Przygotowanie uczniów do matury z informatyki poprzez opracowanie algorytmów zadania 4.2 na maturze z 2015 r.

Roland Zimek

Zadanie 4.2.

Podaj, ile liczb w pliku `liczby.txt` jest podzielnych przez 2 oraz ile liczb jest podzielnych przez 8. Przykład: Dla zestawu liczb:

```
10101101001100110000 (*), (**)
10001001
100100 (*)
10101001010101101000 (*), (**)
100011
```

trzy liczby są podzielne przez 2 (*) i dwie liczby są podzielne przez 8 (**).

Wszystkie analizowane wiersze zawierają tekst składający się z zer i jedynek. Każdy wiersz jest zapisem liczby w systemie binarnym. Ponieważ naszym zadaniem jest zliczenie liczb podzielnych przez 2 i podzielnych przez 8, przekształcimy te liczby na system dziesiętny.

Wiemy, że w tym celu należy począwszy od prawej strony pomnożyć każdy znak przez właściwą potęgę dwójki. Potęgą jest miejsce na którym stoi cyfra, licząc od prawej strony, pomniejszone o 1. Dzięki temu dla prawej skrajnej cyfry, właściwą potęgą jest 0.

W programie dla każdego kolejnego wiersza będę brał po kolei każdą cyfrę i obliczał dla niej jej wartość dziesiętną, uwzględniając miejsce na którym się znajduje, korzystając ze wzoru `cyfra*2`^{n}. Wartość `cyfry` wyznaczę pobierając znak stojący na danej pozycji, a następnie przekształcę go do systemu dziesiętnego funkcją `int()`. Potęgi powinniśmy liczyć od prawej strony liczby dwójkowej, natomiast w języku Python tekst jest indeksowany od `0` począwszy od lewej strony. Właściwą potęgę obliczę więc odejmując od długości sprawdzanego wiersza, indeks na którym się znajduje, pomniejszając wynik o 1.

Następnie będę sumował obliczoną dziesiętną wartość każdego znaku binarnego w zmiennej `dziesietnie`. Jeżeli wartość zmiennej dziesiętnie będzie dzieliła się bez reszty przez 8 lub przez 2, to będę zwiększał o 1 odpowiednie zmienne.

```
with open("liczby.txt") as plik:
    binarne = [i for i in plik]

li2 = li8 = 0
for wiersz in binarne:
    dziesietnie = 0
    for i in range(len(wiersz.strip())):
        dziesietnie += int(wiersz[i])*2**(len(wiersz.strip())-i-1)
    if dziesietnie % 8 == 0:
        li8 += 1
    if dziesietnie % 2 == 0:
        li2 += 1
print(li2, li8)
```

Za każdym razem gdy obliczałem długość wiersza, usuwałem białe znaki metodą `.strip()`.

Powyższy program nie jest zbytnio skomplikowany ani zbyt długi. Wymaga jednak dobrej znajomości sposobu przeliczania systemu binarnego na dziesiętny, co jak widać w programie łatwo może spowodować błędne zapisy, które trudno będzie zlokalizować.

Dlaczego w takim razie nie przeliczyć od razu całego wiersza na system dziesiętny? Przecież korzystam z funkcji `int()` która do tego służy. Gdyż w swojej podstawowej postaci przekształca ona tekst zawierający liczbę w systemie dziesiętnym na jej wartość liczbową. W przypadku cyfr `0` i `1` daje to odpowiednie wyniki. Jednak zapis `int("101")` nie da w wyniku spodziewanej wartości dziesiętnej 5, lecz wartość dziesiętną 101 (sto jeden).

Na szczęście możemy poinformować funkcję `int()` z jakiego systemu liczbowego ma przeliczyć zapis. Wystarczy po przecinku wpisać podstawę systemu. Dzięki temu, zapis `int("101", 2)` da w wyniku liczbę 5 w systemie dziesiętnym.

Wiedząc o tym można program przekształcić do bardziej przejrzystej postaci:

```
with open("liczby.txt") as plik:
    binarne = [i for i in plik]

li2 = li8 = 0
for wiersz in binarne:
    dziesietnie = int(wiersz.strip(), 2)
    if dziesietnie % 8 == 0:
        li8 += 1
    if dziesietnie % 2 == 0:
        li2 += 1
print(li2, li8)
```

Jak zapewne bardziej obeznani w systemie binarnym zauważą, nie ma nawet potrzeby przeliczać wartości binarnej na dziesiętną, gdyż podzielniki 2 i 8 w systemie dziesiętnym oznaczają, że muszą one posiadać końcówki odpowiednio `0` oraz `000`, co w konsekwencji pozwoli zapisać program w poniższej postaci:

```
with open("liczby.txt") as plik:
    binarne = [i for i in plik]

li2 = li8 = 0
for wiersz in binarne:
    if wiersz.strip()[-3:] == "000":
        li8 += 1
    if wiersz.strip()[-1] == "0":
        li2 += 1
print(li2, li8)
```

W programie użyłem zapisów `[-3:]` oraz `[-1]`, aby wyodrębnić z wierszy odpowiednią liczbę znaków z prawej strony.

To nie wszystkie sposoby rozwiązania tego zadania. Ciekawym i bardzo szybkim sposobem będzie zastosowanie odmiany list składanych, tak zwanego filtrowania list[1]. Filtrowanie list zwraca sekwencje (tego samego typu, gdy to możliwe) zawierającą te elementy z sekwencji wejściowej, dla których wywołanie funkcji zwróci wartość prawdziwą.

Ogólna postać filtrowania list jest następująca:

wartość **for** *wartość* **in** *lista* **if** *warunek*

Zwraca ona listę zwierającą jedynie te elementy, dla których *warunek* okaże się prawdziwy.

Przykładowy program:

```
binarne = ["100010", "11", "110100", "10", "11101", "110001"]


binarne2 = [wiersz for wiersz in binarne if wiersz[-1] == "0"]
li2 = len(binarne2)
print(binarne2)
print(li2)
```

wyświetli na ekranie:

```
['100010', '110100', '10']
3
```

Ponieważ wynikiem działania filtrowania list jest nowa lista, utworzę nowe zmienne: `li8` i `li2`. Otrzymają one listy zawierające jedynie wiersze kończące się odpowiednia na `000` oraz `0`.

[1] Wykorzystuję tutaj programowanie funkcyjne, zamiast funkcji `filter()`, która nie jest obecnie zalecana do stosowania i pozostawiona w języku Python dla zachowania zgodności ze starszymi wersjami. Możliwe, że w przyszłości zostanie usunięta z języka Python.

```
with open("liczby.txt") as plik:
    binarne = [i for i in plik]

li8 = [wiersz for wiersz in binarne if wiersz.strip()[-3:] == "000"]
li2 = [wiersz for wiersz in binarne if wiersz.strip()[-1] == "0"]
print(len(li2), len(li8))
```