Family Name : Chabane Chaouche
First Name : Rabah
ID : 222231485010
Soft ING 3rd

Part I

Q1 . Sum Algorithm

Algorithm 1 Sum of First N Integers

```
Var
n, sum, i integer;
Begin
sum ← 0;
i ← 1;
print('Input Integer N>=1 : ')
Read(n);
while i <= n do
    sum ← sum + i;
    i ← i + 1;
end while
print('Sum is ' ,sum);
End
```

Q2 . Time Complexity

we have $f(n) = \sum fr$ since $f(n)$ is sum of frequency of execution (fr) we need to figure out the fr of each instruction and sum them :

sum ← 0	$fr = 1$ (one affectation)
i ← 1	$fr = 1$ (one affectation)
print('Input Integer N>=1 : ')	$fr = 1$ (one print)
Read(n)	$fr = 1$ (one read)
while i <= n **do**	$fr = n + 1$ (check the while condition n+1 times)
sum ← sum + i	$fr = 2n$ (one affectation and one arithmetic operation + (2) inside a while that loops n times (2n))
i ← i + 1	$fr = 2n$ (one affectation and one arithmetic operation + (2) inside a while that loops n times (2n))
print('Sum is ' ,sum)	$fr = 1$ (one print)

$$
\begin{aligned}
f(n) &= \sum fr \\
&= 1+1+1+1+(n+1)+2n+2n+1 \\
&= 5n+6 \\
&= \boxed{5n+6}
\end{aligned}
$$

now that we have the complexity function $f(n)$ we need to find the $T(n)$ we have $T(n) = f(n) \times \Delta t$

$$
\begin{aligned}
T(n) &= f(n) \times \Delta t \\
&= (5n+6) \times \Delta t \\
&= \underbrace{5\Delta t}_{a} n + \underbrace{\Delta t 6}_{b} \\
&= \boxed{an+b}
\end{aligned}
$$

Q3 . Space Complexity

We have in the algorithm a total of 3 variables and 8 instructions in 3+8 = 11 Byte it's constant $\sim O(1)$

Q4 . C Code PSUM_1.c

```c
#include <stdio.h>


int main () {

long N;
long sum = 0;
int index = 1;

printf("Input Integer N >= 1 : ");
scanf("%ld",&N);

while(index<=N){
sum = sum +index;
++index;
}

printf("Sum is %ld\n",sum);

return 0;

}
```

Part II

Q1 . C Code With Clock PSUM_2.c

```c
#include <stdio.h>
#include <time.h>

int main () {

clock_t start_time = clock();

long N;
long sum = 0;
int index = 1;

printf("Input Integer N >= 1 : ");
scanf("%ld",&N);

while(index<=N){
sum = sum +index;
++index;
}

printf("Sum is %ld\n",sum);

clock_t end_time = clock();

double execution_time = (double) (end_time - start_time)/CLOCKS_PER_SEC;

printf("Execution Time %f seconds\n",execution_time);

return 0;

}
```

Q2 . Tables

Experimental

N	10^3	2.10^3	10^4	2.10^4	10^5	2.10^5	10^6	2.10^6	10^7	2.10^7	10^8	2.10^8	10^9	2.10^9
Time (10^{-5})	8	9.8	10.8	19.4	33.6	59.3	261.1	505.7	2458.4	5071.6	24458.7	48759.0	243312.2	487828.6

Theoritical

We first need to find Δt , for that we will take one runtime value from the experimental study and solve a simple equation for n = 10^4 and execution time T(n) = 10.8×10^{-5} :

$$f(n) \times \Delta t = T(n)$$

$$\Delta t = \frac{T(n)}{f(n)}$$

$$\Delta t = \frac{T(n)}{5n+6}$$

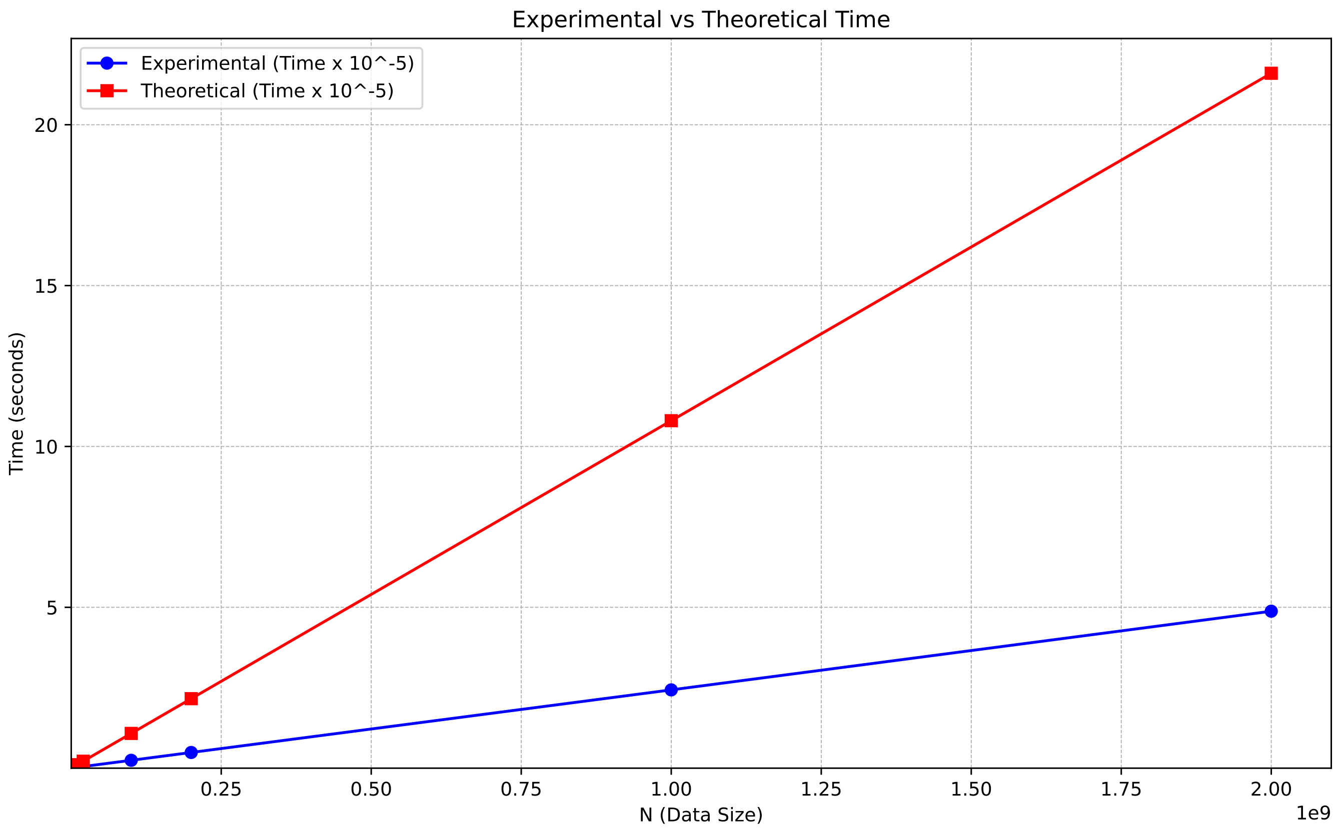
$$\Delta t = \frac{10.8 \times 10^{-5}}{5 \times 10^4 + 6}$$

$$\Delta t = 2.1597408311 \times 10^{-9}$$

$$\boxed{\Delta t \approx 2.16 \times 10^{-9}}$$

N	10^3	2.10^3	10^4	2.10^4	10^5	2.10^5	10^6	2.10^6	10^7	2.10^7	10^8	2.10^8	10^9	2.10^9	10^{10}	2.10^{10}
Time (10^{-5})	1.08	2.16	10.08	21.6	108	216	1080	2160	10800	21600	108 $\times 10^3$	216 $\times 10^3$	108 $\times 10^4$	216 $\times 10^4$	108 $\times 10^5$	216 $\times 10^5$

Q3 . Plots



Conclusion

From the two plots we can conclude that the theoritical complexity plot is similar to the experimental complexity plot, and also that theoritcal complexity follows a pattern in values since its Δt has a static value

To Draw the plots i used the below python script :

```python
import matplotlib.pyplot as plt
import numpy as np

# Data from the tables
N = [1e3, 2e3, 1e4, 2e4, 1e5, 2e5, 1e6, 2e6, 1e7, 2e7, 1e8, 2e8, 1e9, 2e9]
experimental_time = [8, 9.8, 10.8, 19.4, 33.6, 59.3, 261.1, 505.7, 2458.4, 5071.6, 24458.7, 48759.0, 243312.2,
    487828.6]
theoretical_time = [1.08, 2.16, 10.08, 21.6, 108, 216, 1080, 2160, 10800, 21600, 108000, 216000, 108e4, 216e4]

# Convert times to consistent scales
experimental_time = np.array(experimental_time) * 1e-5
theoretical_time = np.array(theoretical_time) * 1e-5

# Plot
plt.figure(figsize=(12, 7))

# Plot experimental time with only the first 14 N values
plt.plot(N[:14], experimental_time, 'o-', label='Experimental (Time x 10^-5)', color='blue')

# Plot theoretical time with all 16 N values
plt.plot(N, theoretical_time, 's-', label='Theoretical (Time x 10^-5)', color='red')



plt.xlim(left=min(N))  # Start x-axis at the minimum N value
plt.ylim(bottom=min(min(experimental_time), min(theoretical_time)))  # Start y-axis at the minimum time value


# Labels and title
plt.xlabel('N (Data Size)')
plt.ylabel('Time (seconds)')
plt.title('Experimental vs Theoretical Time')
plt.legend()
plt.grid(which="both", linestyle="--", linewidth=0.5)

# Save as PDF
plt.savefig('plot.pdf', format='pdf', bbox_inches='tight')

# Show the plot
plt.show()
```