This example demonstrate how powerful hash tables really are in exact-match queries.

Example: *"Create two lists of length N containing random strings of length M. How many common strings do they share?"*

With help of `random` and `string` modules we can create two separate lists and then check how many elements from the first list are in the second list. Let $N = 10^5$ and $M = 5$:

Example 1.1: the solution using lists

```
# example1.py

from random import sample, seed
from string import ascii_lowercase, ascii_uppercase

seed(1)     # to get same result each time

all = ascii_lowercase + ascii_uppercase
N = 10**5
M = 5

list1 = [''.join(sample(all, M)) for _ in range(N)]
list2 = [''.join(sample(all, M)) for _ in range(N)]

count = 0
for word in list1:
    if word in list2:
        count += 1

print(count)
```

Example 1.2: the output with runtime

```
real    1m20.411s
user    1m20.228s
sys     0m0.050s
```

We get the solution but it takes over a minute to compute. The problem of using lists in this kind of task is that searching from list is an $\Theta(n)$ operation. Because the search is repeated n times the whole program performs in $\Theta(n^2)$ time.

With hash tables searching can be done in close to $\Theta(1)$ time making it great choice for these kind of problems. Therefore, the whole program will perform in $\Theta(n)$ time. In Python `set` uses hashing to store values.

Example 2.1: the solution using sets:

```
# example2.py

from random import sample, seed
from string import ascii_lowercase, ascii_uppercase

seed(1)     # to get same result each time

all = ascii_lowercase + ascii_uppercase
N = 10**5
M = 5

set1 = {''.join(sample(all, M)) for _ in range(N)}
set2 = {''.join(sample(all, M)) for _ in range(N)}

count = 0
for word in set1:
    if word in set2:
        count += 1

print(count)
```

Example 2.2: the output with runtime

```
$ time python3 example2.py
29

real    0m0.528s
user    0m0.514s
sys     0m0.013s
```

With this approach we get the solution in less than one second which is a massive improvement!

Last modified: Monday, 3 July 2023, 9:54 AM