A Retrospective On:
Brute-Force Algorithms on the Minimum Enclosing Circle Problem

Table 1							
		Center coordinates	Center coordinates	Radius	Radius	Surface	Surface
File Name	Number of points	Approach 1	Approach 2	Approach 1	Approach 2	Approach 1	Approach 2
Circles_10.txt	10	(-0.153928, 0.0605847)	(0.1, 0.2)	0.86686	0.75	2.36074	1.76715
Circles_20.txt	20	(0.243445, -0.196219)	(0.2, -0.2)	0.826015	0.78	2.14351	1.91134
Circles_100.txt	100	(0.11895, 0.126194)	(0.17626, 0.0819885)	0.757014	0.773858	1.80035	1.88136
Circles_400.txt	400	(0.256797, 0.198627)	(0.208451, 0.195852)	0.628835	0.636879	1.24229	1.27428
Circles_800.txt	800	(0.0724267, 0.00975774)	(0.0625, -1.88371e-09)	0.917721	0.90625	2.64589	2.58016

Circles_1000.txt	1000	(0.270352, 0.0147778)	(0.283085, 0.0172973)	0.887642	0.885074	2.47529	2.46099
Circles_2000.txt	2000	(0.0981695, -0.00247526)	(0.09375, -7.07801e-10)	0.910718	0.90625	2.60566	2.58016
Circles_4000.txt	4000	(0.208318, 0.0135393)	(0.20846, 0.0136782)	0.898361	0.894352	2.53543	2.51285
Circles_20000.txt	20000	Valid Circles couldn't be found of this dataset in time as the runtime required to compute the minimum enclosed circle for datasets > 20000 require multiple years to execute. This is due to the growth trend of n^2 and n^3 time complexities of pairs of points & triplets of points respectively.					
Circles_40000.txt	40000						
Circles_100000.txt	100000						
Circles_400000.txt	400000						
Circles_800000.txt	800000						

Pair-based Approach:

- Computes the MEC using pairs of points.
-
- Runtime complexity: O(n^2).
-
- Parallelized using computer's optimal threads.

Triplet-based Approach:

- Computes the MEC using triplets of points.
-
- Runtime complexity: O(n^3).
-
- Parallelized using computer's optimal threads.

Consistency of Results

The results of the two approaches are not entirely consistent. While the center coordinates, radii, and surface areas are extremely close, they are not identical. This indicates that the two approaches may produce slightly different results due to their algorithmic differences.

Why Are The Results Inconsistent?

Pair-based Approach: This approach computes the MEC using pairs of points. It is faster with a time complexity of O(n^2) but may not always produce the smallest possible circle.

Triplet-based Approach: This approach computes the MEC using triplets of points. It is more accurate but slower with a time complexity of O(n^3). It is more likely to find the true smallest enclosing circle but becomes impractical for large datasets which was experienced with the bottlenecked performance from datasets larger than 4000 points.

The **pair-based approach** may use approximations to speed up computation, leading to slightly less accurate results.

The **triplet-based approach** aims for an exact solution by considering all possible triplets of points, which is more computationally expensive but more accurate.

For datasets with irregular point distributions, the **pair-based approach** may produce a larger circle than necessary, while the **triplet-based approach** is more likely to find the optimal circle.

Table 2							
		Execution Time (in ms)			Asymptotic Complexity as a function of n		
File Name	Number of points	Bruteforce1	Bruteforce2	Bruteforce1/Bruteforce2	Bruteforce1	Bruteforce2	Bruteforce1/Bruteforce2
Circles_10.txt	10	0	0	0/0 (Undef)	O(N^2)	O(N^3)	O(N^3)
Circles_20.txt	20	0	0	0/0(Undef)	O(N^2)	O(N^3)	O(N^3)
Circles_100.txt	100	1	6	0.16666667	O(N^2)	O(N^3)	O(N^3)
Circles_400.txt	400	7	412	0.01699029126	O(N^2)	O(N^3)	O(N^3)

Circles_800.txt	800	39	3082	0.0126541207	O(N^2)	O(N^3)	O(N^3)
Circles_1000.txt	1000	48	5276	0.00909780136	O(N^2)	O(N^3)	O(N^3)
Circles_2000.txt	2000	369	43647	0.00845418929	O(N^2)	O(N^3)	O(N^3)
Circles_4000.txt	4000	2257	306665	0.0073598226	O(N^2)	O(N^3)	O(N^3)
Circles_20000.txt	20000	~36900	~43647000	8.4541892913602309437074713038697e-4	O(N^2)	O(N^3)	O(N^3)
Circles_40000.txt	40000	~147600	~2560000000	0.00005765625	O(N^2)	O(N^3)	O(N^3)
Circles_100000.txt	100000	~922500	~1091175000	8.4541892913602309437074713038697e-4	O(N^2)	O(N^3)	O(N^3)
Circles_400000.txt	400000	~14760000	~17458800000	8.4541892913602309437074713038697e-4	O(N^2)	O(N^3)	O(N^3)
Circles_800000.txt	800000	~59040000	~139670400000	4.2270946456801154718537356519348e-4	O(N^2)	O(N^3)	O(N^3)

Bruteforce1 (Pair-based Approach)
Theoretical Complexity: O(n^2).

The execution time grows quadratically with the number of points, consistent with the $O(n^2)$ complexity.

For example, the ratio of execution times for 40000 points (147600 ms) and 4000 points (2257 ms) is approximately 65.3, which is close to the expected ratio of $40000^2 / 4000^2 = 100$.

Bruteforce2 (Triplet-based Approach):

Theoretical Complexity: O(n^3)

The execution time grows cubically with the number of points, consistent with the $O(n^3)$ complexity.

For example, the ratio of execution times for 1000 points (5276 ms) and 100 points (6 ms) is approximately 879, which is close to the expected ratio of $1000^3 / 100^3 = 1000$.

The error between the actual ratios and theoretical ratios can be explained by the bottleneck of non-ideal computer hardware. The same error can also be explained by the varying increase of points from one dataset to another. For example, 1000 to 4000 points is a different rate of increase than 4000 to 40000.

Dataset	Points	Ratio of Runtimes (Triplet/Pair)	Expected Asymptotic Ratio O(n) *(n^3/n^2)*	Comparison Of Ratio of Runtimes to Expected Asymptotic Ratio

Circles_10.txt	**10**	**Undefined**	**10**	**N/A**
Circles_20.txt	**20**	**Undefined**	**20**	**N/A**
Circles_100.txt	**100**	**6.0**	**100**	**6.0 < 100**
Circles_400.txt	**400**	**58.86**	**400**	**58.86 < 400**
Circles_800.txt	**800**	**79.03**	**800**	**79.03 < 800**
Circles_1000.txt	**1000**	**109.92**	**1000**	**109.92 < 1000**
Circles_2000.txt	**2000**	**118.28**	**2000**	**118.28 < 2000**
Circles_4000.txt	**4000**	**135.87**	**4000**	**135.87 < 4000**
Circles_20000.txt	**20000**	**1182.82**	**20000**	**1182.82 < 20000**
Circles_40000.txt	**40000**	**17347.83**	**40000**	**17347.83 < 40000**
Circles_100000.txt	**100000**	**1182.82**	**100000**	**1182.82 < 100000**
Circles_400000.txt	**400000**	**1182.82**	**400000**	**1182.82 < 400000**
Circles_800000.txt	**800000**	**2365.64**	**800000**	**2365.64 < 800000**

Conclusion

The pair-based approach would be more optimal for smaller, more regular spread of points datasets while the triplet-based approach is more optimal for a more precise MEC at the cost of speed and memory.

The key point our group has taken away from this programming assignment

is the aspect of how significant different algorithms could optimize a solution to a problem.

In this case, we have tried a large number of optimization techniques to reduce runtime and resource expenditure but it is not possible with these two approaches. A more optimal approach would be Welzl's Algorithm with a time complexity of O(n) which would scale linearly based on our provided datasets.

All in all, the restrictions to only these two algorithms inspired a great motivation to create more efficient algorithms suited to every particular problem.