Wrocław, 16.11.2021r.

Lista zadań nr 3

Julia Mazur, 262296

Zadanie 1.

$$P(n,k) = \sum_{i=0}^{k} \frac{n!}{i!(n-i)!} p^i (1-p)^{n-i}$$

Aby zredukować ilość mnożeń używanych w podanym wzorze, musimy wykonać następujące przekształcenia:

1. Wyłączyć $(1-p)^n$ przed sumę otrzymując w ten sposób $(1-p)^n \sum_{i=0}^{k} \frac{n!}{i!(n-i)!} (\frac{p}{1-p})^i$.
2. Rozpisać $\frac{n!}{i!(n-i)!}$ jako $\frac{(n-1)!}{(i-1)!(n-i)!} + \frac{(n-1)!}{i!(n-1-i)!}$ i tworzymy funkcję rekurencyjną.
3. Wykorzystać fakt, że do wyznaczenia kolejnych wyrazów sumy używamy kolejnych potęg $\frac{p}{1-p}$ zatem możemy wykorzystać poprzednio wyliczoną potęgę do wyznaczenia kolejnej.
4. Znaleźć inny sposób na obliczanie potęgi całkowitej danej liczby (taki, który wymaga mniejszej ilości obliczeń). Możemy zauważyć, że gdy chcemy obliczyć x^{16} to możemy wykonać 15 mnożeń albo odpowiednio podzielić potęgi otrzymując $(((x^2)^2)^2)^2$, co daje nam ten sam wynik a wymaga jedynie 4 mnożeń (oczywiście nasz program musi również uwzględniać dzielenia wykonywane na 16, zatem mnożeń będzie dwa razy więcej), podobnie dla liczby nieparzystej możemy rozbić $x^{15} = x(x(x * x^2)^2)^2$ i zamiast 14 mnożeń mamy 6 (bez uwzględnienia dzielenia 15 przez 2).

Z (2) mamy zredukowaną ilość mnożeń w obliczaniu dwumianu newtona do 0.

Z (3) wiemy, że do obliczenia wszystkich potęg $\frac{p}{1-p}$ wystarczy wykonać k-1 mnożeń.

Patrząc na (1) możemy zauważyć że potrzebne nam jeszcze k mnożeń dla każdego z wyrazów sumy.

Następnie musimy wyznaczyć górną granicę ilości mnożeń wykonywanych przez funkcję opisaną w (4). W najgorszym przypadku przy każdym dzieleniu n przez 2 (gdzie n to wykładnik potęgi o podstawie x) musimy wykonać trzy mnożenia ($x * x^2$ oraz $n//2$). Stąd możemy zapisać, że ilość obliczeń to $3y$.

Teraz ustalamy y.

Wiemy, że $n < 2^y$(bo wynik dzielenia całkowitego n przez 2^y musi wynosić 0), czyli możemy założyć, że $n = 2^{y-1}$, bo wtedy powyższy warunek jest na pewno spełniony. Po przekształceniu otrzymujemy, że $y = log_2 n + 1$, czyli ilość mnożeń w tym kroku to $3log_2 n + 3$.

Ostatecznie należy uwzględnić dzielenie $\frac{p}{1-p}$,które wykonujemy na początku działania programu (z pominięciem przypadku gdzie k wynosi 0, wtedy program nie wykonuje tego działania) oraz mnożenie bazy $(1 - p)^n$przez sumę końcową
Zatem ostateczna ilość mnożeń, której ten program nie przekroczy to
$2k + 3log_2 n + 4$.

Przykładowe wywołania:

```
x = L3_ZAD1.probability(20,3,0.2)
```

```
y = L3_ZAD1.probability(100,5,0.5)
```

```
x
```

```
(0.41144886195656943, 20)
```

```
y
```

```
(6.26162256269268e-23, 29)
```

$2 * 3 + 3log_2 20 + 4 \approx 23$

$2 * 5 + 3log_2 100 + 4 \approx 34$

```
L3_ZAD1.probability(15,15,0.5)
```

```
(1.0, 44)
```

```
L3_ZAD1.probability(13,0,0.1)
```

```
(0.25418658283290013, 11)
```

$2 * 15 + 3log_2 15 + 4 \approx 46$

$2 * 0 + 3log_2 13 + 4 \approx 15$

link do kodu zadania:
https://github.com/Swinkawkrawacie/algorytmy2021-22/blob/e3f30098a535028c350fcaa552eab9bec2f19c08/lista3/L3_ZAD1.py

Zadanie 2.

$$W(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

Korzystając ze schematu Hornera możemy ograniczyć ilość mnożeń. Powyższy wzór możemy zapisać jako:

$$W(x) = a_0 + \sum_{i=0}^{n-1} x a_{n-i}$$

Dzięki czemu dla k-elementowej listy musimy wykonać k-1 mnożeń.
Wywołania dla nieoptymalnego programu (liczącego kolejne potęgi i mnożącego przez elementy z listy):

```
L3_ZAD2.ordinary_polynomial_value_calc([1,2,3,4,5,6,7],5)
```

```
(131836, 21, 7)
```

```
L3_ZAD2.ordinary_polynomial_value_calc([7,3,2,1,9,4,5,8],5)
```

```
(721447, 28, 8)
```

Wywołania dla optymalnego programu (wykorzystującego przedstawiony schemat):

```
L3_ZAD2.smart_polynomial_value_calc([1,2,3,4,5,6,7],5)
```

```
(131836, 6, 6)
```

```
L3_ZAD2.smart_polynomial_value_calc([7,3,2,1,9,4,5,8],5)
```

```
(721447, 7, 7)
```

Widzimy, że ilość mnożeń znacznie się zmniejszyła w przypadku optymalnego programu, bo aż od 3.5 do 4 razy.

link do kodu zadania:
https://github.com/Swinkawkrawacie/algorytmy2021-22/blob/e3f30098a535028c350fcaa552eab9bec2f19c08/lista3/L3_ZAD2.py

Zadanie 3.

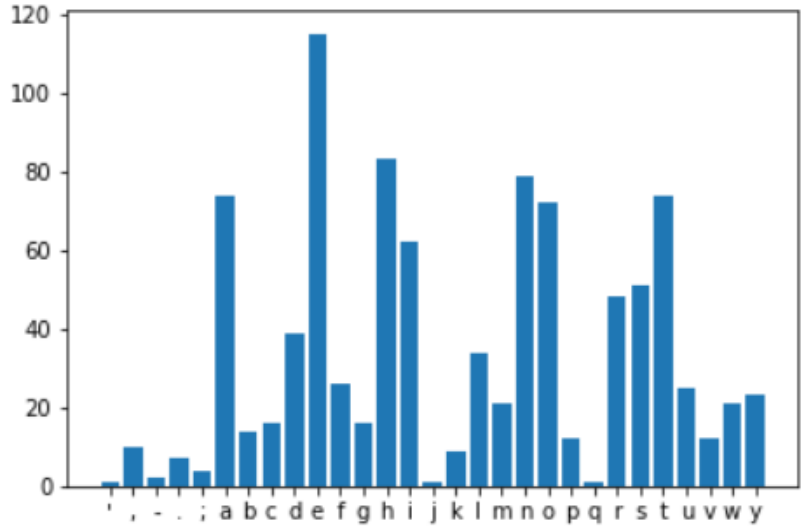
Aby wykonać to zadanie dany tekst sformatowałam tak, aby zawierał tylko małe litery (pozostałe znaki oczywiście również były zliczane). Stworzyłam listę oraz zbiór wszystkich elementów tego tekstu. Zbiór przekształciłam w drugą listę i obie posortowałam. Następnie zliczyłam wystąpienie kolejnych znaków usuwając kolejno elementy z pierwszego (a w zasadzie zerowego) miejsca listy. Następnie usunęłam znaki białe (spacje, znaki nowej linii oraz tabulatory). I zwróciłam ostateczny wynik w formie słownika.

Przykład działania funkcji (na podstawie udostępnionego tekstu):

```
L3_ZAD3.counting_chars_without_ifs('L3_ZAD3_sample_text.txt')
```

```
{"'": 1,
 ',': 10,
 '-': 2,
 '.': 7,
 ';': 4,
 'a': 74,
 'b': 14,
 'c': 16,
 'd': 39,
 'e': 115,
 'f': 26,
 'g': 16,
 'h': 83,
 'i': 62,
 'j': 1,
 'k': 9,
 'l': 34,
 'm': 21,
 'n': 79,
 'o': 72,
 'p': 12,
 'q': 1,
 'r': 48,
 's': 51,
 't': 74,
 'u': 25,
 'v': 12,
 'w': 21,
 'y': 23}
```

Żeby wynik był bardziej czytelny przedstawiam go na wykresie:



link do kodu zadania:
https://github.com/Swinkawkrawacie/algorytmy2021-22/blob/e3f30098a535028c350fcaa552eab9bec2f19c08/lista3/L3_ZAD3.py