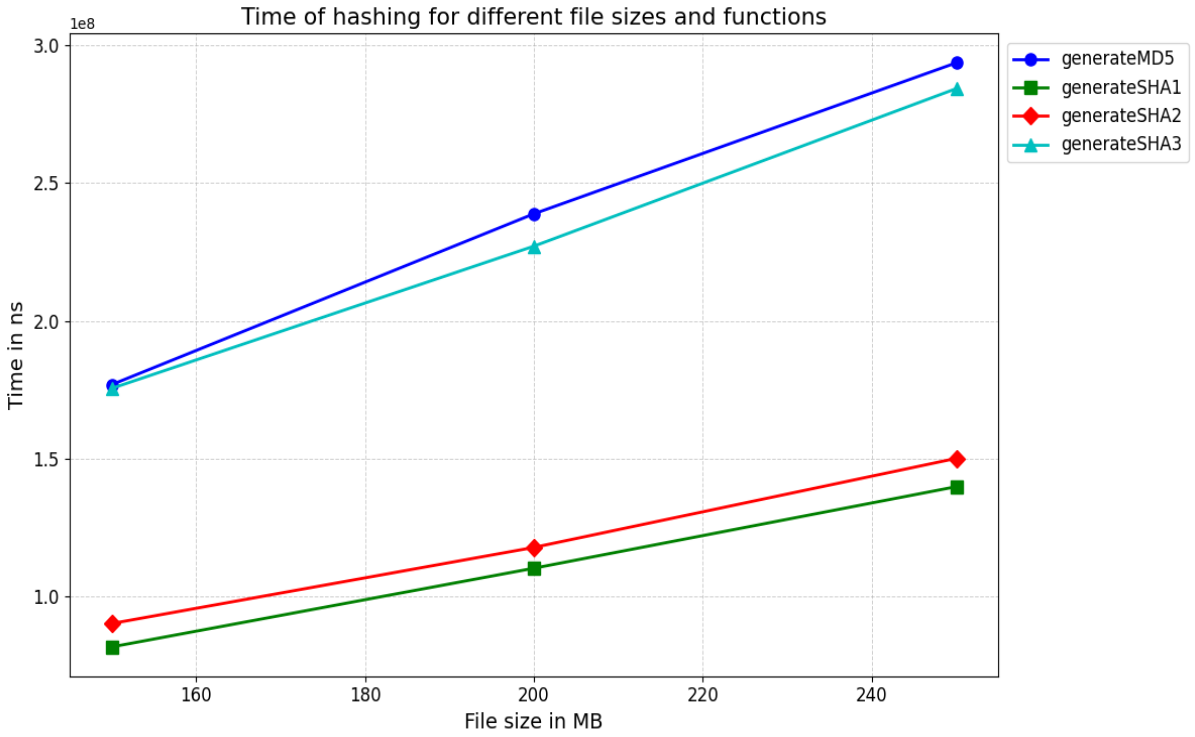
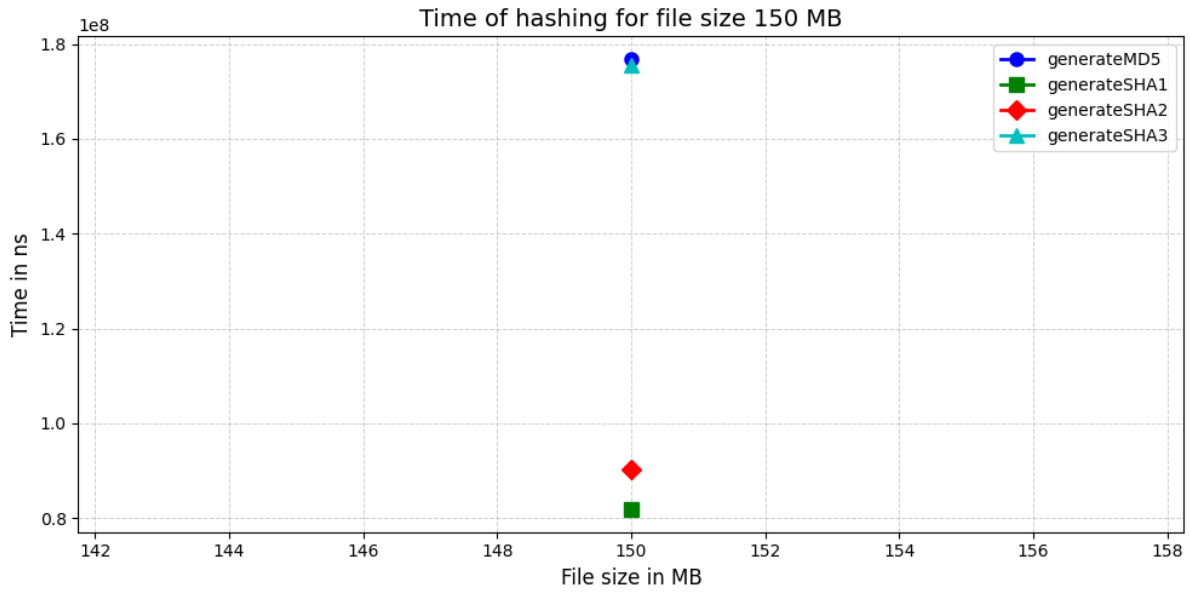
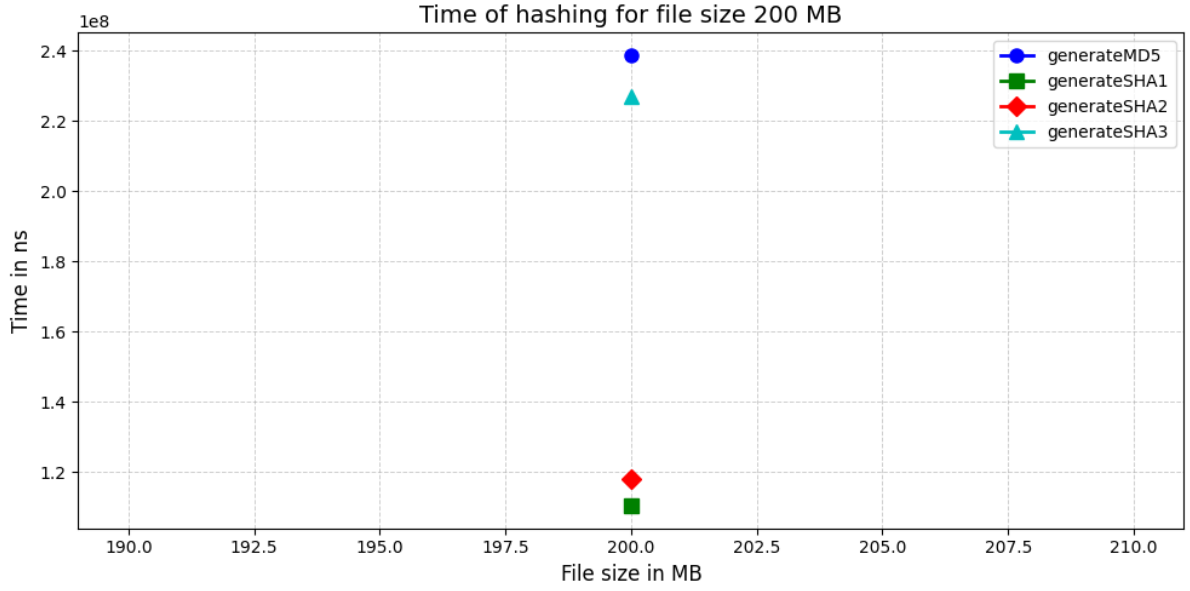
Miłosz Sobol 155905 Funkcje skrótu

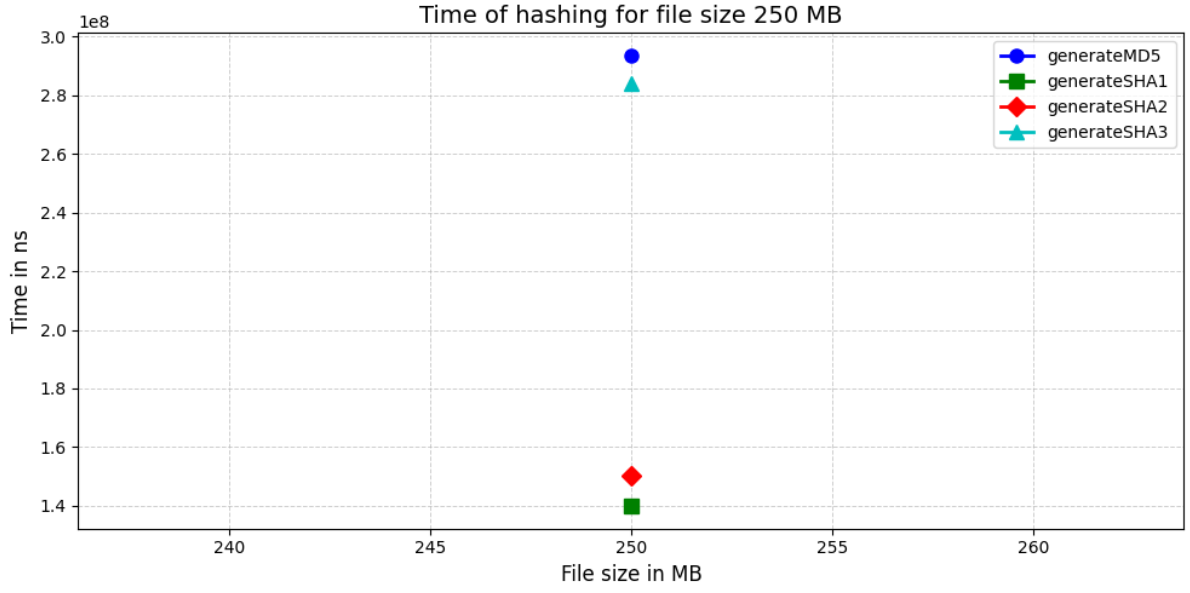
1. Zestawienie zależności czasu potrzebnego na wykonanie funkcji skrótu do wielkości przetwarzanego pliku.



Widoczny jest liniowy wzrost czasu zależnego od wielkości pliku przyjmowanego na wejście funkcji. Czasy te, mimo sporej wielkości plików, są bardzo niskie, świadczy to o niskiej złożoności obliczeniowej haszowania. Jest to zarówno plus jak i minus, niska złożoność obliczeniowa pozwala na stosunkowo szybkie przeprowadzanie ataków takich jak „atak urodzinowy” polegający na znalezieniu kolizji funkcji haszującej.

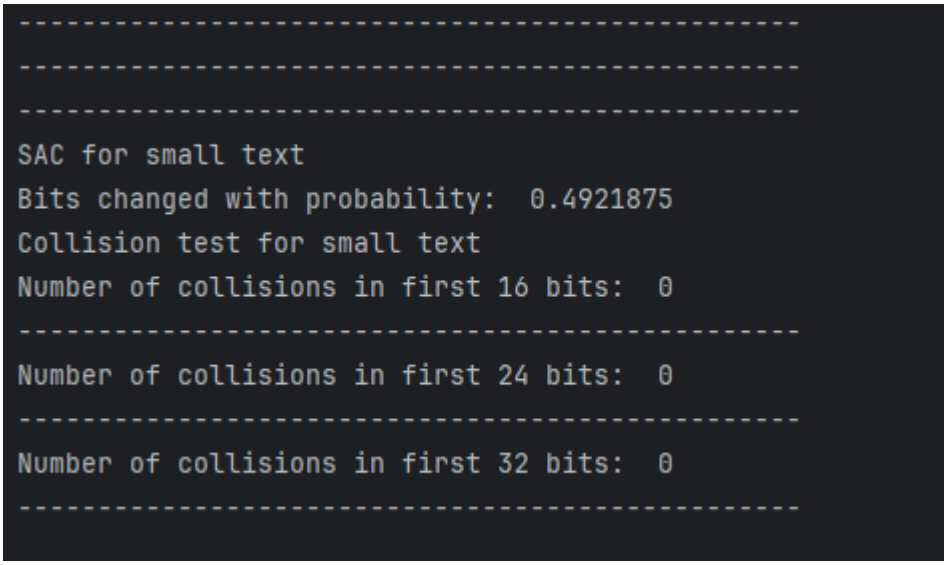

Time of hashing for file size 150 MB
1e8
1.8
1.6
1.4
1.2
1.0
0.8
Time in ns
142
144
146
148
150
152
154
156
158
File size in MB
generateMD5
generateSHA1
generateSHA2
generateSHA3


Time of hashing for file size 200 MB
1e8
2.4
2.2
2.0
1.8
1.6
1.4
1.2
Time in ns
190.0
192.5
195.0
197.5
200.0
202.5
205.0
207.5
210.0
File size in MB
generateMD5
generateSHA1
generateSHA2
generateSHA3


Time of hashing for file size 250 MB
1e8
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
Time in ns
240
245
250
255
260
File size in MB
generateMD5
generateSHA1
generateSHA2
generateSHA3

Na przedstawionych na poprzedniej stronie wykresach możemy zaobserwować pewną rozbieżność. Haszowanie za pomocą funkcji skrótu MD5 zajmuje niemalże tyle samo czasu co za pomocą SHA384. Owa rozbieżność najprawdopodobniej wynika z specyfikacji implementacji funkcji MD5 w Pythonowej bibliotece „hashlib” (https://stackoverflow.com/questions/59955854/what-is-md5-md5-and-why-is-hashlib-md5-so-much-slower).

2. Test SAC



Test został parokrotnie przeprowadzony dla plików o różnych rozmiarach (Na obrazku widoczny wynik dla pliku o wielkości 150MB oraz funkcji haszującej SHA384). Za każdym razem wynik oscylował w okolicach wartości 0.5.

3. Test kolizji

```
smallText = "ko"
#zad3
print("Collision test for small text")
res = solution.testCollision(smallText.encode(), numberOfTests: 1000 , solution.generateMD5Digest, numberOfTestingBytes: 2)
print('Number of collisions in first 16 bits: ', res)
printLine()
res =solution.testCollision(smallText.encode(), numberOfTests: 1000, solution.generateMD5Digest, numberOfTestingBytes: 3)
print('Number of collisions in first 24 bits: ', res)
printLine()
res =solution.testCollision(smallText.encode(), numberOfTests: 1000, solution.generateMD5Digest, numberOfTestingBytes: 4)
print('Number of collisions in first 32 bits: ', res)
printLine()
```

Mimo wykorzystania funkcji haszującej MD5, test charakteryzował się zupełnym brakiem kolizji dla stringa o długości 3 znaków. Poniżej widoczne wyniki dla kolejno, słowa „k” oraz słowa „ko”.

```
Collision test for small text
Number of collisions in first 16 bits:  16
--------------------------------------------------
Number of collisions in first 24 bits:  16
--------------------------------------------------
Number of collisions in first 32 bits:  11
--------------------------------------------------
```

```
Collision test for small text
Number of collisions in first 16 bits:  0
--------------------------------------------------
Number of collisions in first 24 bits:  1
--------------------------------------------------
Number of collisions in first 32 bits:  0
--------------------------------------------------
```