Języki skryptowe

Projekt – *Rozkład Fibonacciego*

Karol Zając

Politechnika Śląska

Wydział Matematyki Stosowanej

Informatyka stopień I, rok II, semestr III

1. Temat projektu

Zadanie „Rozkład Fibonacciego” z XIX Olimpiady Informatycznej:
https://szkopul.edu.pl/problemset/problem/w1QbhPufazp-sH6X-u4pTnNu/site/?key=statement

Program ma na celu dla podanej liczby k znaleźć minimalną liczbę liczb Fibonacciego potrzebną do zapisania liczby k jako ich sumy lub różnicy.

2. Model matematyczny

Ciąg Fibonacciego dany jest wzorem:

$$F_0 = 0, \qquad F_1 = 0, \qquad F_n = F_{n-1} + F_{n-2}, \qquad n \geq 2$$

Sposób znalezienia dla podanej liczby k minimalnej liczby liczb Fibonacciego potrzebnej do zapisania liczby k jako ich sumy lub różnicy:

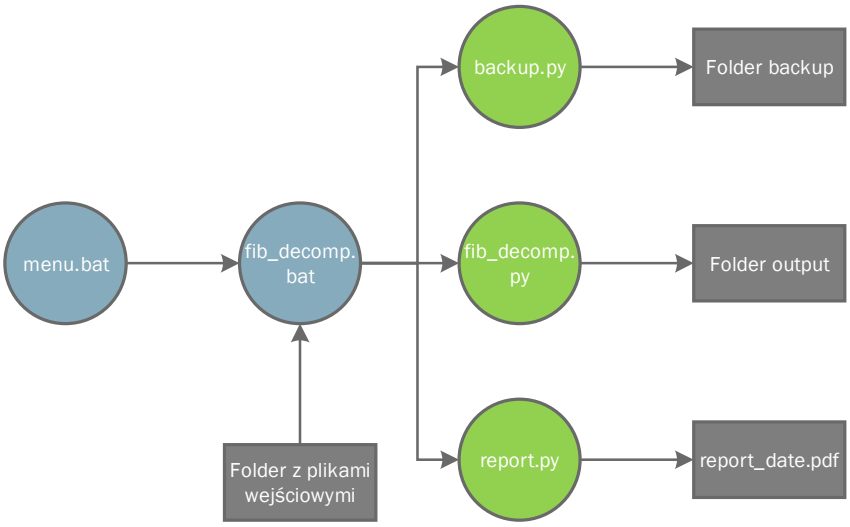
1. Dopóki liczba k jest różna od 0:
   1.1. Znaleźć liczbę Fibonacciego F_n najbliższą $|k|$.
   1.2. Jeżeli k jest dodatnie, obliczyć nowe $k = k - F_n$, jeżeli k jest ujemne, $k = k + F_n$.
2. Wynikiem jest liczba znalezionych liczb n Fibonacciego.

Przykład:

Znaleźć minimalną liczbę liczb Fibonacciego potrzebną do zapisania liczby $k = 85$ jako ich sumy lub różnicy.

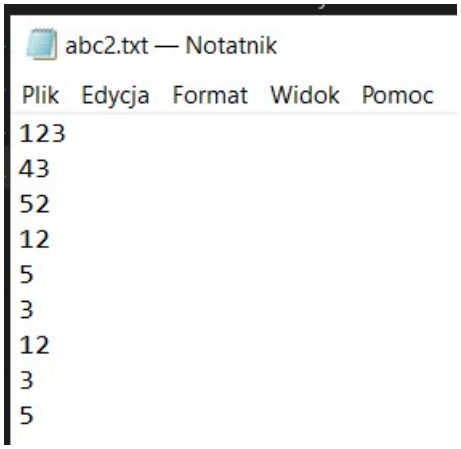
1. Najbliższą liczbie 85 liczbą Fibonacciego jest $F_{11} = 89$. Podstawić $k = 85 - 89 = -4$.
2. Najbliższą liczbie $|-4|$=4 liczbą Fibonacciego jest $F_4 = 3$. Podstawić $k = -4 + 3 = -1$.
3. Najbliższą liczbie $|-1|$=1 liczbą Fibonacciego jest $F_1 = 1$. Podstawić $k = -1 + 1 = 0$.
4. Liczbę 85 można przedstawić jako $89 - 3 - 1$, zatem liczba szukanych liczb Fibonacciego to 3.

3. Algorytm

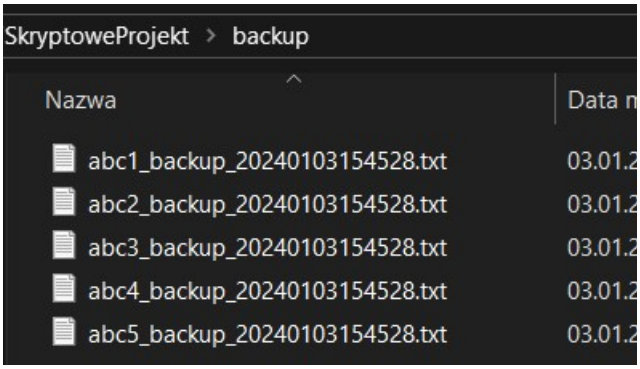


Rysunek 1 - schemat obrazujący działanie serwera

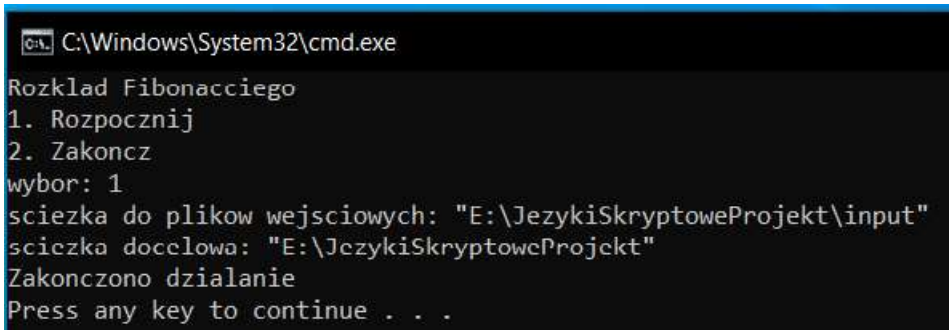
Skrypt menu.bat wywołuje skrypt fib_decomp.bat (skrypt menu.bat ma na celu ułatwić obsługę, możliwe jest wywołanie fib_decomp.bat z linii poleceń podając odpowiednie argumenty). Skrypt fib_decomp.bat przyjmuje pliki tekstowe z podanego folderu i wywołuje skrypt backup.py, który w folderze backup zapisuje kopie plików wejściowych. Następnie wywołuje skrypt fib_decomp.py, który na podstawie danych z plików wejściowych rozwiązuje zadania związane z rozkładem Fibonacciego i ich wyniki zapisuje w plikach tekstowych w folderze output. Następnie skrypt report.py sporządza raport w formie pliku pdf report_date.pdf.



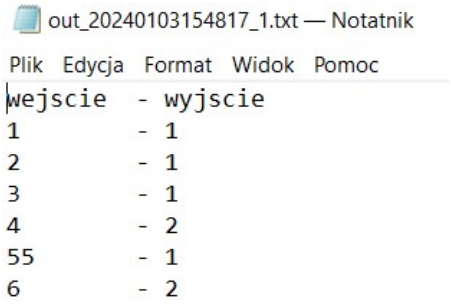
Zrzut ekranu 1 - przykładowy plik wejściowy



Zrzut ekranu 2 - kopie plików wykonane przez backup.py



Zrzut ekranu 3 - działanie skryptu menu.bat



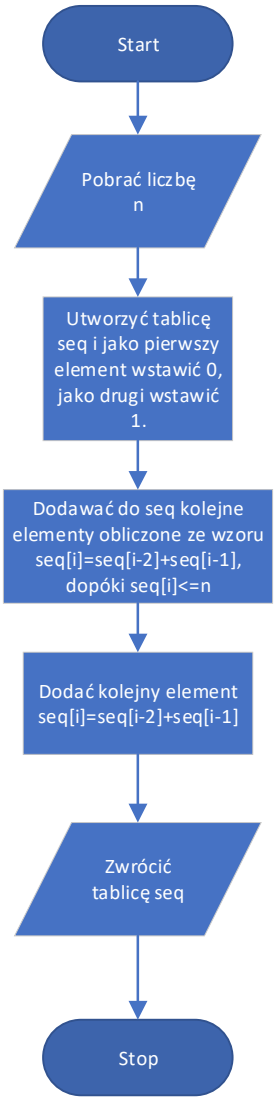
Zrzut ekranu 4 - przykładowy plik wyjściowy

report_20240103154817

nr.	wejscie	wyjscie	czas wykonania
1.	E:\JezykiSkryptoweProjekt\input\abc1.txt	E:\JezykiSkryptoweProjekt\output\out_20240103154817_1.txt	0.0022022079693603516
2.	E:\JezykiSkryptoweProjekt\input\abc2.txt	E:\JezykiSkryptoweProjekt\output\out_20240103154817_2.txt	0.0029981136322021484
3.	E:\JezykiSkryptoweProjekt\input\abc3.txt	E:\JezykiSkryptoweProjekt\output\out_20240103154817_3.txt	0.0019986629486083984
4.	E:\JezykiSkryptoweProjekt\input\abc4.txt	E:\JezykiSkryptoweProjekt\output\out_20240103154817_4.txt	0.0019991397857666016
5.	E:\JezykiSkryptoweProjekt\input\abc5.txt	E:\JezykiSkryptoweProjekt\output\out_20240103154817_5.txt	0.0019986629486083984

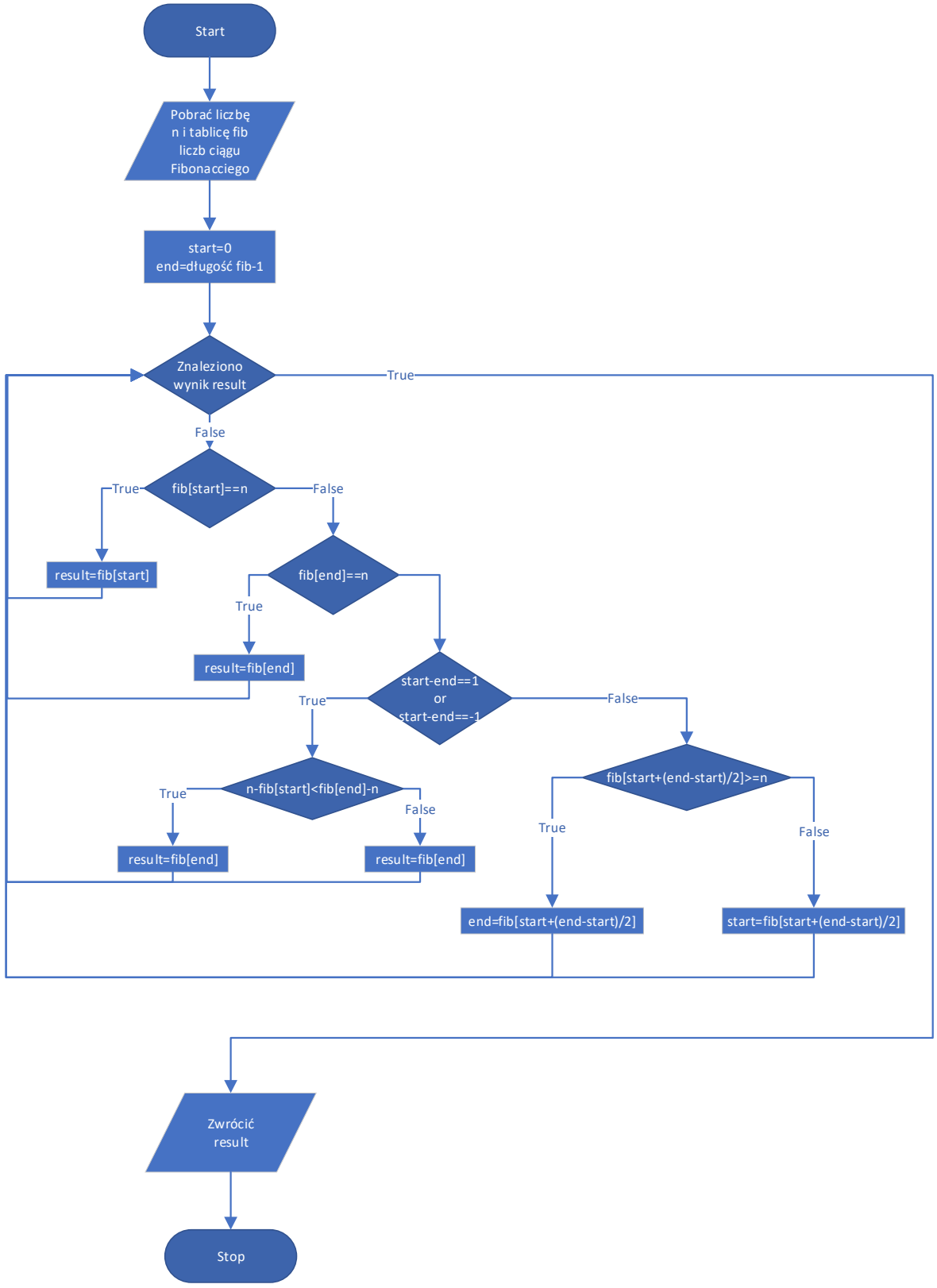
Zrzut ekranu 5 - przykładowy raport stworzony przez report.py

Schematy blokowe funkcji ze skryptu fib_decomp.py wykorzystanych do rozwiązania zadania rozkładu Fibonacciego:



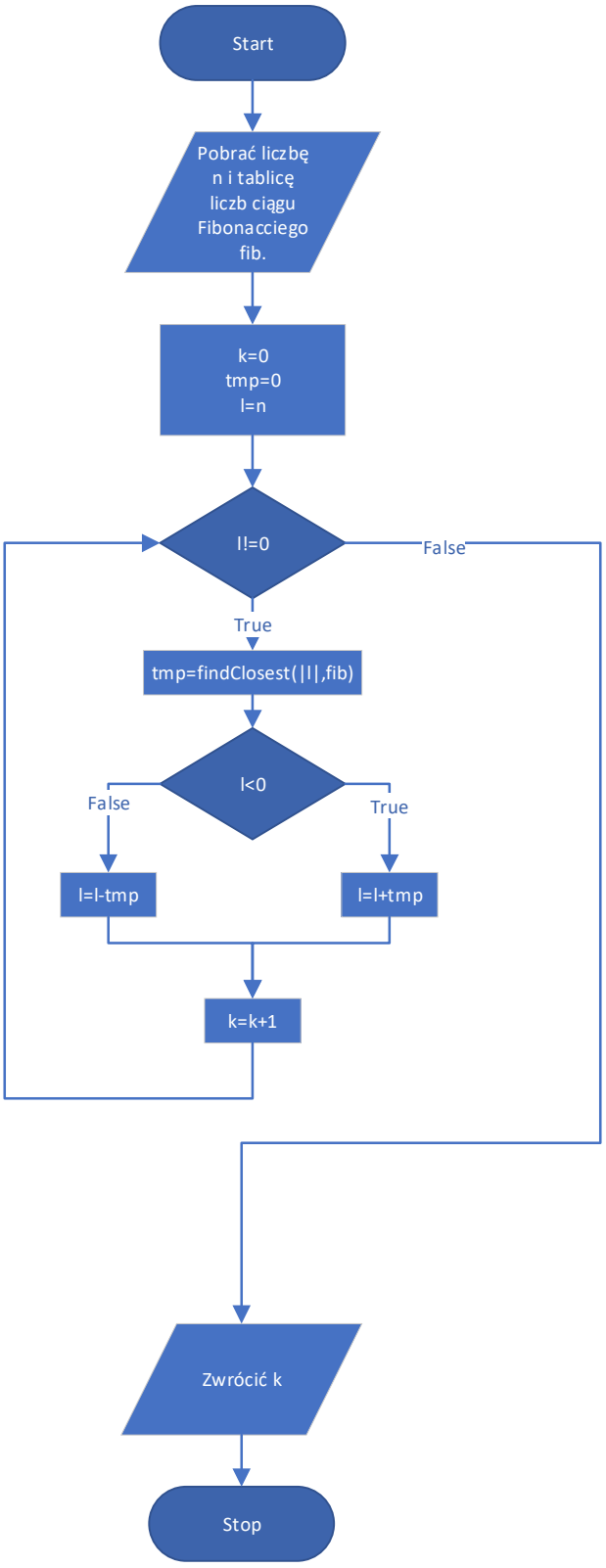
Rysunek 2 - schemat blokowy funkcji fibonacciSequence

Funkcja fibonacciSequence w skrypcie fib_decomp.py wykorzystana jest w celu obliczenia liczb Fibonacciego potrzebnych do rozwiązania zadania. W skrypcie tablica liczb Fibonacciego obliczana jest na podstawie największej liczby z pliku wejściowego, dzięki czemu w tablicy znajdują się wszystkie liczby Fibonacciego mniejsze od niej oraz jedna większa, które potrzebne są do rozwiązania zadania.



Rysunek 3 – schemat blokowy funkcji findClosest

Funkcja findClosest w skrypcie fib_decomp.py wykorzystana jest w celu znalezienia liczby Fibonacciego najbliższej do podanej liczby.



Rysunek 3 - schemat blokowy funkcji findMinFib

Funkcja findMinFib w skrypcie fib_decomp.py wykorzystana jest w celu znalezienia rozwiązania zadania i zwraca minimalną liczbę liczb Fibonacciego potrzebną do zapisania rozważanej liczby jako ich sumy lub różnicy.

4. Implementacja

4.1. Skrypt menu.bat

@echo off cd /D "%~dp0" :menu echo Rozklad Fibonacciego echo 1. Rozpocznij echo 2. Zakoncz set /p option="wybor: " if %option%==1 goto start if %option%==2 goto end :start set /p l1="sciezka do plikow wejsciowych: " set /p l2="sciezka docelowa: " call "fib_decomp.bat" %l1% %l2% goto menu :end pause	Skrypt menu.bat wyświetla proste menu w wierszu poleceń. Użytkownik może wybrać opcję rozpocznij i zakończ. W przypadku opcji rozpocznij pytany jest o katalog wejściowy oraz docelowy, na których zostanie wykonany skrypt fib_decomp.bat. W przypadku opcji zakończ skrypt kończy działanie.

4.2. Skrypt fib_decomp.bat

@echo off REM arg 1- folder z plikami wejsciowymi, arg 2- folder docelowy REM utworzyc foldery backup i output if not exist "%2\backup\" (mkdir "%2\backup") if not exist "%2\output\" (mkdir "%2\output") REM pobrac obecna date i czas set "d=%date:~-4%%date:~3,2%%date:~0,2%%time:~0,2%%time:~3,2%%time:~6,2%" set /A "n=0" SETLOCAL EnableDelayedExpansion REM dla wszystkich plikow wejsciowych .txt utworzyc backup i wykonac rozklad if exist "%1\" (cd /D "%~dp0" type nul>"%2\report_%d%.txt" for %%i in ("%1*.txt") do (set /A "n+=1"	Skrypt fib_decomp.bat jako argument pierwszy pobiera ścieżkę do folderu z plikami wejściowymi i jako drugi argument ścieżkę do folderu docelowego. Następnie w docelowym folderze tworzone są katalogi backup oraz output oraz plik tekstowy, na podstawie którego sporządzony zostanie raport. Pobierana jest obecna data, która wykorzystana zostanie w nazwach plików wyjściowych. Dla wszystkich plików wejściowych wywoływane są skrypty backup.py tworzący kopię oraz fib_decomp.py rozwiązujący zadanie. Następnie wywoływany jest report.py tworzący raport w pliku pdf.

```
                py "backup.py" "%%i" "%2\backup" "!d!"
                py "fib_decomp.py" "%%i"
"%2\output\out_!d!_!n!.txt" "%2\report_!d!.txt"
        )
        REM utworzyc raport
        py "report.py" "%2\report_%d%.txt" "%2"
) else (echo Folder nie istnieje)

echo Zakonczono dzialanie
endlocal
pausepause
```

4.3. Skrypt backup.py

```
import sys
from shutil import copy2

#skopiowac plik wejsciowy do folderu
backup pod nowa nazwa
inp=sys.argv[1]
backupFolder=sys.argv[2]
date=sys.argv[3]
outp=inp.split("\\")
outp=outp[len(outp)-1]
outp=outp[:-4]+"_backup_"+date+".txt"
outp=backupFolder+"\\"+outp
copy2(inp,outp)
```

Skrypt backup.py jako argumenty przyjmuje ścieżkę do pliku wejściowego oraz ścieżkę do folderu backup.
Plik wejściowy kopiowany jest pod nową nazwą (plik_backup_[data].txt) do folderu backup.

4.4. Skrypt fib_decomp.py

```
import sys
import time

#obliczyc elementy ciagu fibonacciego do elementu wiekszego od n
def fibonacciSequence(n):
    l=abs(n)
    seq=[]
    seq.append(0)
    seq.append(1)
    i=2
    while(l>=seq[i-2]+seq[i-1]):
        seq.append(seq[i-2]+seq[i-1])
        i+=1
    seq.append(seq[i - 2] + seq[i - 1])
    return(seq)

#znajduje najblizsza podanej liczbie n liczbe z ciagu fibonacciego
def findClosest(n, fib): #fib-tablica z liczbami ciągu fibonacciego
    start = 0 #indeks pierwszego elementu rozwaznej czesci
    end = len(fib)-1 #indeks ostatniego elementu rozwaznej czesci
    result=None
    cont=True

    while cont: #wykonuje sie do znalezienia najblizszej liczby
fibonacciego
        #sprawdzic czy liczba znajduje sie na koncach przedzialu
```

```
        if fib[start]==n:
            result=fib[start]
            cont=False
        elif fib[end]==n:
            result=fib[end]
            cont=False
        #jesli zostaly do rozwazenia 2 liczby zwrocic blizsza n
        elif ((start-end)==1 or (start-end)==-1):
            if (n-fib[start])<fib[end]-n:
                result=fib[start]
                cont=False
            else:
                result=fib[end]
                cont=False
        #zawezic obszar poszukiwan
        elif (fib[start+(end-start)//2]>=n):
            end=start+(end-start)//2
        else:
            start=start+(end-start)//2

    return result

#znajduje minimalna liczbe liczb fibonacciego w n
def findMinFib(n,fib): #fib-tablica z liczbami ciągu fibonacciego
    k=0
    tmp=0
    l=n
    while(l!=0): #znalezc najblizsza liczbe fibonacciego do l i dodac lub
odjac az l==0
        tmp=findClosest(abs(l),fib)
        if l<0:
            l+=tmp
        else:
            l-=tmp
        k+=1

    return(k)

start = time.time()
inp=sys.argv[1]
outp=sys.argv[2]
report=sys.argv[3]

n=[]
#wczytaj liczby z pliku
try:
    file=open(inp,"r")
    lines=file.readlines()
    for line in lines:
        n.append(int(line))
    file.close()
except Exception as e:
    print(e)
    sys.exit()


fib=fibonacciSequence(max(n))

file=open(outp,"w")
file.write("wejscie  - wyjscie\n")
res=[]
```

```
for i in n:
    file.write(str(i).ljust(8)+" -
"+str(findMinFib(i,fib)).ljust(8)+"\n")
file.close()

end = time.time()

file=open(report,"a")
file.write(inp+";"+outp+";"+str(end-start)+"\n")
file.close()
```

Skrypt fib_decomp.py przyjmuje jako argumenty ścieżkę do pliku wejściowego, ścieżkę do folderu z plikami wyjściowym oraz ścieżkę do pliku tekstowego, na podstawie którego sporządzony zostanie raport.

Skrypt wczytuje z pliku wejściowego liczby, następnie oblicza odpowiednią ilość liczb Fibonacciego.

Skrypt tworzy plik wyjściowy i zapisuje w nim wynik zadań rozwiązanych przy pomocy funkcji findMinFib na podstawie liczb z pliku wejściowego. Ścieżki do pliku wejściowego i wyjściowego oraz czas wykonywania obliczeń zapisywane są do pliku tekstowego report.

4.5. Skrypt report.py

```
import sys
import os
from fpdf import FPDF
from fpdf.enums import XPos, YPos

inp=sys.argv[1]
outp=sys.argv[2]

#pobrac dane do raportu
n=[]
try:
    file=open(inp,"r")
    lines=file.readlines()
    for line in lines:
        n.append(line.replace("\n",""))
    file.close()
    os.remove(inp)

except Exception as e:
    print(e)
    sys.exit()

for i in range(0,len(n)):
    n[i]=n[i].split(";")

inp=inp.split("\\")
inp=inp[len(inp)-1]

#utworzenie raportu w formie tabeli w pliku pdf
pdf=FPDF('P','mm','A4')
pdf.add_font("Arial",'',"C:\\Windows\\Fonts\\arial.ttf")
pdf.add_font("Arial",'B',"C:\\Windows\\Fonts\\arialbd.ttf")
pdf.add_page()
```

```
pdf.set_font('Arial','B',16)
pdf.cell(0,7,inp[:-4],new_x=XPos.LMARGIN, new_y=YPos.NEXT)

pdf.set_font('Arial','B',6)
pdf.cell(7, 4, "nr.", border=True)
pdf.cell(80, 4, "wejscie", border=True)
pdf.cell(80, 4, "wyjscie", border=True)
pdf.cell(0, 4, "czas wykonania", new_x=XPos.LMARGIN, new_y=YPos.NEXT,
border=True)

pdf.set_font('Arial','',4)
for i in range(0,len(n)):
    pdf.cell(7, 4, (str(i + 1) + "."), border=True)
    pdf.cell(80, 4, n[i][0],border=True)
    pdf.cell(80, 4, n[i][1], border=True)
    pdf.cell(0, 4, n[i][2], new_x=XPos.LMARGIN, new_y=YPos.NEXT,
border=True)


pdf.output(outp+"\\"+inp[:-4]+".pdf")
```

W celu wykonania raportu w formie pliku pdf w skrypcie report.py wykorzystano bibliotekę fpdf2 umożliwiająca tworzenie i formatowanie plików pdf.

Skrypt report.py jako argumenty przyjmuje ścieżkę do pliku tekstowego, na podstawie którego wykonywany jest raport oraz ścieżkę do lokalizacji docelowej, w której ma się pojawić raport.

Skrypt pobiera zawartość pliku tekstowego, następnie na jej podstawie tworzy plik pdf mający formę tabeli zawierającej informacje o lokalizacji plików wejściowych, wyjściowych oraz czasów rozwiązywania zadań dla poszczególnych plików.

5. Podsumowanie

W ramach projektu udało się stworzyć skrypty rozwiązujące zadanie rozkładu Fibonacciego dla wielu plików wejściowych, tworzące kopie zapasowe plików wejściowych oraz sporządzające raport z wykonanych zadań.

Dalszy rozwój:

- Rozszerzenie działania skryptu rozwiązującego zadanie rozkładu Fibonacciego tak, aby zwracał również liczby Fibonacciego wykorzystane w rozwiązaniu.
- Ponieważ zadanie rozkładu Fibonacciego może być rozwiązane przy wykorzystaniu różnych kombinacji liczb Fibonacciego, można rozszerzyć skrypt rozwiązujący zadanie, aby podawał wszystkie możliwe kombinacje oraz ich liczbę.
- Zmodyfikowanie wyglądu pliku pdf będącego raportem przy użyciu funkcji biblioteki fpdf2 w taki sposób, aby był on bardziej przejrzysty i łatwiejszy do odczytania przez użytkownika.
- Umożliwienie wykorzystania plików wejściowych o różnych formach (np. liczby przedzielone średnikiem).

6. Źródła

- https://szkopul.edu.pl/problemset/problem/w1QbhPufazp-sH6X-u4pTnNu/site/?key=statement
- https://skowronski.tech/2012/10/rozklad-fibonacciego/
- http://www.ftj.agh.edu.pl/~lenda/cicer/FIBO.HTM
- https://www.geeksforgeeks.org/number-of-ways-to-represent-a-number-as-sum-of-k-fibonacci-numbers/