BILKENT UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

CS 201: FUNDAMENTAL STRUCTURES OF COMPUTER SCIENCE I

HOMEWORK 2 DISCUSSIONS

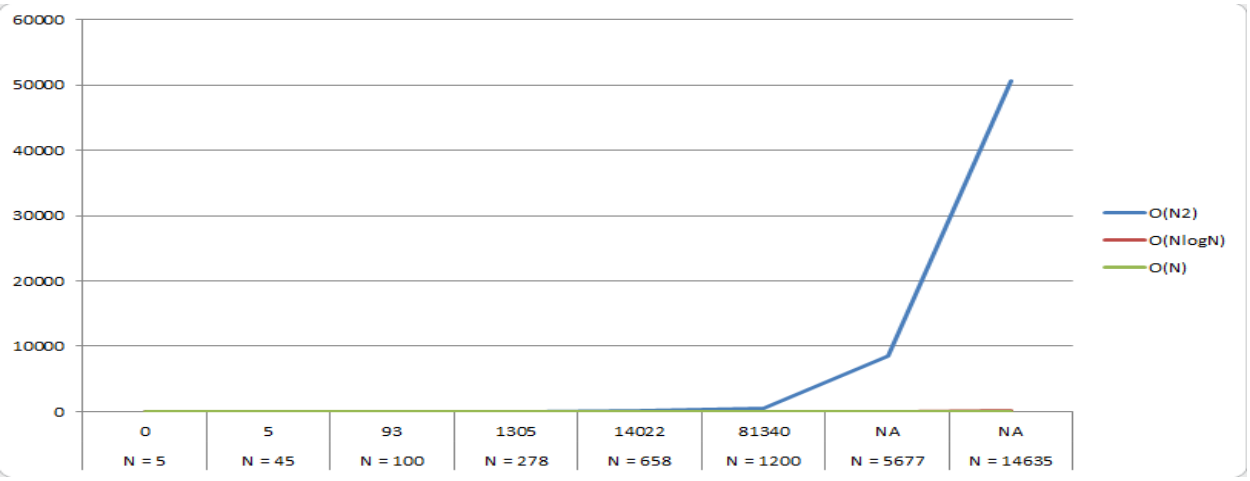
&

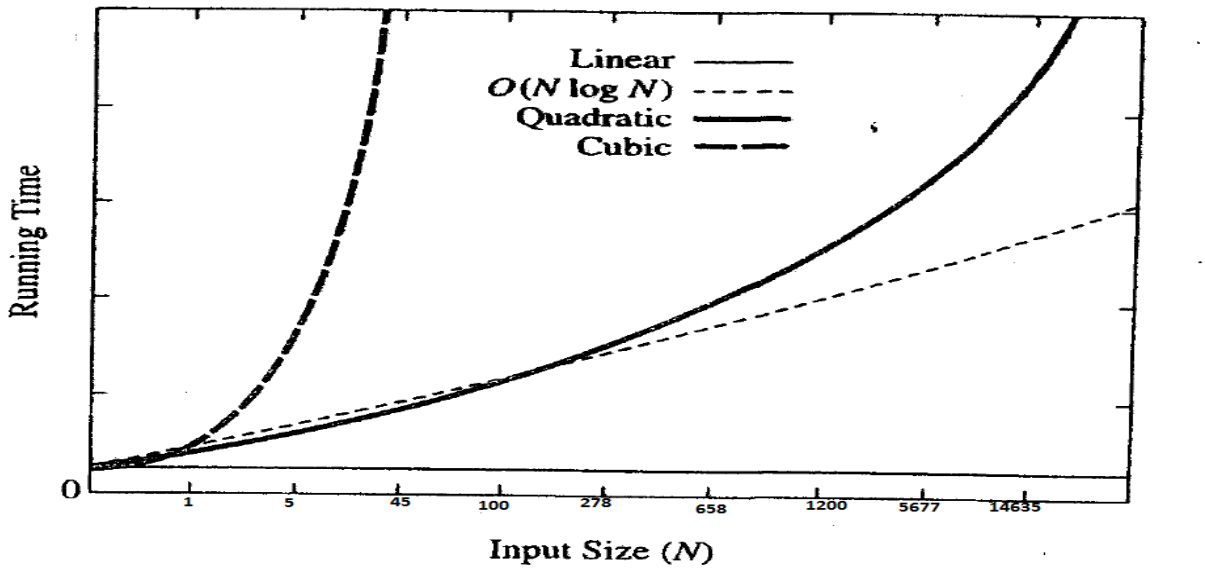
C++ CODES

BERAT BİÇER - 21503050 - SECTION 2

EXPERIMENT 1

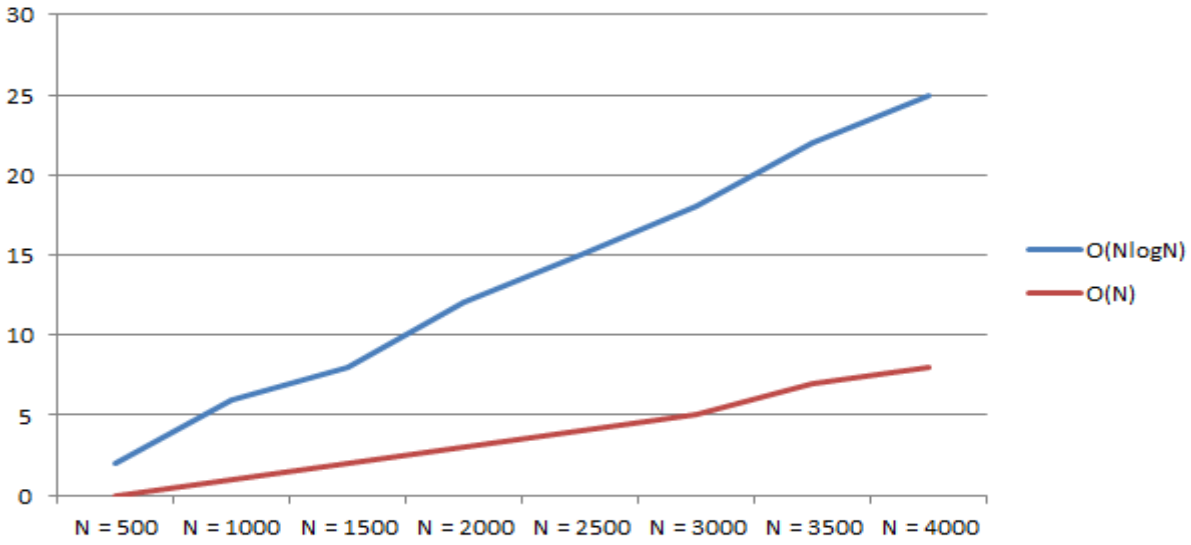
ARRAY SIZE\ALGORITHM COMPLEXITY	O(N^3)	O(N^2)	O(NlogN)	O(N)
N = 5	0	0	0	0
N = 45	5	1	0	0
N = 100	93	3	0	0
N = 278	1305	28	1	0
N = 658	14022	187	3	1
N = 1200	81340	483	7	1
N = 5677	NA	8511	32	3
N = 14635	NA	50563	73	23

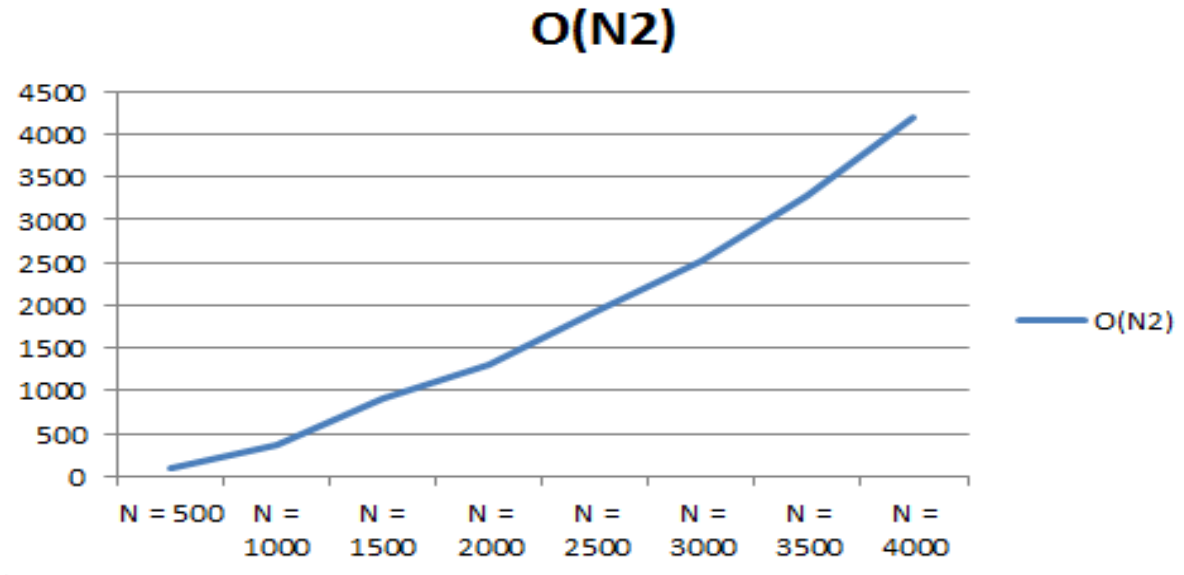




EXPERIMENT 2

ARRAY SIZE\ALGORITHM COMPLEXITY	$O(N^3)$	$O(N^2)$	O(NlogN)	O(N)
N = 500	6970	84	2	0
N = 1000	48904	355	6	1
N = 1500	161777	921	8	2
N = 2000	NA	1314	12	3
N = 2500	NA	1921	15	4
N = 3000	NA	2517	18	5
N = 3500	NA	3274	22	7
N = 4000	NA	4205	25	8





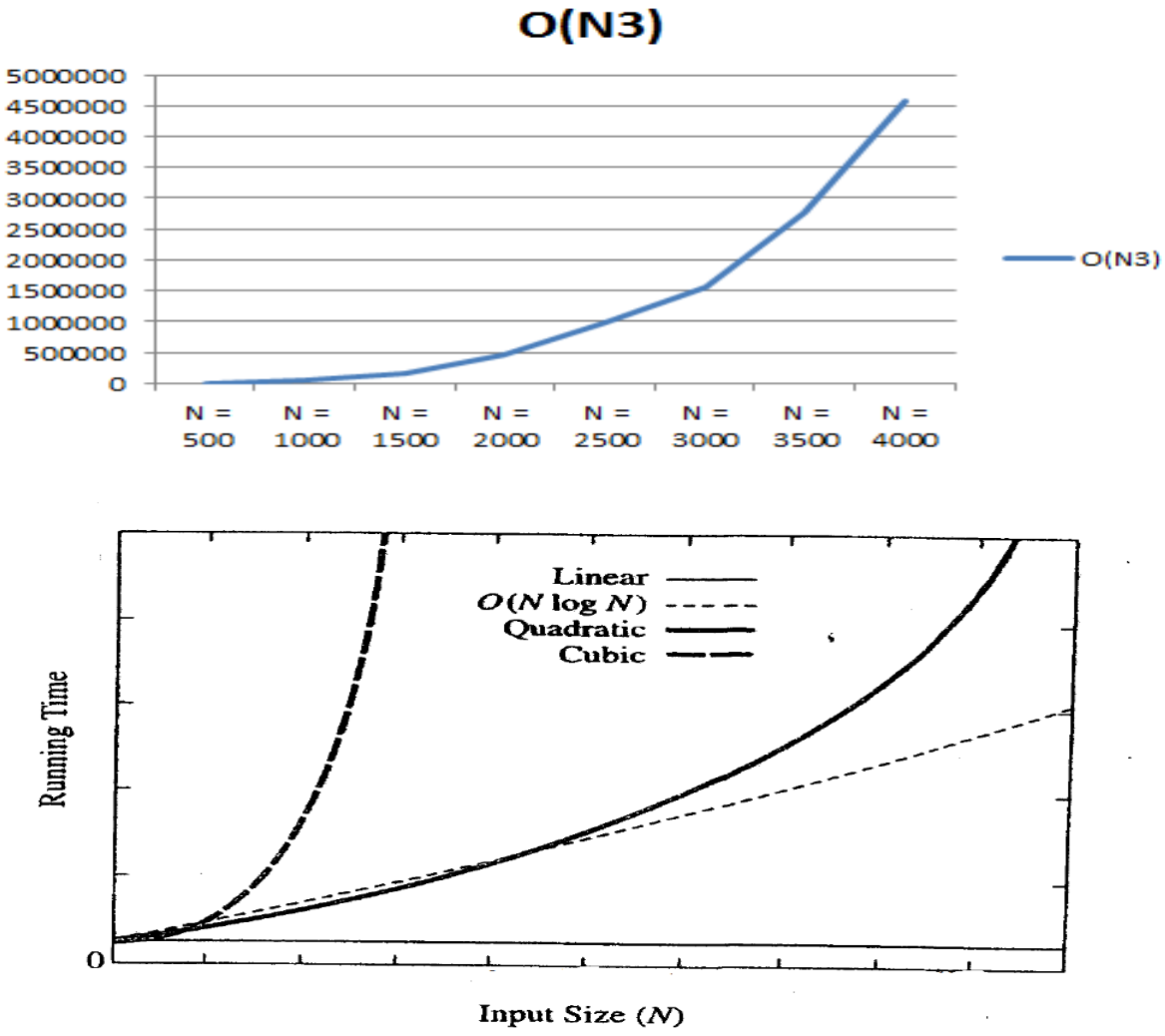


Figure 1: Expected Graph

DISCUSSION

In both cases as the array size gets bigger, I observed that time required to run a specific algorithm becomes larger. Since algorithms have different rate of growth, their growth are different as expected. To be more specific,

- We know algorithm 1 has growth rate O(n^3) and algorithm 4 has growth rate O(n). From the graphs above, we see that our theoretical calculations are correct since algorithm 1 works the slowest whereas algorithm 4 works the fastest.
- We can observe that

    - Algorithm 3 and 4 are quite close to each other eventhough size of the input N grows larger. This implies that these algorithms work similarly and their efficiency are similar as well.
    - Eventhough algorithm 1 and 2 are similar in theory that is, their growth rate is relatively close to each other, in practice they are quite different: algorithm 1 becomes larger faster that algorithm 2 in very small time intervals.
    - For large data, it's wise to use the most efficient algorithm to work with because it saves time and resources. However, for small data, the difference between these algorithms isn't quite clear and any one of them can be chosen to work with.
- It is also worth mentioning that for different computer systems and CPU's, these results may vary and therefore it's important to note my computer's specifications: Intel Core i7-4510u CPU @ 2.00-2.60 GHz, 8 GB RAM with Windows 7 64 bit OS.

C++ CODES

MAIN – 1:

```
#include <iostream>

#include <ctime>

#include<vector>

using namespace std;


int maxSubSum1(const vector<int> & a);

int maxSubSum2(const vector<int> & a);

int maxSumRec(const vector<int> & a, int left, int right);

int max3(int a, int b, int c);

int maxSubSum3(const vector<int> & a);

int maxSubSum4(const vector<int> & a);
```

```
int maxSubSum1(const vector<int> & a) {
        int maxSum = 0;

        for(size_t i = 0; i < a.size(); i++)
                for (size_t j = i; j < a.size(); j++) {
                        int thisSum = 0;

                        for (size_t k = i; k <= j; k++)
                                thisSum += a[k];

                        if (thisSum > maxSum)
                                maxSum = thisSum;
                }
        return maxSum;
}

int maxSubSum2(const vector<int> & a) {
        int maxSum = 0;

        for (size_t i = 0; i < a.size(); i++) {
                int thisSum = 0;

                for (size_t j = i; j < a.size(); j++) {
                        thisSum += a[j];
```

```
                if (thisSum > maxSum)
                        maxSum = thisSum;
            }
        }
        return maxSum;
}

int maxSumRec(const vector<int> & a, int left, int right) {
        if (left == right)
            if(a[left] > 0)
                return a[left];
            else
                return 0;

        int center = (left + right) / 2;
        int maxLeftSum = maxSumRec(a, left, center);
        int maxRightSum = maxSumRec(a, center + 1, right);

        int maxLeftBorderSum = 0, leftBorderSum = 0;
        for (int i = center; i >= left; i--) {
            leftBorderSum += a[i];

            if (leftBorderSum > maxLeftBorderSum)
                maxLeftBorderSum = leftBorderSum;
        }
```

```
	int maxRightBorderSum = 0, rightBorderSum = 0;
	for (int j = center + 1; j <= right; j++) {
		rightBorderSum += a[j];

		if (rightBorderSum > maxRightBorderSum)
			maxRightBorderSum = rightBorderSum;
	}

	return max3(maxLeftSum, maxRightSum, maxLeftBorderSum + maxRightBorderSum);
}

int max3(int a, int b, int c) {
	int max = 0;
	if (a > b)
		max = a;
	else
		max = b;

	if (c > max)
		max = c;
	else
		max = max;
	return max;
}
```

```
int maxSubSum3(const vector<int> & a) {
	return maxSumRec(a, 0, a.size() - 1);
}

int maxSubSum4(const vector<int> & a) {
	int maxSum = 0, thisSum = 0;

	for (size_t j = 0; j < a.size(); j++) {
		thisSum += a[j];

		if (thisSum > maxSum)
			maxSum = thisSum;
		else if (thisSum < 0)
			thisSum = 0;
	}
	return maxSum;
}

int main() {
	int base = 0;
	//int a[5];
	//int b[45];
	//int c[100];
	//int d[278];
```

```
//int e[658];

//int f[1200];

//int g[5677];

int h[14635];



/*for (int i = 0; i < 5; i++) {

	a[i] = ((base*base + 7) / 5) + 8;

	base++;

}*/


/*for (int i = 0; i < 45; i++) {

	b[i] = ((base*base + 7) / 5) + 8;

	base++;

}*/


/*for (int i = 0; i < 100; i++) {

	c[i] = ((base*base + 7) / 5) + 8;

	base++;

}*/


/*for (int i = 0; i < 278; i++) {

	d[i] = ((base*base + 7) / 5) + 8;

	base++;

}*/
```

```
/*for (int i = 0; i < 658; i++) {
		e[i] = ((base*base + 7) / 5) + 8;
		base++;
}*/

/*for (int i = 0; i < 1200; i++) {
		f[i] = ((base*base + 7) / 5) + 8;
		base++;
}*/

/*for (int i = 0; i < 5677; i++) {
		g[i] = ((base*base + 7) / 5) + 8;
		base++;
}*/

for (int i = 0; i < 14635; i++) {
		h[i] = ((base*base + 7) / 5) + 8;
		base++;
}

//vector<int> v(a, a + sizeof a / sizeof a[0]);
//vector<int> v(b, b + sizeof b / sizeof b[0]);
//vector<int> v(c, c + sizeof c / sizeof c[0]);
//vector<int> v(d, d + sizeof d / sizeof d[0]);
```

```
        //vector<int> v(e, e + sizeof e / sizeof e[0]);
        //vector<int> v(f, f + sizeof f / sizeof f[0]);
        //vector<int> v(g, g + sizeof g / sizeof g[0]);
        vector<int> v(h, h + sizeof h / sizeof h[0]);



        // Store the starting time
        double duration;
        clock_t startTime = clock();
        // Code block

        int x = maxSubSum4(v);

        //Compute the number of milliseconds that passed since the starting time
        duration = 1000 * double(clock() - startTime) / CLOCKS_PER_SEC;
        cout << "Execution took " << duration << " milliseconds." << endl;

        int q;
        cin >> q;
}
```

MAIN – 2:

```
#include <iostream>
#include <ctime>
#include<vector>
using namespace std;
```

```
int maxSubSum1(const vector<int> & a);

int maxSubSum2(const vector<int> & a);

int maxSumRec(const vector<int> & a, int left, int right);

int max3(int a, int b, int c);

int maxSubSum3(const vector<int> & a);

int maxSubSum4(const vector<int> & a);


int maxSubSum1(const vector<int> & a) {

	int maxSum = 0;


	for(size_t i = 0; i < a.size(); i++)

		for (size_t j = i; j < a.size(); j++) {

			int thisSum = 0;


			for (size_t k = i; k <= j; k++)

				thisSum += a[k];


			if (thisSum > maxSum)

				maxSum = thisSum;

		}

	return maxSum;

}


int maxSubSum2(const vector<int> & a) {
```

```
	int maxSum = 0;

	for (size_t i = 0; i < a.size(); i++) {
		int thisSum = 0;

		for (size_t j = i; j < a.size(); j++) {
			thisSum += a[j];

			if (thisSum > maxSum)
				maxSum = thisSum;
		}
	}
	return maxSum;
}

int maxSumRec(const vector<int> & a, int left, int right) {
	if (left == right)
		if(a[left] > 0)
			return a[left];
		else
			return 0;

	int center = (left + right) / 2;
	int maxLeftSum = maxSumRec(a, left, center);
	int maxRightSum = maxSumRec(a, center + 1, right);
```

```
	int maxLeftBorderSum = 0, leftBorderSum = 0;
	for (int i = center; i >= left; i--) {
		leftBorderSum += a[i];

		if (leftBorderSum > maxLeftBorderSum)
			maxLeftBorderSum = leftBorderSum;
	}

	int maxRightBorderSum = 0, rightBorderSum = 0;
	for (int j = center + 1; j <= right; j++) {
		rightBorderSum += a[j];

		if (rightBorderSum > maxRightBorderSum)
			maxRightBorderSum = rightBorderSum;
	}

	return max3(maxLeftSum, maxRightSum, maxLeftBorderSum + maxRightBorderSum);
}

int max3(int a, int b, int c) {
	int max = 0;
	if (a > b)
		max = a;
	else
```

```
            max = b;


        if (c > max)
            max = c;
        else
            max = max;
        return max;
}


int maxSubSum3(const vector<int> & a) {
        return maxSumRec(a, 0, a.size() - 1);
}


int maxSubSum4(const vector<int> & a) {
        int maxSum = 0, thisSum = 0;


        for (size_t j = 0; j < a.size(); j++) {
            thisSum += a[j];


            if (thisSum > maxSum)
                maxSum = thisSum;
            else if (thisSum < 0)
                thisSum = 0;
        }
        return maxSum;
```

```
}

int main() {
	int base = 0;
	int a[500];
	int b[1000];
	int c[1500];
	int d[2000];
	int e[2500];
	int f[3000];
	int g[3500];
	int h[4000];


	for (int i = 0; i < 500; i++) {
		a[i] = ((base*base + 7) / 5) + 8;
		base++;
	}

	for (int i = 0; i < 1000; i++) {
		b[i] = ((base*base + 7) / 5) + 8;
		base++;
	}

	for (int i = 0; i < 1500; i++) {
```

```
            c[i] = ((base*base + 7) / 5) + 8;
            base++;
}

for (int i = 0; i < 2000; i++) {
            d[i] = ((base*base + 7) / 5) + 8;
            base++;
}

for (int i = 0; i < 2500; i++) {
            e[i] = ((base*base + 7) / 5) + 8;
            base++;
}

for (int i = 0; i < 3000; i++) {
            f[i] = ((base*base + 7) / 5) + 8;
            base++;
}

for (int i = 0; i < 3500; i++) {
            g[i] = ((base*base + 7) / 5) + 8;
            base++;
}

for (int i = 0; i < 4000; i++) {
```

```
        h[i] = ((base*base + 7) / 5) + 8;
        base++;
}

//vector<int> v(a, a + sizeof a / sizeof a[0]);
//vector<int> v(b, b + sizeof b / sizeof b[0]);
//vector<int> v(c, c + sizeof c / sizeof c[0]);
//vector<int> v(d, d + sizeof d / sizeof d[0]);
//vector<int> v(e, e + sizeof e / sizeof e[0]);
//vector<int> v(f, f + sizeof f / sizeof f[0]);
//vector<int> v(g, g + sizeof g / sizeof g[0]);
vector<int> v(h, h + sizeof h / sizeof h[0]);


// Store the starting time
double duration;
clock_t startTime = clock();
// Code block

int x = maxSubSum4(v);

//Compute the number of milliseconds that passed since the starting time
duration = 1000 * double(clock() - startTime) / CLOCKS_PER_SEC;
cout << "Execution took " << duration << " milliseconds." << endl;
```

```
        int q;
        cin >> q;
}
```