Dokumentacja projektu

Projekt zaliczeniowy z przedmiotu Języki Skryptowe

Maciej Jurczyk

Wydział Matematyki Stosowanej
Informatyka 2020/2021
studia stacjonarne
semestr III
grupa 1A

21 stycznia 2021

Spis treści

1 Opis problemu

Zadaniem programu jest znalezienie odpowiedniego rozmieszczenia ulepszonych wież w państwie gracza, tak aby całe jego państwo było chronione przed nieprzyjacielem. Możemy jednak postawić tylko k ulepszonych wież. Do dyspozycji mamy dwa rodzaje wieży. Zwykła chroni miasto w którym się znajduje oraz miasta sąsiednie bezpośrednio połączone z nim drogą. Zaś ulepszona wieża ochrania dodatkowo wszystkie miasta połączone drogą z sąsiadami.

Zadaniem programu jest odnalezienie i wypisanie dwóch wierszy opisujących rozmieszczenie ulepszonych wież w państwie gracza. Pierwszy wiersz zawiera liczbę całkowitą oznaczającą liczbę ulepszonych wież, które powinien zbudować gracz. Drugi wiersz zawiera opis rozmieszczenia tych wież, oznaczających numery miast, w których należy zbudować ulepszone wieże strażnicze. W przypadku, gdy istnieje więcej niż jedno rozwiązanie, wypisuje jedną z możliwości.

2 Problem matematyczny

2.1 Opis modelu matematycznego[1]

W rozwiązaniu przydatny jest graf nieskierowany reprezentujący państwo gracza. Wierzchołki grafu oznaczają miasta, a krawędzie bezpośrednie drogi między nimi. Przekładając odpowiednie pojęcia na teorię grafów otrzymamy poniższą treść.

Niech G=(V,E) będzie grafem o zbiorze wierzchołków V i zbiorze nieskierowanych krawędzi E. Powiemy, że dwa wierzchołki u,v $\in$ V są w odległości nie większej niż r, jeśli w grafie G istnieje ścieżka łącząca u i v, zawierająca co najwyżej r krawędzi. Dla liczby całkowitej dodatniej r, zbiór wierzchołków X w grafie G = (V,E) nazwiemy r-dominującym, jeśli dla każdego wierzchołka ze zbioru V istnieje wierzchołek ze zbioru X w odległości nie większej niż r. Zbiór 1-dominujący będziemy nazywać zazwyczaj po prostu dominującym.

Zauważmy, że X = V jest zbiorem dominującym w grafie G. Ciekawym problemem jest jednak znajdowanie jak najmniejszego zbioru dominującego lub r-dominującego. W szczególności, problem postawienia k (nieulepszonych) wież strażniczych w państwie gracza to, używając właśnie wprowadzonej definicji, problem znalezienia w danym grafie zbioru dominującego o co najwyżej k wierzchołkach. Stwierdzenie że "Wystarczy k wież by ochronić całe państwo." możemy więc przetłumaczyć na język teorii grafów jako „W tym grafie istnieje zbiór dominujący o k wierzchołkach!".

Co zaś zmieniają ulepszone wieże? Zauważmy, że problem postawienia k ulepszonych wież to tak naprawdę problem znalezienia zbioru 2-dominującego o co najwyżej k wierzchołkach. Zadanie, które ma przed sobą gracz, brzmi więc:

Wiedząc, że w grafie G istnieje zbiór dominujący wielkości k,
znajdź w G zbiór 2-dominujący wielkości k.

2.2 Przykładowe rozwiązanie

Aby rozwiązać problem spróbujmy postąpić zachłannie: dopóki istnieje choć jedno niepilnowane miasto, postawmy w nim ulepszoną wieże. A w języku teorii grafów: konstruujemy zbiór wierzchołków X, zaczynając od X = $\emptyset$, i, dopóki X nie jest zbiorem 2-dominującym, bierzemy dowolny wierzchołek v w odległości większej niż 2 od wszystkich wierzchołków z X i dodajemy go do zbioru X. Oczywiście, w ten sposób otrzymamy zbiór 2-dominujący. Nie jest jednak jasne, że nie postawimy w ten sposób więcej niż k wież. By to pokazać, rozważmy zbiór dominujący Z wielkości co najwyżej k, zgodnie ze stwierdzeniem o k wieżach. Przeanalizujemy jeden krok naszego algorytmu, w którym do zbioru X dodajemy wierzchołek v (czyli stawiamy ulepszona wieże w v). Zgodnie z definicja zbioru dominującego, istnieje wierzchołek z$\in$Z w odległości co najwyżej 1 od v. Poczyńmy kluczową obserwacje: każdy wierzchołek, pilnowany przez (nieulepszoną) wieże postawioną w wierzchołku z, jest również pilnowany przez ulepszoną wieże postawioną w v. Innymi słowy, w tym kroku zbiór wierzchołków pilnowanych przez postawione dotychczas ulepszone wieże „połyka" wszystkie wierzchołki pilnowane przez (nieulepszoną) wieże w z. W Związku z tym w każdym kroku algorytmu mamy do czynienia z innym wierzchołkiem z. A ponieważ |Z|$\leqslant$k, więc wykonamy nie więcej niż k kroków, czyli postawimy co najwyżej k ulepszonych wież.

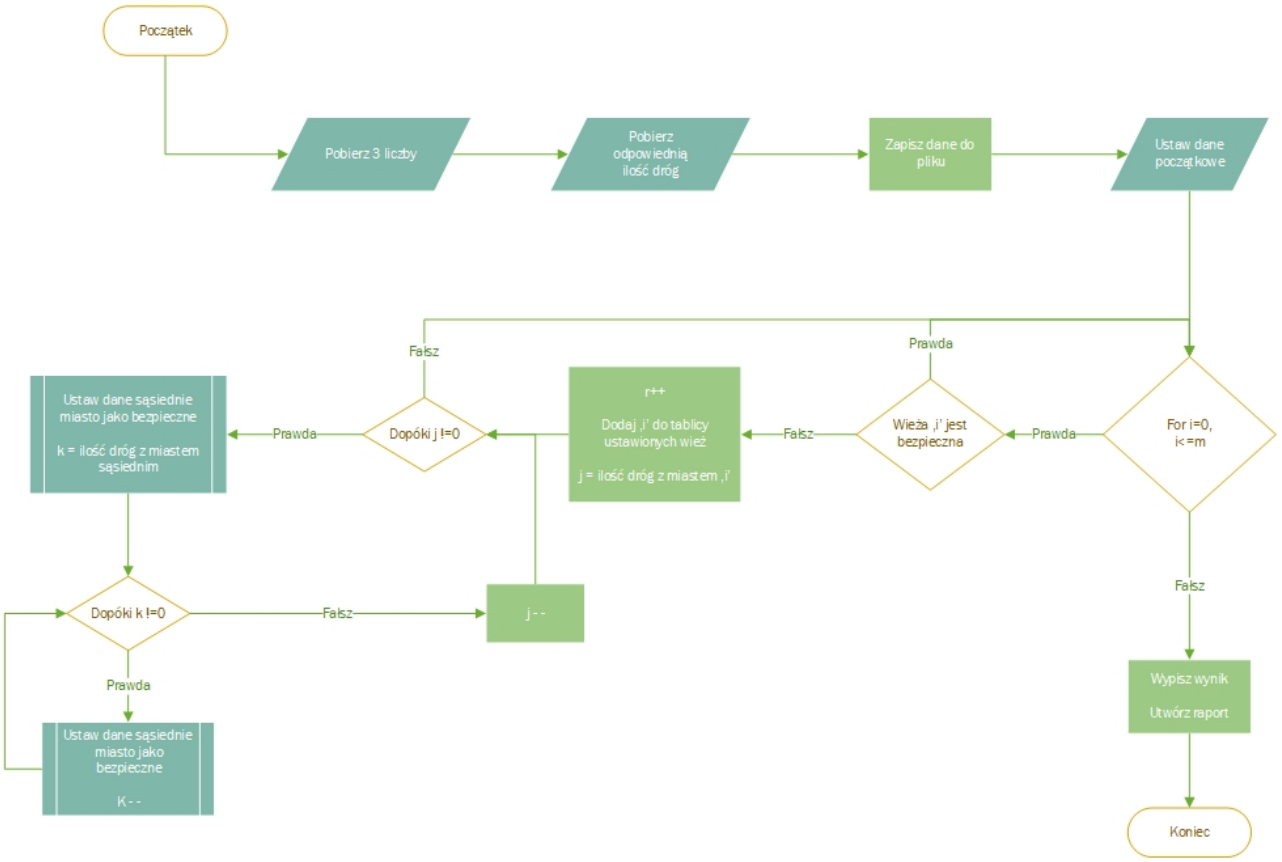
[1] treść opisu modelu matematycznego pochodzi z oficjalnej dokumentacji z rozwiązaniami zadań konkursowych, źródło oryginalne: `https://oi.edu.pl/static/attachment/20140306/oi20_no_ref.pdf`

3 Algorytm

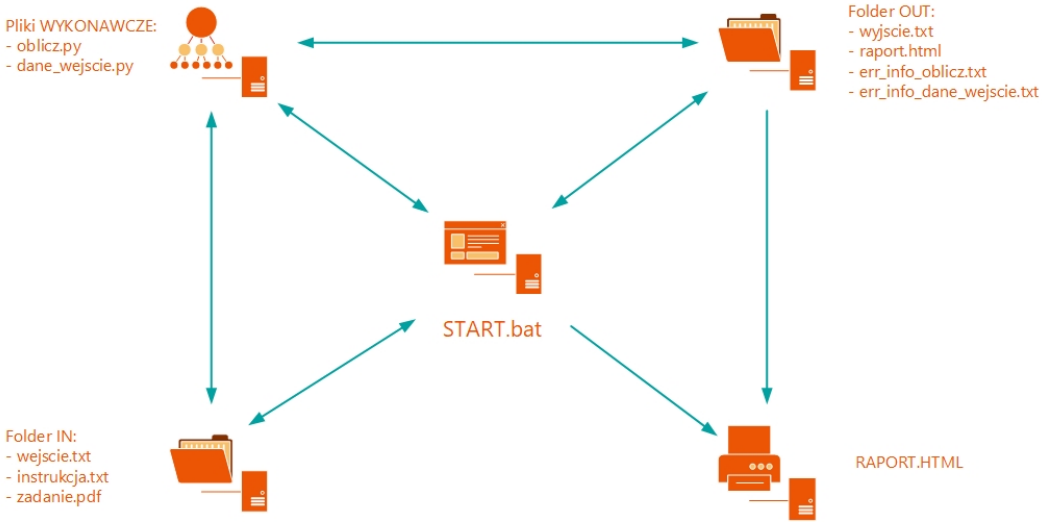
3.1 Algorytm słowny

1. Pobierz 3 liczby informacji ogólnych
2. Pobierz zadaną ilość wartości dróg w państwie
3. Zapisz pobrane dane do pliku "/in/wejscie.txt"
4. Wczytaj dane początkowe: tablice z drogami, n oraz m,
5. For i=0, i⩽m rób:
   (a) Jeśli i-ta wieża nie jest jeszcze bezpieczna rób:
      i. Zwiększ licznik wież r++ oraz dodaj 'i' do tablicy ustawionych wież
      ii. j = ilość dróg z miastem 'i'
      iii. Dopóki j != 0, rób:
         A. Ustaw sąsiednie miasto jako zabezpieczone
         B. k = ilość dróg z miastem sąsiednim
         C. Dopóki k != 0, rób:
            - Ustaw sąsiednie miasto sąsiada jako zabezpieczone
            - Zmniejsz k o 1
         D. Zmniejsz j o 1
6. Wypisz wynik na ekran oraz zapisz do pliku "/out/wyjscie.txt"
7. Stwórz raport na podstawie danych wejściowych i wyjściowych

3.2 Schemat blokowy

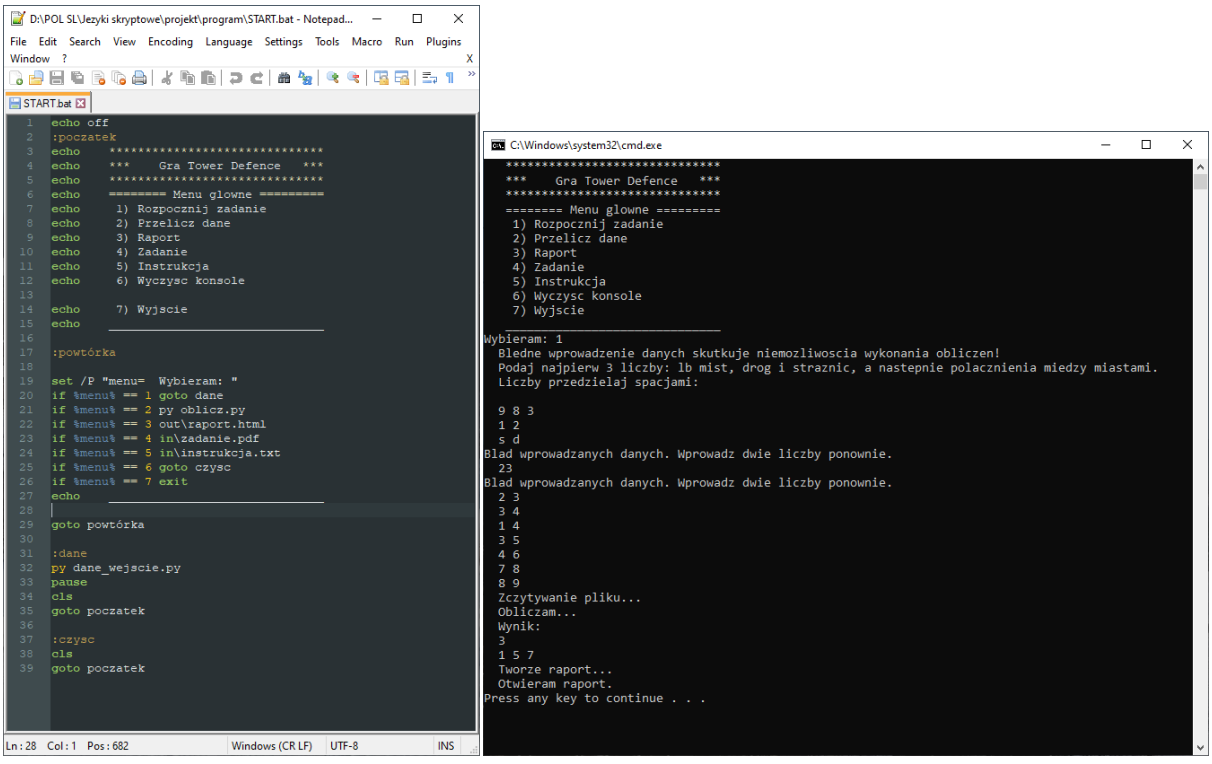


3.3 Schemat działania programu



4 Implementacja

4.1 START.bat



D:\POL SL\Jezyki skryptowe\projekt\program\START.bat - Notepad...

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

START.bat

```
echo off
:poczatek
echo     ******************************
echo     ***    Gra Tower Defence   ***
echo     ******************************
echo     ======== Menu glowne =========
echo      1) Rozpocznij zadanie
echo      2) Przelicz dane
echo      3) Raport
echo      4) Zadanie
echo      5) Instrukcja
echo      6) Wyczysc konsole

echo      7) Wyjscie
echo     _____________________________

:powtórka

set /P "menu=  Wybieram: "
if %menu% == 1 goto dane
if %menu% == 2 py oblicz.py
if %menu% == 3 out\raport.html
if %menu% == 4 in\zadanie.pdf
if %menu% == 5 in\instrukcja.txt
if %menu% == 6 goto czysc
if %menu% == 7 exit
echo     _____________________________

goto powtórka

:dane
py dane_wejscie.py
pause
cls
goto poczatek

:czysc
cls
goto poczatek
```

Ln : 28 Col : 1 Pos : 682 Windows (CR LF) UTF-8 INS

C:\Windows\system32\cmd.exe

```
  ******************************
  ***    Gra Tower Defence   ***
  ******************************
  ======== Menu glowne =========
   1) Rozpocznij zadanie
   2) Przelicz dane
   3) Raport
   4) Zadanie
   5) Instrukcja
   6) Wyczysc konsole
   7) Wyjscie
  _____________________________
Wybieram: 1
  Bledne wprowadzenie danych skutkuje niemozliwoscia wykonania obliczen!
  Podaj najpierw 3 liczby: lb mist, drog i straznic, a nastepnie polacznienia miedzy miastami.
  Liczby przedzielaj spacjami:

  9 8 3
  1 2
  s d
Blad wprowadzanych danych. Wprowadz dwie liczby ponownie.
  23
Blad wprowadzanych danych. Wprowadz dwie liczby ponownie.
  2 3
  3 4
  1 4
  3 5
  4 6
  7 8
  8 9
  Zczytywanie pliku...
  Obliczam...
  Wynik:
  3
  1 5 7
  Tworze raport...
  Otwieram raport.
Press any key to continue . . .
```

4.2 Wprowadzanie danych

```
import ...

file = open("in/dane.txt", "w")
err_log = open("out/err_info_dane_wejscie.txt", "w")

print("  Bledne wprowadzenie danych skutkuje niemozliwoscia wykonania obliczen!")
print("  Podaj najpierw 3 liczby: lb miast, drog i straznic, a nastepnie polaczenia miedzy miastami.\n"
      "  Liczby przedzielaj spacjami:")
print()
try:
    while True:
        nmk = input("  ")
        txt = nmk.replace(" ", "")
        tab = nmk.split()
        m = tab[1]
        if txt.isnumeric() and len(txt) >= 3 and len(tab) == 3:  # czy podano dokładnie 3 liczby, z dowolną ilością spacji
            break
        print("Blad wprowadzanych danych. Wprowadz trzy liczby ponownie.")
        err_log.write(nmk + " -> Zly typ wprowadzanych danych lub ich niewlasciwa ilosc.\n")
    file.write(nmk + "\n")
    for i in range(int(m)):
        while True:
            line = input("  ")
            tkst = line.replace(" ", "")
            tabl = line.split()
            if tkst.isnumeric() and len(tkst) >= 2 and len(tabl) == 2:  # czy podano dokładnie 2 liczby, z dowolną ilością spacji
                file.write(line + "\n")
                break
            print("Blad wprowadzanych danych. Wprowadz dwie liczby ponownie.")
            err_log.write(line + " -> Zly typ wprowadzanych danych lub ich niewlasciwa ilosc.\n")
except Exception as err:
    print("Blad krytyczny. Program przerwany. Po wiecej informacji zerknij do err_info_dane_wejscie.txt.")
    err_log.write(err + "\n")

f = open("out/err_info_oblicz.txt", "w")
f.close()
file.close()
err_log.close()
os.system('py oblicz.py')
```

```
C:\Windows\system32\cmd.exe
  ********************************
  ***    Gra Tower Defence     ***
  ********************************
  -------- Menu glowne ---------
   1) Rozpocznij zadanie
   2) Przelicz dane
   3) Raport
   4) Zadanie
   5) Instrukcja
   6) Wyczysc konsole
   7) Wyjscie
  ------------------------------
Wybieram: 1
  Bledne wprowadzenie danych skutkuje niemozliwoscia wykonania obliczen!
  Podaj najpierw 3 liczby: lb mist, drog i straznic, a nastepnie polaczenia miedzy miastami.
  Liczby przedzielaj spacjami:

  9 8 3
  1 2
  s d
Blad wprowadzanych danych. Wprowadz dwie liczby ponownie.
  23
Blad wprowadzanych danych. Wprowadz dwie liczby ponownie.
  2 3
  3 4
  1 4
  3 5
  4 6
  7 8
  8 9
  Zczytywanie pliku...
  Obliczam...
  Wynik:
  3
  1 5 7
  Tworze raport...
  Otwieram raport.
Press any key to continue . . .
```

4.3 Obliczanie danych i tworzenie raportu

```
#!/usr/bin/python3

# autor: Maciej Jurczyk
import pathlib
import time
import sys
import os

err_log = open("out/err_info_oblicz.txt", "a")
wyjscie = open("out/wyjscie.txt", "w")
try:
    dane = open("in/dane.txt", "r")
except FileExistsError as err:
    print("Plik dane.txt nie istnieje.")
    err_log.write(str(err) + "\n")
max_n = 50000
max_m = 1000000
towers = []
test = True
r = 0

try:
    print("  Zczytywanie pliku...")
    time.sleep(0.75)
    text = (dane.readline()).split()
    n = int(text[0])  # liczba miast
    m = int(text[1])  # liczba dróg w państwie
    k = int(text[2])  # maksymalna liczba strażnic
    c = 0
    edges = [0] * (max_m+1)
    already_safed = [0] * max_m
    neighbour = [0] * max_m
    neighbour2 = [0] * max_m

    # zczytywanie danych i tworzenie tablic
```

```
    # zczytywanie danych i tworzenie tablic
    for i in range(m):
        tekst = dane.readline()
        if tekst == "":
            print("\nBlad danych! Nie mozna wykonac obliczen.")
            wyjscie.write("\nBlad danych! Nie mozna wykonac obliczen.")
            err_log.write("Blad danych! Nie mozna wykonac obliczen.\n")
            sys.exit()
        text = tekst.split()
        c += 1
        neighbour[c] = int(text[1])
        neighbour2[c] = edges[int(text[0])]
        edges[int(text[0])] = c
        c += 1
        neighbour[c] = int(text[0])
        neighbour2[c] = edges[int(text[1])]
        edges[int(text[1])] = c

    # główny algorytm rozmieszczania strażnic
    print("  Obliczam...")
    time.sleep(0.5)
    for i in range(1, n+1):
        if not already_safed[i]:
            r += 1
            towers.append(i)
            j = edges[i]
            while not j == 0:
                already_safed[neighbour[j]] = True
                k = edges[neighbour[j]]
                while not k == 0:
                    already_safed[neighbour[k]] = True
                    k = neighbour2[k]
                j = neighbour2[j]

    print("  Wynik:")
    print("  " + str(r), end="\n  ")
    wyjscie.write(str(r) + "\n")

```

```
    print("  Wynik:")
    print("  " + str(r), end="\n  ")
    wyjscie.write(str(r) + "\n")

    for i in range(r):
        wyjscie.write(str(towers[i]) + " ")
        print(str(towers[i]), end=" ")
    print()

except Exception as err:
    print("Blad krytyczny. Program przerwany. Po wiecej informacji zerknij do err_info_oblicz.txt.")
    err_log.write(str(err) + "\n")
err_log.close()
wyjscie.close()
dane.close()
```

```
# ================================== RAPORT ========================================
print("  Tworze raport...")
time.sleep(0.75)

raport = open('out/raport.html', 'w')
wyjscie = open("out/wyjscie.txt", "r")
err_log = open('out/err_info_oblicz.txt', 'r')
dane = open("in/dane.txt", "r")
try:
    err_log_wejscie = open('out/err_info_dane_wejscie.txt', 'r')
except FileExistsError as err:
    print("Plik err_info_dane_wejscie.txt nie istnieje.")
    err_log.write(str(err) + "\n")
    raport.close()
    wyjscie.close()
    err_log.close()
    sys.exit()

raport.write("<!DOCTYPE html>\n<html>\n<head>\n<title>\nRaport dzialania programu\n</title>\n</head>\n"
             "<body>\n<h1 style=\"color: SlateBlue;\">Raport</h1>\n")
raport.write("<h3>Wyniki:</h3>\n")
a = "Dla danych wejsciowych:".ljust(30) + "poprawnym wynikiem jest:\n\n"
raport.write("<pre>" + a)
i = 0
for line in dane:
    i += 1
    a = line.ljust(30)
    if i == 1 or i == 2:
        a += wyjscie.readline()
    a = a.replace("\n", "")
    raport.write(a+"\n")
raport.write("\n\n</pre>")

raport.write("<h3>Bledy napotkane przy pracy, od ostatniego pelnego uruchomienia programu :</h3>\n<p>\n")
for line in err_log_wejscie:
    raport.write(line + "\n<br>")
raport.write("</p>\n<p>\n")
for line in err_log:
    raport.write(line + "\n<br> ")
```

Raport

Wyniki:

```
Dla danych wejsciowych:       poprawnym wynikiem jest:

9 8 3                         3
1 2                           1 5 7
2 3
3 4
1 4
3 5
4 6
7 8
8 9
```

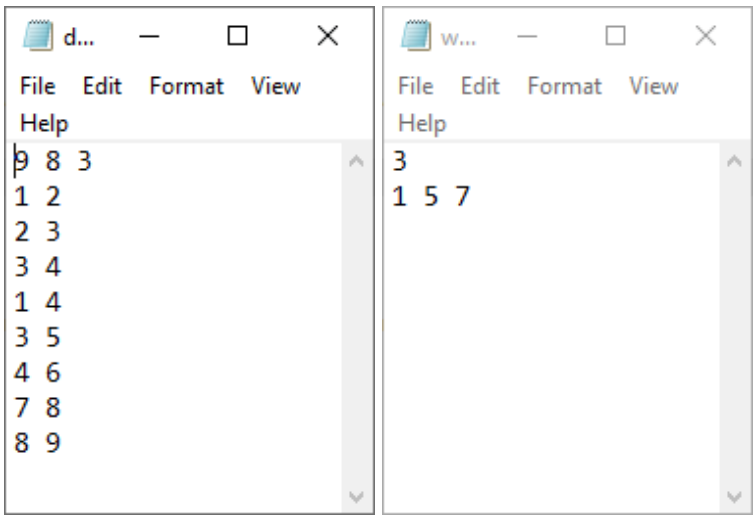
```
    raport.write("</body>\n</html>")
    raport.close()
    dane.close()
    wyjscie.close()

    print("  Otwieram raport.")
    time.sleep(0.5)
    os.system("out\\raport.html")
```

Bledy napotkane przy pracy, od ostatniego pelnego uruchomienia programu :

s d -> Zly typ wprowadzanych danych lub ich niewlasciwa ilosc.
23 -> Zly typ wprowadzanych danych lub ich niewlasciwa ilosc.

4.4 Przykład



5 Podsumowanie

5.1 Zrobione

W ramach projektu zaimplementowałem algorytm. Wzbogaciłem go o dodatkowe zabezpieczenia przy wprowadzaniu danych, tak aby zawsze mieć pewność, że wprowadzone dane nie spowodują żadnych błędów. Ponadto postarałem się zabezpieczyć prawie każdą możliwość zmiany danych wejściowych, które użytkownik mógłby zmienić ręcznie w plikach. Jeśli jakieś dane by się nie zgadzały to program powinien wypisać odpowiedni komunikat z czym jest błąd. Dodatkowo wiele z błędów przed którymi uchroniłem program jest zapisywanych do pliku i lista błędów napotkanych od ostatniego pełnego uruchomienia programu jest wypisywana w raporcie.

Program został przetestowany na innych systemach operacyjnych. Na systemie Windows 8.1 działa on poprawnie po zainstalowaniu dostępnej możliwie najnowszej wersji python. Dla wersji 7, miałem problem z zainstalowaniem pythona. Z racji przystosowania całego programu dla środowiska Windows w systemie Linux aplikacja nie odpala się.

5.2 Propozycje rozbudowy

W zaprezentowanym projekcie można by udoskonalić i rozbudować system zabezpieczeń po przez zwiększenie ilości przechwytywanych błędów oraz listy błędów. Ponadto można by dodać opcję obliczeniową wyznaczającą minimalną ilość wież potrzebnych do zabezpieczenia całego państwa. Ciekawym pomysłem wydaje się również stworzenie gry logicznej na podstawie zagadnienia poruszanego w projekcie. Użytkownik własnoręcznie musiałby rozwiązać przedstawiony wyżej problem dla danych wylosowanych przez system. Rozwiązanie musiałoby być minimalnym dla danej sytuacji.