

Report on runtime analysis of programs

Experimental Setup

- **Language:** C
- **Compiler:** GCC
- **Timing function:** clock() from <time.h>
- **OS:** Linux (Ubuntu)
- **Memory allocation:** malloc() (dynamic allocation)
- All arrays initialized with sequential integer values.

C Codes and Runtime Analysis Outputs

1) Arraysize=10**3, 10**4, 10**5

generate an array whose array size is mentioned as above
calculate the time taken to do sum of all elements in array

Code

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

void test_arr(int size) {
    int *arr = (int *)malloc(size * sizeof(int));
    if (arr == NULL) {
        printf("Memory allocation failed\n");
        return;
    }

    // Initialization
    for (int i = 0; i < size; i++) {
        arr[i] = i + 1;
    }
}
```

```
clock_t start, end;
long long sum = 0;

start = clock();

for (int i = 0; i < size; i++) {
    sum += arr[i];
}

end = clock();

double time_taken = (double)(end - start) / CLOCKS_PER_SEC;

printf("Array Size: %d\n", size);
printf("Sum: %lld\n", sum);
printf("Time Taken: %f s\n\n", time_taken);

free(arr);
}

int main() {
    test_arr(1000);
    test_arr(10000);
    test_arr(100000);
    return 0;
}
```

OUTPUT:

```
sumeet@LAPTOP-04N89MSC:~/infobell_OpenMpI$ ./array_sum
Array Size: 1000
Sum: 500500
Time Taken: 0.000003 s

Array Size: 10000
Sum: 50005000
Time Taken: 0.000014 s

Array Size: 100000
Sum: 5000050000
Time Taken: 0.000125 s
```

- Time increases approximately linearly.
- Confirms time complexity: $O(N)$.

2) Calculate the run time for accessing elements in row-wise and column wise for 2-d array

$N \times N \Rightarrow N$ as $10^{**2}, 10^{**3}$

Code

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

void test(int N)
{
    int arr[N][N];
    long long sum = 0;
    clock_t start, end;
```

```
// Array Initialization  
for (int i = 0; i < N; i++)  
    for (int j = 0; j < N; j++)  
        arr[i][j] = i + j;
```

```
// Row-wise traversal  
start = clock();  
for (int i = 0; i < N; i++)  
    for (int j = 0; j < N; j++)  
        sum += arr[i][j];  
end = clock();
```

```
double row_time = (double)(end - start) / CLOCKS_PER_SEC;
```

```
// Column-wise traversal  
sum = 0;  
start = clock();  
for (int j = 0; j < N; j++)  
    for (int i = 0; i < N; i++)  
        sum += arr[i][j];  
end = clock();
```

```
double col_time = (double)(end - start) / CLOCKS_PER_SEC;
```

```
printf("Array Size = %d\n", N);  
printf("Row-wise time = %f sec\n", row_time);  
printf("Column-wise time = %f sec\n\n", col_time);
```

```
}
```



```
int main()
{
    test(100);
    test(1000);
    return 0;
}
```

OUTPUT:

```
sumeet@LAPTOP-04N89MSC:~/infobell_OpenMpish ./RowColumn_Access
Array Size = 100
Row-wise time = 0.000023 sec
Column-wise time = 0.000026 sec

Array Size = 1000
Row-wise time = 0.002623 sec
Column-wise time = 0.003468 sec
```