Maximum Sum Sub-Array Problem with brute force

Code:

```
#include <iostream>
#include <vector>
#include <climits>
#include <chrono>
using namespace std;
int maxSubarrayBruteforce(const vector<int>& numbers) {
 int size = numbers.size();
 int maxSum = INT_MIN;
 for (int start = 0; start < size; ++start) {
 for (int end = start; end < size; ++end) {
 int currentSum = 0;
 for (int k = start; k <= end; ++k) {
 currentSum += numbers[k];
 }
 maxSum = max(maxSum, currentSum);
 }
 }
 return maxSum;
}
int main() {
 int numElements;
 cout << "Enter number of elements: ";
 cin >> numElements;
 vector<int> numbers(numElements);
 cout << "Enter the elements: ";
 for (int i = 0; i < numElements; ++i) {
 cin >> numbers[i];
 }
 auto startTime = chrono::high_resolution_clock::now();
```

```
 int result = maxSubarrayBruteforce(numbers);
 auto endTime = chrono::high_resolution_clock::now();
 chrono::duration<double, micro> duration = endTime - startTime;
 cout << "Brute Force: " << result << endl;
 cout << "Time taken: " << duration.count() << " microseconds" << endl;
 return 0;
}
```

Maximum Sum Sub-Array Problem with Divide n

Conquer

Code:

```
#include <iostream>
#include <vector>
#include <climits>
#include <chrono>
#include <algorithm> // Required for std::max
using namespace std;
int maxCrossingSum(const vector<int>& numbers, int left, int middle, int right) {
 int leftSum = INT_MIN;
 int rightSum = INT_MIN;
 int sum = 0;
 for (int i = middle; i >= left; --i) {
 sum += numbers[i];
 leftSum = max(leftSum, sum);
 }
 sum = 0;
 for (int i = middle + 1; i <= right; ++i) {
 sum += numbers[i];
 rightSum = max(rightSum, sum);
 }
 return leftSum + rightSum;
}
```

```
int maxSubarrayDivideConquer(const vector<int>& numbers, int left, int right) {
 if (left == right) {
 return numbers[left];
 }
 int middle = (left + right) / 2;
 int leftSum = maxSubarrayDivideConquer(numbers, left, middle);
 int rightSum = maxSubarrayDivideConquer(numbers, middle + 1, right);
 int crossSum = maxCrossingSum(numbers, left, middle, right);
 return max({leftSum, rightSum, crossSum});
}
int main() {
 int numElements;
 cout << "Enter number of elements: ";
 cin >> numElements;
 vector<int> numbers(numElements);
 cout << "Enter the elements: ";
 for (int i = 0; i < numElements; ++i) {
 cin >> numbers[i];
 }
 auto startTime = chrono::high_resolution_clock::now();
 int result = maxSubarrayDivideConquer(numbers, 0, numElements - 1);
 auto endTime = chrono::high_resolution_clock::now();
 chrono::duration<double, milli> duration = endTime - startTime;
 cout << "Divide and Conquer: " << result << endl;
 cout << "Time taken: " << duration.count() << " ms" << endl;
 return 0;
}
```

Maximum Sum Sub-Array Problem with Kadane's

Code:

```
#include <iostream>
#include <vector>
#include <climits>
#include <chrono>
#include <algorithm>
using namespace std;
int maxSubarrayKadane(const std::vector<int>& array) {
 int maxCurrent = array[0];
 int maxGlobal = array[0];
 for (size_t i = 1; i < array.size(); ++i) {
 maxCurrent = std::max(array[i], maxCurrent + array[i]);
 if (maxCurrent > maxGlobal) {
 maxGlobal = maxCurrent;
 }
 }
 return maxGlobal;
}
int main() {
 int numElements;
 cout << "Enter number of elements: ";
 cin >> numElements;
 vector<int> array(numElements);
 cout << "Enter the elements: ";
 for (int i = 0; i < numElements; ++i) {
 cin >> array[i];
 }
 auto startTime = chrono::high_resolution_clock::now();
 int result = maxSubarrayKadane(array);
```

```
 auto endTime = chrono::high_resolution_clock::now();
 chrono::duration<double, std::micro> duration = endTime - startTime;
 cout << "Kadane's Algorithm: " << result << endl;
 cout << "Time taken: " << duration.count() << " microseconds" << std::endl;
 return 0;
}
```

Kadane’s with starting and ending index of subarray

Code:

```
#include <iostream>
#include <vector>
#include <climits>
#include <chrono>
using namespace std;
struct Result {
 int maxSum;
 int startIndex;
 int endIndex;
};
Result maxSubarrayKadane(const vector<int>& array) {
 Result result;
 result.maxSum = array[0];
 result.startIndex = 0;
 result.endIndex = 0;
 int maxCurrent = array[0];
 int maxGlobal = array[0];
 int tempStart = 0;
 for (size_t i = 1; i < array.size(); ++i) {
 if (array[i] > maxCurrent + array[i]) {
 maxCurrent = array[i];
```

```
tempStart = i;
} else {
maxCurrent += array[i];
}
if (maxCurrent > maxGlobal) {
maxGlobal = maxCurrent;
result.startIndex = tempStart;
result.endIndex = i;
}
}
result.maxSum = maxGlobal;
return result;
}
int main() {
int numElements;
cout << "Enter number of elements: ";
cin >> numElements;
vector<int> array(numElements);
cout << "Enter the elements: ";
for (int i = 0; i < numElements; ++i) {
cin >> array[i];
}
auto startTime = chrono::high_resolution_clock::now();
Result result = maxSubarrayKadane(array);
auto endTime = chrono::high_resolution_clock::now();
chrono::duration<double, micro> duration = endTime - startTime;
cout << "Kadane's Algorithm: " << result.maxSum << endl;
cout << "Start index: " << result.startIndex << endl;
cout << "End index: " << result.endIndex << endl;
cout << "Time taken: " << duration.count() << " microseconds" << endl;
return 0;
```

}