Dworzec kolejowy” do stacji o numerze i. Cena ta jest równa długości trasy. W trzecim wierszu znajduje się ciąg l2, l3, . . . , ln−1 w takim samym formacie, opisujący ceny biletów na trasach od stacji „Lotnisko”.

Przykład:

Wyjście

Jeżeli nie istnieje żaden plan połączeń pomiędzy stacjami zgodny z informacjami uzyskanymi przez Bajtazara, na standardowe wyjście należy wypisać jeden wiersz ze słowem NIE. W przeciwnym wypadku w pierwszym wierszu należy wypisać jedno słowo TAK, a w kolejnych n−1 wierszach połączenia pomiędzy stacjami. Każdy z tych wierszy powinien zawierać trzy liczby całkowite a, b i c oddzielone pojedynczymi odstępami, co oznacza, że stacje o numerach a i b są połączone tunelem o długości c. Jeżeli jest więcej niż jedna poprawna odpowiedź, Twój program powinien wypisać dowolną z nich.

Przykład:



Źródło: https://szkopul.edu.pl/problemset/problem/vvd6w7n7EXFVEg3nkqGxEirV/site/?key=statement

Dodatkowe informacje

Program po wykonaniu swojego zadania tworzy raport w postaci strony www. Po wybraniu opcji „Backup” w menu skrypt tworzy kopię zapasową całego folderu w którym znajduje się projekt.

System operacyjny: Windows

Interpreter języka python: wersja 3.9 lub nowsza

Część II

Implementacja

Menu.bat

Menu sprawdza która opcje wybrał użytkownik i wykonuje odpowiednie zadanie (schemat projektu).

```
@ echo off
:MENU
echo ----------MENU----------
echo 1. Uruchom projekt
echo 2. Informacje o projekcie
echo 3. Backup
echo 4. Wyjscie

set /p wybor=Wybierz opcje:

if %wybor%==1 (
goto START
)
if %wybor%==2 (
goto INFO
)
if %wybor%==3 (
goto BACK
)
if %wybor%==4 (
goto WY
) else (
echo Niepoprawna wartosc.
pause
cls
goto MENU
)
:START
for %%X in (in\wejscie*.txt) do (call program.py %%X)
call raport.py
pause
cls
goto MENU
:INFO
echo Projekt Plan Metra. Autor: Michal Osiewicz
pause
cls
goto MENU
:BACK
call backup.bat
pause
cls
goto MENU
:WY
exit
```

Backup.bat

Tworzy kopię zapasową całego folderu w którym znajduje się projekt.

```
set data=%DATE%
mkdir Backup%data%
copy *.* Backup%data%
xcopy in Backup%data%\in\
xcopy out Backup%data%\out\
```

Program.py

Wykonuje główny algorytm dla danych wejściowych i zwraca wynik. Program sprawdza czy został podany plik wejścia i wywołuje klasę Metro i jej metodę.

```
if(len(sys.argv)==2):
    M=Metro(sys.argv[1])
    M.plan()
else:
    print("Błędne argumenty wejścia.")
```

Metody klasy Metro:

Metoda init rozpoczyna prace programu, sprawdza czy plik wejściowy istnieje, wczytuje dane z pliku wejściowego, sprawdza czy dane są poprawnie zapisane.

```
def __init__(self,dane):
    self.indeks = dane[len(dane) - 5]
    try:
        wejscie = open(dane, "r")
        self.ilosc_stacji = int(wejscie.read(1))
        self.odleglosc_dworzec = []
        self.odleglosc_lotnisko = []
        self.wynik=[]
        liczba = ""
        while (len(self.odleglosc_dworzec) != self.ilosc_stacji - 2):
            znak = wejscie.read(1)
            if (znak != " " and znak != "\n"):
                liczba += znak
            elif (len(liczba) > 0):
                self.odleglosc_dworzec.append(int(liczba))
                liczba = ""
        liczba = ""
        while (len(self.odleglosc_lotnisko) != self.ilosc_stacji - 2):
            znak = wejscie.read(1)
            if (znak != " " and znak != "\n" and znak != ""):
                liczba += znak
            elif (len(liczba) > 0):
                self.odleglosc_lotnisko.append(int(liczba))
                liczba = ""
        wejscie.close()
    except:
        wyjscie = open("out\\wyjscie" + self.indeks + ".txt", "w")
        wyjscie.write("Źle zapisany plik wejscia.")
        wyjscie.close()
        exit()
```

Następnie wykonuje się metoda plan która dodaje pierwsze stacje(A) na różne sposoby i tworzy plany metra. Dla każdej stacji A wywołuje metodę algorytm. W przypadku przejścia wszystkich przypadków zwraca „NIE” na plik wyjściowy.

```
def plan(self):
    dodane_stacje = []
    wynik1 = []
    wynik2 = []
    dodane_D = []
    dodane_L = []
    dodane_stacje_D = []
    dodane_stacje_L = []
    for i in range(self.ilosc_stacji - 2):
        dodane_stacje.append(i)
        odleglosc_DL = self.poczatek_A(i, wynik1)
        self.algorytm(i, dodane_stacje, odleglosc_DL, wynik1, dodane_D, dodane_L, dodane_stacje_D, dodane_stacje_L)
        if (self.odleglosc_lotnisko[i] > self.odleglosc_dworzec[i]):
            odleglosc_DL = self.poczatek_D(i, wynik2)
            self.algorytm(i, dodane_stacje, odleglosc_DL, wynik2, dodane_D, dodane_L, dodane_stacje_D, dodane_stacje_L)
        else:
            odleglosc_DL = self.poczatek_L(i, wynik2)
            self.algorytm(i, dodane_stacje, odleglosc_DL, wynik2, dodane_D, dodane_L, dodane_stacje_D, dodane_stacje_L)
        wynik1.clear()
        wynik2.clear()
        dodane_stacje.clear()
    wyjscie = open("out\wyjscie"+self.indeks+".txt", "w")
    wyjscie.write("NIE")
    wyjscie.close()
```

Metoda algorytm dodaje do stworzonego planu stację(B) sprawdza wszystkie możliwości oraz każda stację następnie po dodaniu stacji tworzy nowy plan metra i wywołuje sama siebie dopóki nie przeanalizuje wszystkich możliwości, jeśli znajdzie rozwiązanie wywołuje metodę wyjscie i kończy prace programu.

```
def algorytm(self, i, dodane, odl, tab, dodane_D, dodane_L, d_stacje_D, d_stacje_L):
    if (len(tab) == self.ilosc_stacji - 1):
        self.wyjscie(tab)
        exit()
    tab1 = []
    tab2 = []
    tab3 = []
    tab4 = []
    tab5 = []
    tablica_dodatkowa_D = []
    tablica_dodatkowa_L = []
    for x in range(len(dodane_D)):
        tablica_dodatkowa_D.append([])
        for j in range(len(tab)):
            tablica_dodatkowa_D[x].append(tab[j])
    for x in range(len(dodane_L)):
        tablica_dodatkowa_L.append([])
        for j in range(len(tab)):
            tablica_dodatkowa_L[x].append(tab[j])
    for x in range(len(tab)):
        tab1.append(tab[x])
        tab2.append(tab[x])
        tab3.append(tab[x])
        tab4.append(tab[x])
        tab5.append(tab[x])
    licznik = len(tab)
    for j in range(self.ilosc_stacji - 2):
        if (dodane.count(j) == 0):
            self.dodanie_do_A(self.odleglosc_dworzec[i], self.odleglosc_lotnisko[i],
                              self.odleglosc_dworzec[j], self.odleglosc_lotnisko[j], i + 2, j + 2, tab1)
            for x in range(len(dodane_D)):
                self.dodanie_do_A(self.odleglosc_dworzec[d_stacje_D[x] - 2], self.odleglosc_lotnisko[d_stacje_D[x] - 2],
                                  self.odleglosc_dworzec[j], self.odleglosc_lotnisko[j], d_stacje_D[x], j + 2,
                                  tablica_dodatkowa_D[x])
            for x in range(len(dodane_L)):
                self.dodanie_do_A(self.odleglosc_dworzec[d_stacje_L[x] - 2], self.odleglosc_lotnisko[d_stacje_L[x] - 2],
                                  self.odleglosc_dworzec[j], self.odleglosc_lotnisko[j], d_stacje_L[x], j + 2,
                                  tablica_dodatkowa_L[x])
            self.dodanie_do_D(odl, self.odleglosc_dworzec[j], self.odleglosc_lotnisko[j], j + 2, tab2)
            self.dodanie_do_L(odl, self.odleglosc_dworzec[j], self.odleglosc_lotnisko[j], j + 2, tab3)
            if (tab[0][2] == self.odleglosc_dworzec[i] or len(dodane_D) > 0 or len(dodane_L) > 0):
                self.dodanie_do_Lsr(self.odleglosc_dworzec[i], self.odleglosc_lotnisko[i],
                                    self.odleglosc_dworzec[j], self.odleglosc_lotnisko[j], j + 2, tab4, dodane_D, d_stacje_D)
                self.dodanie_do_Psr(self.odleglosc_dworzec[i], self.odleglosc_lotnisko[i],
                                    self.odleglosc_dworzec[j], self.odleglosc_lotnisko[j], j + 2, tab5, dodane_L, d_stacje_L)
```

```
if (len(tab1) > licznik):
    dodane.append(j)
    self.algorytm(i, dodane, odl, tab1, dodane_D, dodane_L, d_stacje_D, d_stacje_L)
    tab1.pop(len(tab1) - 1)
    licznik = len(tab)
if (len(tab2) > licznik):
    dodane.append(j)
    self.algorytm(i, dodane, odl, tab2, dodane_D, dodane_L, d_stacje_D, d_stacje_L)
    tab2.pop(len(tab2) - 1)
    licznik = len(tab)
if (len(tab3) > licznik):
    dodane.append(j)
    self.algorytm(i, dodane, odl, tab3, dodane_D, dodane_L, d_stacje_D, d_stacje_L)
    tab3.pop(len(tab3) - 1)
    licznik = len(tab)
if (len(tab4) > licznik):
    dodane.append(j)
    self.algorytm(i, dodane, odl, tab4, dodane_D, dodane_L, d_stacje_D, d_stacje_L)
    tab4.pop(len(tab4) - 1)
    dodane_D.pop(len(dodane_D) - 1)
    d_stacje_D.pop(len(d_stacje_D) - 1)
    licznik = len(tab)
if (len(tab5) > licznik):
    dodane.append(j)
    self.algorytm(i, dodane, odl, tab5, dodane_D, dodane_L, d_stacje_D, d_stacje_L)
    tab5.pop(len(tab5) - 1)
    dodane_L.pop(len(dodane_L) - 1)
    d_stacje_L.pop(len(d_stacje_L) - 1)
    licznik = len(tab)
for x in range(len(dodane_D)):
    if (len(tablica_dodatkowa_D[x]) > licznik):
        dodane.append(j)
        self.algorytm(i, dodane, odl, tablica_dodatkowa_D[x], dodane_D, dodane_L, d_stacje_D, d_stacje_L)
        tablica_dodatkowa_D[x].pop(len(tablica_dodatkowa_D[x]) - 1)
        licznik = len(tab)
for x in range(len(dodane_L)):
    if (len(tablica_dodatkowa_L[x]) > licznik):
        dodane.append(j)
        self.algorytm(i, dodane, odl, tablica_dodatkowa_L[x], dodane_D, dodane_L, d_stacje_D, d_stacje_L)
        tablica_dodatkowa_L[x].pop(len(tablica_dodatkowa_L[x]) - 1)
        licznik = len(tab)
```

Metoda wynik zapisuje plan metra do pliku wyjściowego.

```
def wyjscie(self,wynik):
    wyjscie=open("out\wyjscie"+self.indeks+".txt","w")
    wyjscie.write("TAK\n")
    for i in range(len(wynik)):
        for j in range(len(wynik[i])):
            wyjscie.write(str(wynik[i][j]))
            if(j!=len(wynik[i])-1):
                wyjscie.write(" ")
        if(i!=len(wynik)-1):
            wyjscie.write("\n")
    wyjscie.close()
```

Raport.py

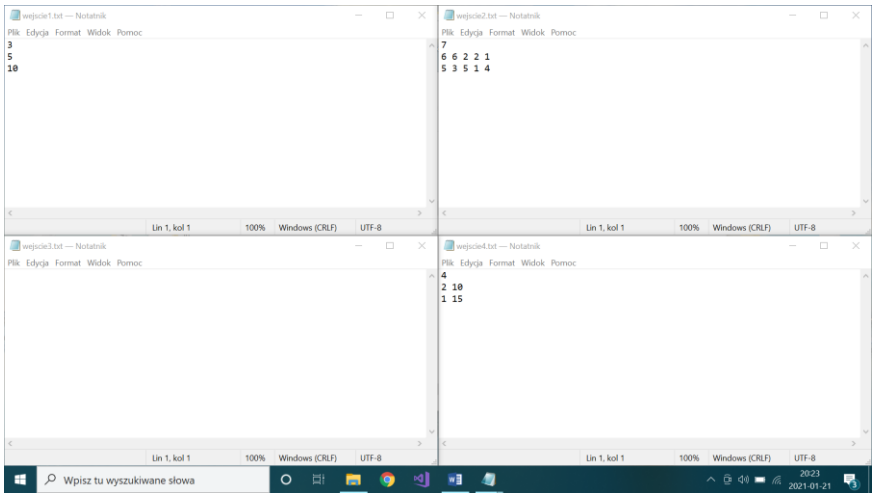
Skrypt dla danych wejściowych i uzyskanych danych wyjściowych tworzy raport w postaci prostej strony html.

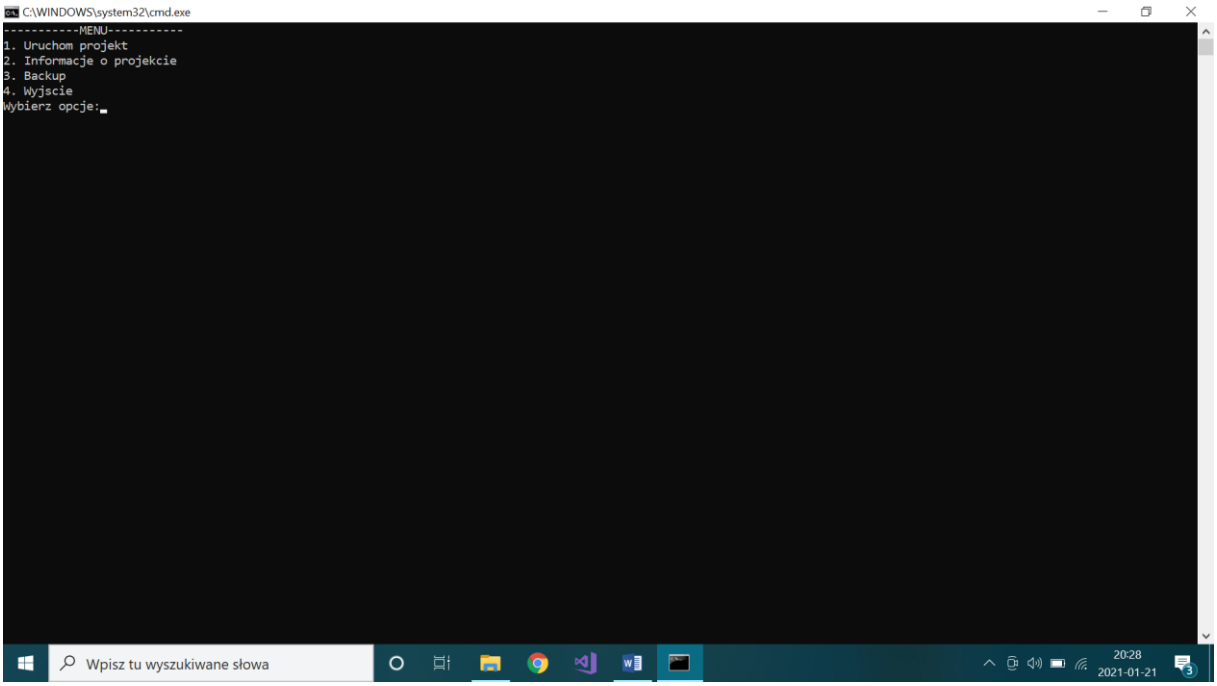
```
import os
import datetime

lista_plikow_in=list(os.listdir("in"))
lista_plikow_out=list(os.listdir("out"))
data=datetime.datetime.now()

html="<html>\n<head>\n<title>Raport projektu</title>\n<style>\n#ekran\n{\nwidth: 500px;\nmargin-left:auto;\n"
html+="margin-right:auto;\n}\n#wstep\n{\npadding: 10px;\n}\n#wejscie\n{\nfloat: left;\nwidth: 230px;\n"
html+="padding: 10px;\n}\n#wyjscie\n{\nfloat: left;\nwidth: 230px;\npadding: 10px;\n}\n</style>\n</head>\n"
html+='<body>\n<div id="ekran">\n<div id="wstep">\n<h1>Raport projektu Plan Metra</h1>\n'
html+='Liczba plików wejścia: '+str(len(lista_plikow_in))+'<br /><br />\nData wykonania raportu: '+str(data)+'\n'
html+='<br /><br />\nAutor: Michał Osiewicz\n</div>\n<div id="wejscie">\n<h2>Wejscia:</h2>\n'
html_in=""
html_out=""
for i in range(len(lista_plikow_in)):
    html_in+="<h3>"+lista_plikow_in[i]+"</h3>\n"
    html_out+="<h3>"+lista_plikow_out[i]+"</h3>\n"
    plik_in=open("in/"+lista_plikow_in[i],"r")
    plik_out=open("out/"+lista_plikow_out[i],"r")
    linie_in=plik_in.readlines()
    linie_out=plik_out.readlines()
    plik_in.close()
    plik_out.close()
    x=0
    while(x<len(linie_in) or x<len(linie_out)):
        if(x<len(linie_in)):
            html_in+=linie_in[x]
        if (x < len(linie_out)):
            html_out+=linie_out[x]
        html_in+="<br />\n"
        html_out+="<br />\n"
        x+=1
html+=html_in+'</div>\n<div id="wyjscie">\n<h2>Wyjscia:</h2>\n'+html_out+'</div>\n</div>\n</body>\n</html>'

strona=open("raport.html","w")
strona.write(html)
```

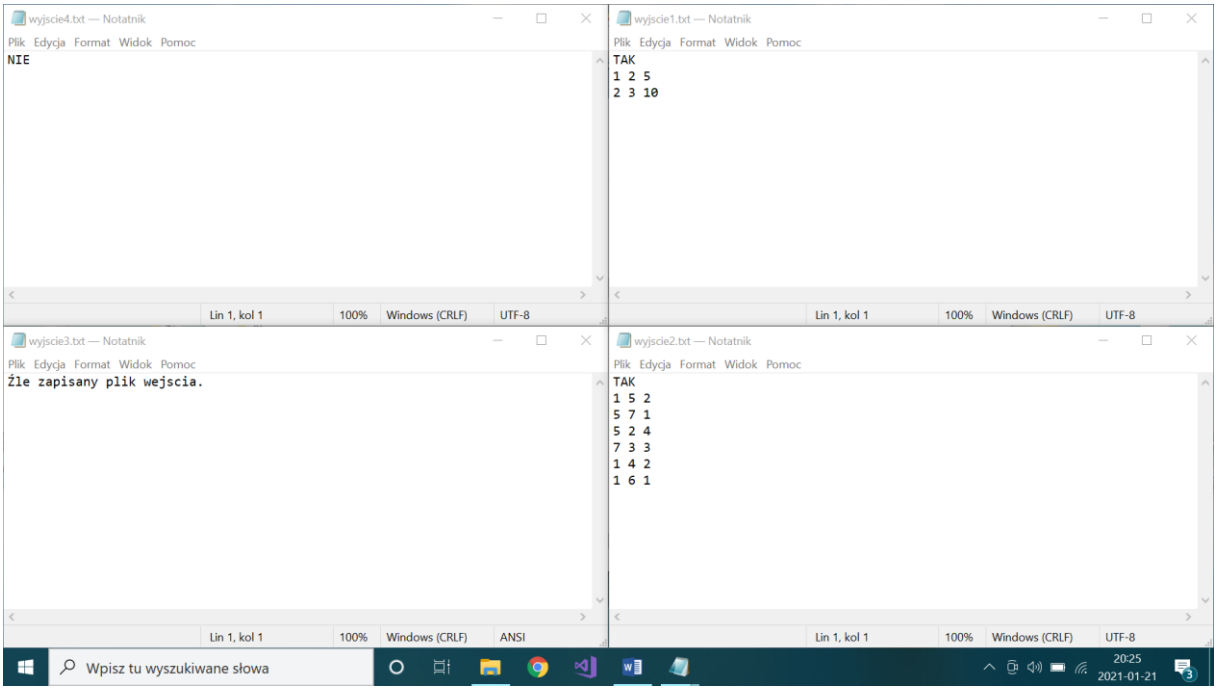
Testy

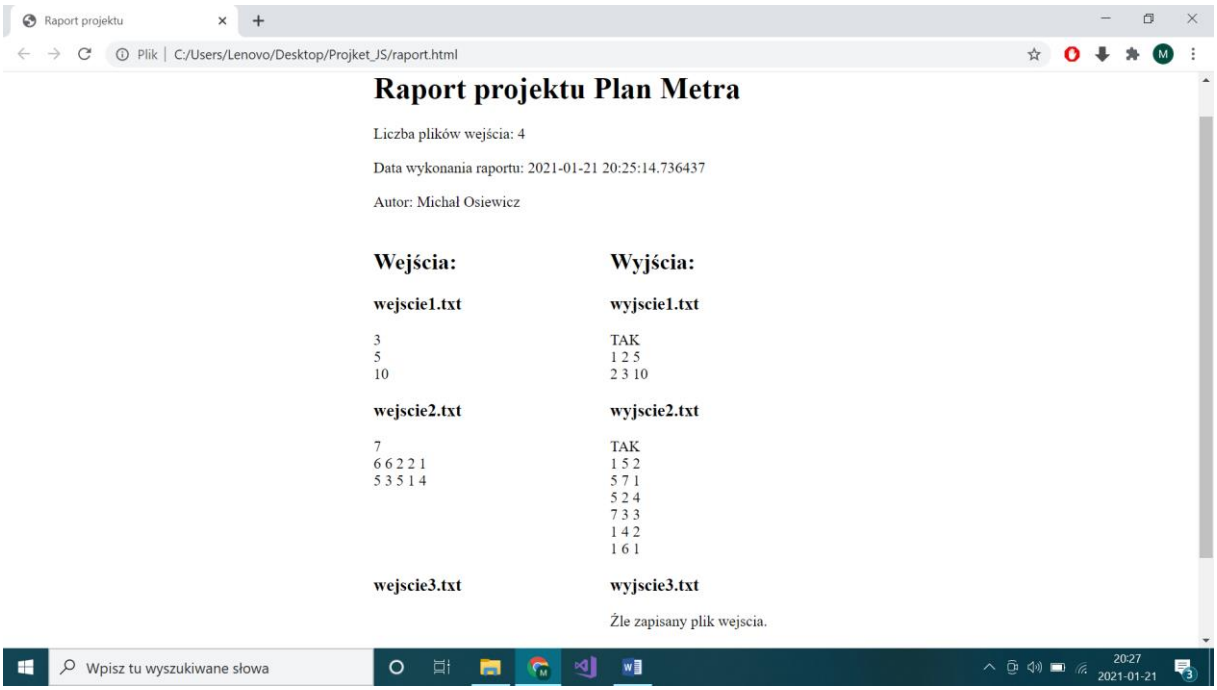
Program został przetestowany na systemie operacyjnym Windows 10 oraz interpreterze języka python 3.9 dla różnych danych zapisanych poprawnie oraz niepoprawnie.

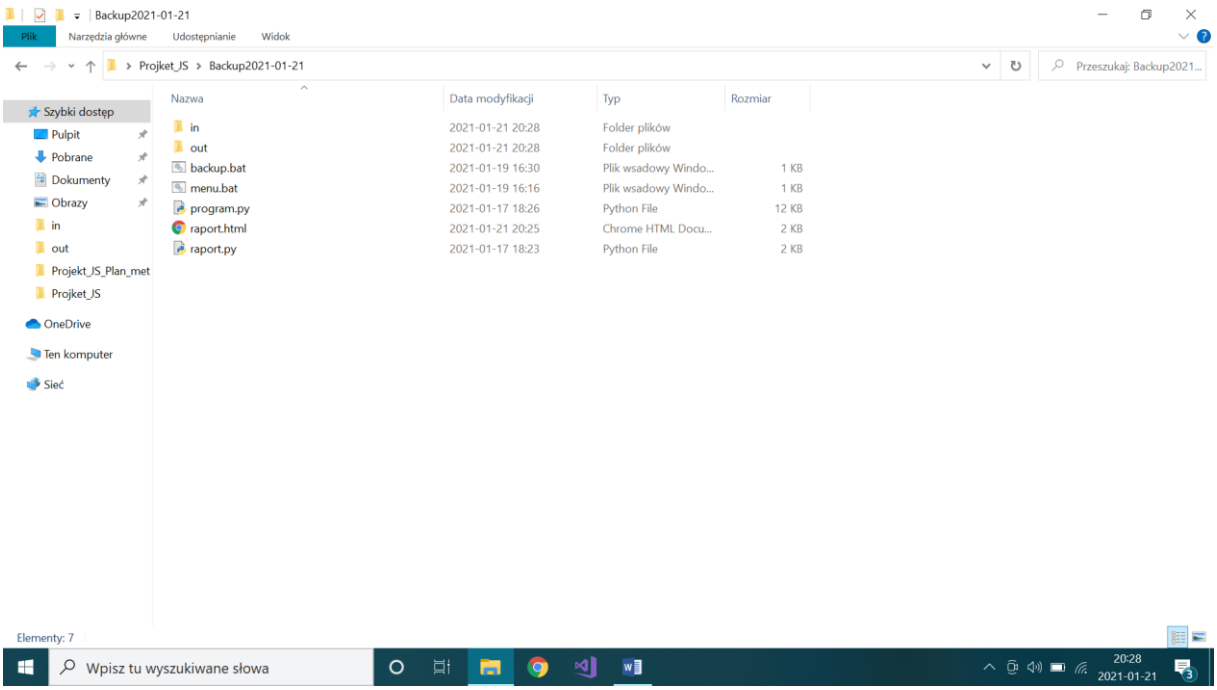
Przykład:




C:\WINDOWS\system32\cmd.exe
----------MENU----------
1. Uruchom projekt
2. Informacje o projekcie
3. Backup
4. Wyjscie
Wybierz opcje:
Wpisz tu wyszukiwane słowa
20:28
2021-01-21


wyjscie4.txt — Notatnik
Plik Edycja Format Widok Pomoc
NIE
Lin 1, kol 1
100%
Windows (CRLF)
UTF-8
wyjscie1.txt — Notatnik
Plik Edycja Format Widok Pomoc
TAK
1 2 5
2 3 10
Lin 1, kol 1
100%
Windows (CRLF)
UTF-8
wyjscie3.txt — Notatnik
Plik Edycja Format Widok Pomoc
Źle zapisany plik wejscia.
Lin 1, kol 1
100%
Windows (CRLF)
ANSI
wyjscie2.txt — Notatnik
Plik Edycja Format Widok Pomoc
TAK
1 5 2
5 7 1
5 2 4
7 3 3
1 4 2
1 6 1
Lin 1, kol 1
100%
Windows (CRLF)
UTF-8
Wpisz tu wyszukiwane słowa
20:25
2021-01-21


Raport projektu
Plik | C:/Users/Lenovo/Desktop/Projket_JS/raport.html
Raport projektu Plan Metra
Liczba plików wejścia: 4
Data wykonania raportu: 2021-01-21 20:25:14.736437
Autor: Michał Osiewicz
Wejścia:
Wyjścia:
wejscie1.txt
3
5
10
wyjscie1.txt
TAK
1 2 5
2 3 10
wejscie2.txt
7
6 6 2 2 1
5 3 5 1 4
wyjscie2.txt
TAK
1 5 2
5 7 1
5 2 4
7 3 3
1 4 2
1 6 1
wejscie3.txt
wyjscie3.txt
Źle zapisany plik wejscia.
Wpisz tu wyszukiwane słowa
20:27
2021-01-21

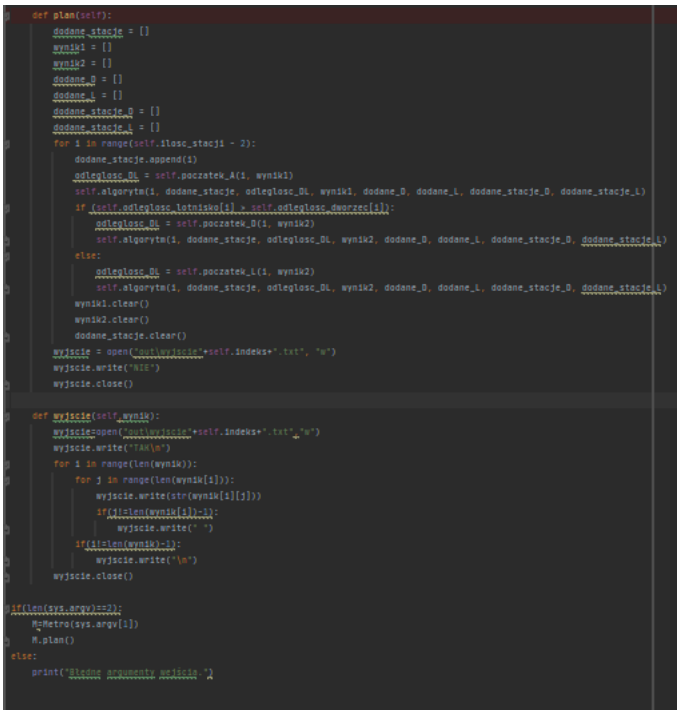

Backup2021-01-21
Plik
Narzędzia główne
Udostępnianie
Widok
Projket_JS > Backup2021-01-21
Przeszukaj: Backup2021...
Szybki dostęp
Pulpit
Pobrane
Dokumenty
Obrazy
in
out
Projekt_JS_Plan_met
Projket_JS
OneDrive
Ten komputer
Sieć
Nazwa
Data modyfikacji
Typ
Rozmiar
in 2021-01-21 20:28 Folder plików
out 2021-01-21 20:28 Folder plików
backup.bat 2021-01-19 16:30 Plik wsadowy Windo... 1 KB
menu.bat 2021-01-19 16:16 Plik wsadowy Windo... 1 KB
program.py 2021-01-17 18:26 Python File 12 KB
raport.html 2021-01-21 20:25 Chrome HTML Docu... 2 KB
raport.py 2021-01-17 18:23 Python File 2 KB
Elementy: 7
Wpisz tu wyszukiwane słowa
20:28
2021-01-21

Pełen kod aplikacji

Główny algorytm program.py

```
def poczatek_D(self,i, tab):
    tab.append([])
    tab.append([])
    tab[0].append(self.ilosc_stacji)
    tab[0].append(i)
    tab[0].append(self.odleglosc_lotnisko[i] + self.odleglosc_dworzec[i])
    tab[1].append(i)
    tab[1].append(i + 2)
    tab[1].append(self.odleglosc_dworzec[i])
    return self.odleglosc_lotnisko[i] - self.odleglosc_dworzec[i]

def algorytm(self,i, dodane, odl, tab, dodane_D, dodane_L, d_stacje_D, d_stacje_L):
    if (len(tab) == self.ilosc_stacji - 1):
        self.wyjscie(tab)
        exit()
    tab1 = []
    tab2 = []
    tab3 = []
    tab4 = []
    tab5 = []
    tablica_dodatkowa_D = []
    tablica_dodatkowa_L = []
    for x in range(len(dodane_D)):
        tablica_dodatkowa_D.append([])
        for j in range(len(tab)):
            tablica_dodatkowa_D[x].append(tab[j])
    for x in range(len(dodane_L)):
        tablica_dodatkowa_L.append([])
        for j in range(len(tab)):
            tablica_dodatkowa_L[x].append(tab[j])
    for x in range(len(tab)):
        tab1.append(tab[x])
        tab2.append(tab[x])
        tab3.append(tab[x])
        tab4.append(tab[x])
        tab5.append(tab[x])
    licznik = len(tab)
    for j in range(self.ilosc_stacji - 2):
        if (dodane.count(i) == 0):
            self.dodanie_do_A(self.odleglosc_dworzec[i], self.odleglosc_lotnisko[i],
                              self.odleglosc_dworzec[j], self.odleglosc_lotnisko[j], i + 2, j + 2, tab1)
            for x in range(len(dodane_D)):
                self.dodanie_do_A(self.odleglosc_dworzec[d_stacje_D[x] - 2], self.odleglosc_lotnisko[d_stacje_D[x] - 2],
                                  self.odleglosc_dworzec[j], self.odleglosc_lotnisko[j], d_stacje_D[x], j + 2,
                                  tablica_dodatkowa_D[x])
            for x in range(len(dodane_L)):
                self.dodanie_do_A(self.odleglosc_dworzec[d_stacje_L[x] - 2], self.odleglosc_lotnisko[d_stacje_L[x] - 2],
                                  self.odleglosc_dworzec[j], self.odleglosc_lotnisko[j], d_stacje_L[x], j + 2,
                                  tablica_dodatkowa_L[x])
            self.dodanie_do_D(odl, self.odleglosc_dworzec[j], self.odleglosc_lotnisko[j], j + 2, tab3)
            self.dodanie_do_L(odl, self.odleglosc_dworzec[j], self.odleglosc_lotnisko[j], j + 2, tab5)
            if (tab[0][2] == self.odleglosc_dworzec[j] or len(dodane_D) > 0 or len(dodane_L) > 0):
                self.dodanie_do_Lsr(self.odleglosc_dworzec[i], self.odleglosc_lotnisko[i],
                                    self.odleglosc_dworzec[j], self.odleglosc_lotnisko[j], j + 2, tab4, dodane_D, d_stacje_D)
                self.dodanie_do_Psr(self.odleglosc_dworzec[i], self.odleglosc_lotnisko[i],
                                    self.odleglosc_dworzec[j], self.odleglosc_lotnisko[j], j + 2, tab5, dodane_L, d_stacje_L)
            if (len(tab1) > licznik):
                dodane.append(j)
                self.algorytm(i, dodane, odl, tab1, dodane_D, dodane_L, d_stacje_D, d_stacje_L)
                tab1.pop(len(tab1) - 1)
                licznik = len(tab)
            if (len(tab2) > licznik):
                dodane.append(j)
                self.algorytm(i, dodane, odl, tab2, dodane_D, dodane_L, d_stacje_D, d_stacje_L)
                tab2.pop(len(tab2) - 1)
                licznik = len(tab)
```

```
            if (len(tab3) > licznik):
                dodane.append(j)
                self.algorytm(i, dodane, odl, tab3, dodane_D, dodane_L, d_stacje_D, d_stacje_L)
                tab3.pop(len(tab3) - 1)
                licznik = len(tab)
            if (len(tab4) > licznik):
                dodane.append(j)
                self.algorytm(i, dodane, odl, tab4, dodane_D, dodane_L, d_stacje_D, d_stacje_L)
                tab4.pop(len(tab4) - 1)
                dodane_D.pop(len(dodane_D) - 1)
                d_stacje_D.pop(len(d_stacje_D) - 1)
                licznik = len(tab)
            if (len(tab5) > licznik):
                dodane.append(j)
                self.algorytm(i, dodane, odl, tab5, dodane_D, dodane_L, d_stacje_D, d_stacje_L)
                tab5.pop(len(tab5) - 1)
                dodane_L.pop(len(dodane_L) - 1)
                d_stacje_L.pop(len(d_stacje_L) - 1)
                licznik = len(tab)
            for x in range(len(dodane_D)):
                if (len(tablica_dodatkowa_D[x]) > licznik):
                    dodane.append(j)
                    self.algorytm(i, dodane, odl, tablica_dodatkowa_D[x], dodane_D, dodane_L, d_stacje_D, d_stacje_L)
                    tablica_dodatkowa_D[x].pop(len(tablica_dodatkowa_D[x]) - 1)
                    licznik = len(tab)
            for x in range(len(dodane_L)):
                if (len(tablica_dodatkowa_L[x]) > licznik):
                    dodane.append(j)
                    self.algorytm(i, dodane, odl, tablica_dodatkowa_L[x], dodane_D, dodane_L, d_stacje_D, d_stacje_L)
                    tablica_dodatkowa_L[x].pop(len(tablica_dodatkowa_L[x]) - 1)
                    licznik = len(tab)
```

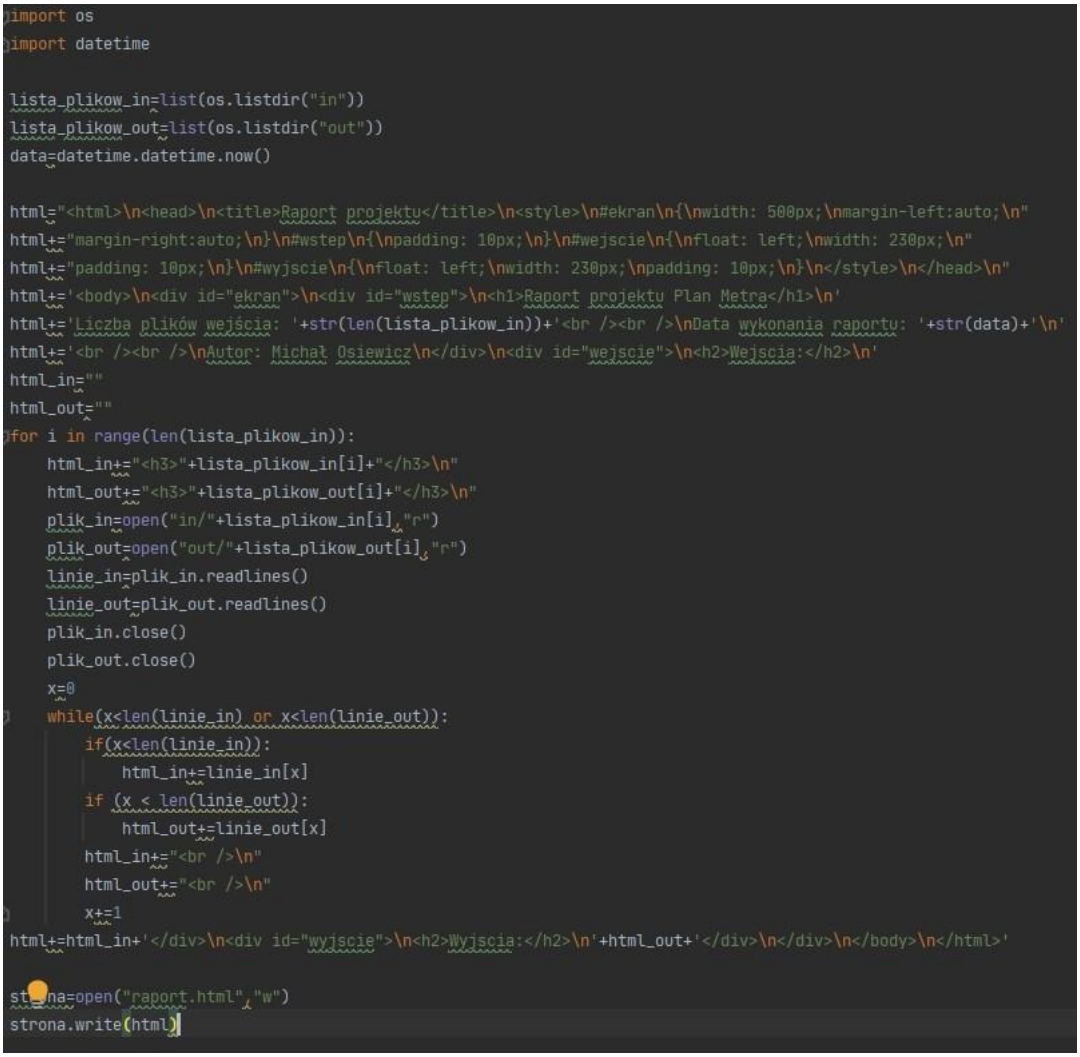


```
    def plan(self):
        dodane_stacje = []
        wynik1 = []
        wynik2 = []
        dodane_D = []
        dodane_L = []
        dodane_stacje_D = []
        dodane_stacje_L = []
        for i in range(self.ilosc_stacji - 2):
            dodane_stacje.append(i)
            odleglosc_DL = self.poczatek_A(i, wynik1)
            self.algorytm(i, dodane_stacje, odleglosc_DL, wynik1, dodane_D, dodane_L, dodane_stacje_D, dodane_stacje_L)
            if (self.odleglosc_lotnisko[i] > self.odleglosc_dworzec[i]):
                odleglosc_DL = self.poczatek_D(i, wynik2)
                self.algorytm(i, dodane_stacje, odleglosc_DL, wynik2, dodane_D, dodane_L, dodane_stacje_D, dodane_stacje_L)
            else:
                odleglosc_DL = self.poczatek_L(i, wynik2)
                self.algorytm(i, dodane_stacje, odleglosc_DL, wynik2, dodane_D, dodane_L, dodane_stacje_D, dodane_stacje_L)
            wynik1.clear()
            wynik2.clear()
            dodane_stacje.clear()
        wyjscie = open("out\wyjscie"+self.indeks+".txt", "w")
        wyjscie.write("NIE")
        wyjscie.close()

    def wyjscie(self,wynik):
        wyjscie=open("out\wyjscie"+self.indeks+".txt","w")
        wyjscie.write("TAK\n")
        for i in range(len(wynik)):
            for j in range(len(wynik[i])):
                wyjscie.write(str(wynik[i][j]))
                if(j!=len(wynik[i])-1):
                    wyjscie.write(" ")
            if(i!=len(wynik)-1):
                wyjscie.write("\n")
        wyjscie.close()

if(len(sys.argv)==2):
    M=Metro(sys.argv[1])
    M.plan()
else:
    print("Bledne argumenty wejscia.")
```

Menu.bat

```
@ echo off
:MENU
echo -----------MENU-----------
echo 1. Uruchom projekt
echo 2. Informacje o projekcie
echo 3. Backup
echo 4. Wyjscie

set /p wybor=Wybierz opcje:

if %wybor%==1 (
goto START
)
if %wybor%==2 (
goto INFO
)
if %wybor%==3 (
goto BACK
)
if %wybor%==4 (
goto WY
) else (
echo Niepoprawna wartosc.
pause
cls
goto MENU
)
:START
for %%X in (in\wejscie*.txt) do (call program.py %%X)
call raport.py
pause
cls
goto MENU
:INFO
echo Projekt Plan Metra. Autor: Michal Osiewicz
pause
cls
goto MENU
:BACK
call backup.bat
pause
cls
goto MENU
:WY
exit
```

Raport.py



```
import os
import datetime

lista_plikow_in=list(os.listdir("in"))
lista_plikow_out=list(os.listdir("out"))
data=datetime.datetime.now()

html="<html>\n<head>\n<title>Raport projektu</title>\n<style>\n#ekran\n{\nwidth: 500px;\nmargin-left:auto;\n"
html+="margin-right:auto;\n}\n#wstep\n{\npadding: 10px;\n}\n#wejscie\n{\nfloat: left;\nwidth: 230px;\n"
html+="padding: 10px;\n}\n#wyjscie\n{\nfloat: left;\nwidth: 230px;\npadding: 10px;\n}\n</style>\n</head>\n"
html+='<body>\n<div id="ekran">\n<div id="wstep">\n<h1>Raport projektu Plan Metra</h1>\n'
html+='Liczba plików wejścia: '+str(len(lista_plikow_in))+'<br /><br />\nData wykonania raportu: '+str(data)+'\n'
html+='<br /><br />\nAutor: Michał Osiewicz\n</div>\n<div id="wejscie">\n<h2>Wejscia:</h2>\n'
html_in=""
html_out=""
for i in range(len(lista_plikow_in)):
    html_in+="<h3>"+lista_plikow_in[i]+"</h3>\n"
    html_out+="<h3>"+lista_plikow_out[i]+"</h3>\n"
    plik_in=open("in/"+lista_plikow_in[i],"r")
    plik_out=open("out/"+lista_plikow_out[i],"r")
    linie_in=plik_in.readlines()
    linie_out=plik_out.readlines()
    plik_in.close()
    plik_out.close()
    x=0
    while(x<len(linie_in) or x<len(linie_out)):
        if(x<len(linie_in)):
            html_in+=linie_in[x]
        if (x < len(linie_out)):
            html_out+=linie_out[x]
        html_in+="<br />\n"
        html_out+="<br />\n"
        x+=1
html+=html_in+'</div>\n<div id="wyjscie">\n<h2>Wyjscia:</h2>\n'+html_out+'</div>\n</div>\n</body>\n</html>'

strona=open("raport.html","w")
strona.write(html)
```

Backup.bat

```
set data=%DATE%
mkdir Backup%data%
copy *.* Backup%data%
xcopy in Backup%data%\in\
xcopy out Backup%data%\out\
```