Zajęcia 10: funkcje mieszające (inne nazwy: hash function, suma kontrolna, funkcja skrótu)

Funkcje mieszające liczą sumę kontrolną (o ustalonej ilości bitów) ciągu znaków (albo pliku). Ważne jest żeby prawdopodobieństwo, że dwa różne pliki mają taką samą sumę kontrolną (tzw. kolizja) było jak najmniejsze.
Np. można rozważyć sumę kontrolną, która będzie sumą kodów ascii wszystkich znaków w pliku, ale będzie to "zła" suma kontrolna, bo np. "test" i "tset" będą miały taką samą sumę kontrolną.

Poniżej opis dwóch bardzo prostych sum kontrolnych:

- funkcja DJB
  - *hash* jest 4-bajtową liczbą całkowitą dodatnią i startuje jako 5381_{10} = $1\ 0101\ 0000\ 0101_2$
  - wczytując kolejny *znak* liczymy *hash* ==> *hash* *32 + *hash* + *znak*, tj. wymnażamy nasz *hash* przez 32 (co jest równoznaczne przesunięciu w lewo o 5 bitów), dodajemy starą wartość i kod ASCII odczytanego znaku.
  - Po odczytaniu ostatniego znaku hash to nasza wartość funkcji mieszającej

Uwaga – bardzo szybko wyjdziemy poza zakres 32 bitów i najwyższe bity się „zgubią" ale nic nie szkodzi, jest to część algorytmu, żeby „podsumować" cały łańcuch tekstowy w 32 bitach

Przykład dla „test":
t: 177689
te: 5863838
tes: 193506769
test: 2090756197 - tu wyszliśmy poza zakres i suma kontrolna zaczyna być przycięte do 32 bitów

- funkcja adler32 (z biblioteki kompresji zlib),
  - dwie sumy: A i B, zainicjowane A=1, B=0, A będzie sumą wszystkich bajtów, B będzie sumą wszystkich A, obie liczone mod P = 65521 (największa liczba pierwsza < 2^{16})
  - na każdym kroku A ==> (A+ znak) mod P
    B==> (B+A) mod P
  - Na końcu adler = B * 65536 + A (albo inaczej mówiąc B przesunięte bitowo o 16 pozycji w lewo i dodane A, tak że w ostatecznej liczbie bity B i A się nie „zazębiają")

Przykład dla „test":
t: a=117 b=117
te: a=218 b=335
tes: a=333 b=668
test: a=449 b=1117 - liczby rosną wolno, więc nie doszliśmy jeszcze do dzielenia modulo przez P, które jednak zapewnia że ostateczna liczba zawsze się zmieści w 32 bitach
Adler32=1117 * 65536 + 449 = 73204161

Państwa zadanie to przetestować jak dobre są te sumy kontrolne (tj. jak często powodują kolizje)

Ilość kolizji sumy kontrolnej:
Jak mamy b bitów to ilość możliwych sum to $S=2^b$, jeżeli suma jest losowa to prawdopodobieństwo, że dwie sumy są takie same to $P_1=2^{-b}$,
Sprawdzając n sum otrzymamy losowo $n*(n-1)*2^{-b} / 2$ czyli około $(n^2 / 2) * 2^{-b}$ kolizji
(pierwsze n jest dlatego że mamy n sum, drugie n-1 bo każda suma może mieć kolizję z każdą inną, a dzielenie przez 2 dlatego, żeby nie policzyć każdej kolizji podwójnie, tj np. 1 z 2 i 2 z 1)

Przykład: int ma 32 bity, tj 4×10^9 możliwości, jeżeli wylosujemy 2^{16} = 65 536 ciągów tekstu to mamy szanse 0.5 na złapanie kolizji. To jest tylko dla „dobrej" sumy kontrolnej, która jest faktycznie losowa, w innym przypadku ilość kolizji będzie większa.

Do zrobienia:

1. dwie funkcje które liczą adler32 i DJB dla podanego stringa
2. wygenerować 100 000 losowych ciągów znaków (składających się tylko z małych i dużych liter a-zA-Z) o podanej długości
3. policzyć sumy kontrolne każdego ciągu znaków
4. policzyć ilość kolizji oraz wypisać (tylko 1 !) kolizje (oba ciągi znaków i ich wspólną sumę kontrolną) o ile jakaś została znaleziona
5. punkty 2-4 wykonać dla obu sum kontrolnych oraz długości ciągu znaków D równej 8 i 100 liter

wynik proszę zaprezentować w dokładnie takiej postaci:
ADLER 32, D=8, N=100 000
XXX kolizji
<pierwszy_tekst> <drugi_tekst> <wspólna_wartość_sumy_kontrolnej>

ADLER 32, D=100, N=100 000
YYY kolizji
<pierwszy_tekst> <drugi_tekst> <wspólna_wartość_sumy_kontrolnej>

DJB, D=8, N=100 000
ZZZ kolizji
<pierwszy_tekst> <drugi_tekst> <wspólna_wartość_sumy_kontrolnej>

DJB, D=100, N=100 000
NNN kolizji
<pierwszy_tekst> <drugi_tekst> <wspólna_wartość_sumy_kontrolnej>

Prosze NIE wypisywać wszystkich kolizji, a jedynie jedną