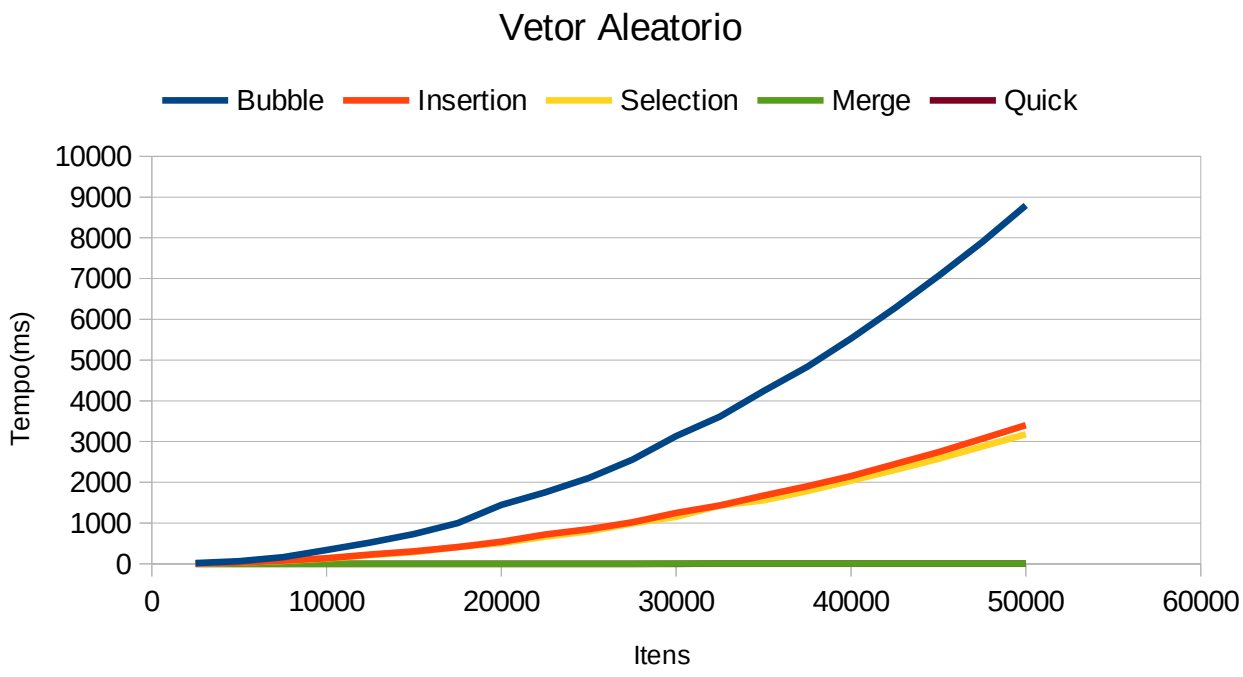
Nesse trabalho foi possível observar os diferentes tipos de ordenação, seus tempos de execução, seus piores casos e concluir qual algoritmo é melhor em cada situação.

Nos testes foram utilizados vetores de 2500 elementos, o proximo de 5000, e depois de 7500, até um vetor limite de 50000 elementos, no total 20 vetores diferentes para cada caso.

O primeiro caso envolveu ordenar um vetor aleatório, nesse caso os algoritmos de ordenação elementares(Selection, Insertion, Bubble) praticamente tiverem a complexidade constante de n^2. Já os algoritmos mais eficientes de ordenação(Quick,Merge) manteram a complexidade em nlog(n) basicamente em todos os testes.

Vetor Aleatorio

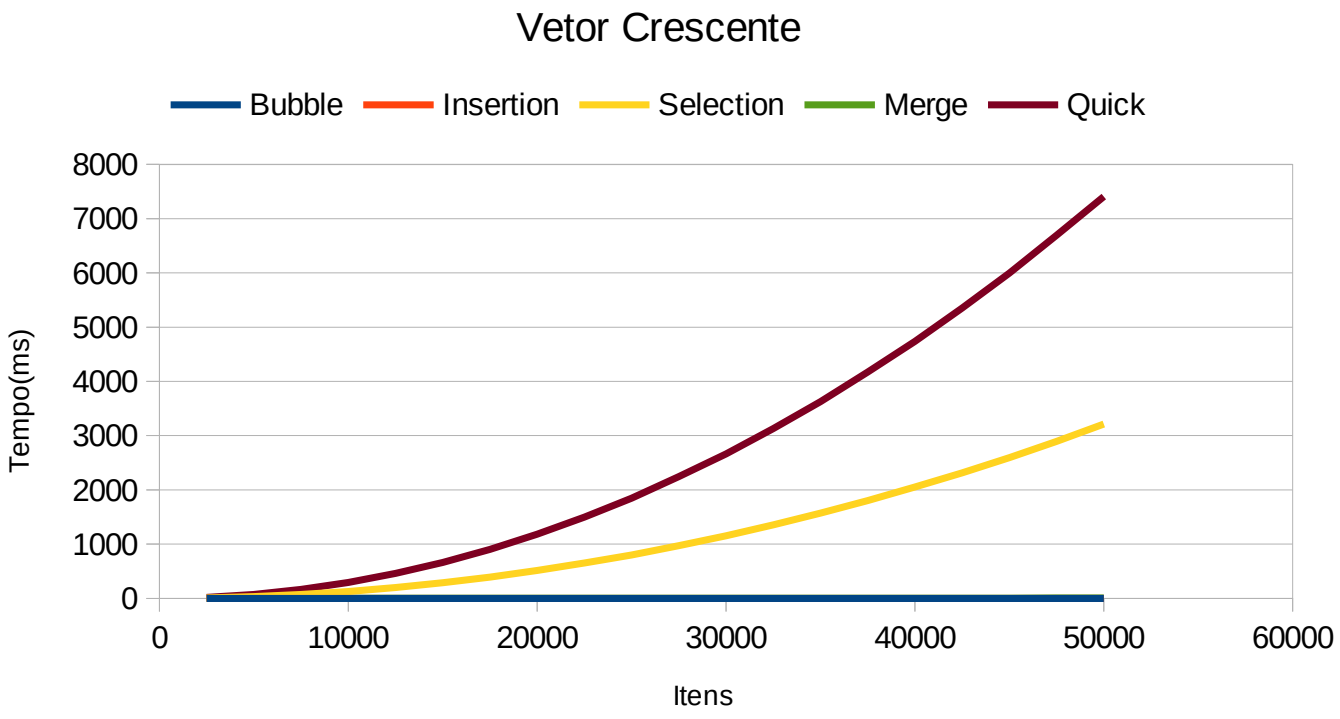
Itens	Bubble	Insertion	Selection	Merge	Quick
2500	21	9	8	0	0
5000	68	34	32	0	0
7500	166	79	74	1	1
10000	338	141	135	1	1
12500	528	228	220	2	2
15000	735	308	287	2	2
17500	1001	415	409	3	2
20000	1444	546	510	3	2
22500	1752	723	668	3	3
25000	2105	854	798	4	3
27500	2561	1023	996	4	4
30000	3132	1247	1153	5	4
32500	3610	1435	1432	6	5
35000	4241	1672	1560	6	5
37500	4837	1906	1788	6	6
40000	5530	2153	2040	7	6
42500	6274	2450	2300	7	6
45000	7066	2741	2575	8	7
47500	7894	3071	2876	8	7
50000	8790	3406	3182	9	8
Tempo Total	109 seg				



No caso de um vetor crescente, alguns algorimos elementares foram mais eficientes que os mais complexos. Bubble e Insertion tiveram o tempo nulo em todos os casos independente do tamanho do vetor, isso porque o melhor caso para esses algoritmos é esse, já o Seletion teve um desemepenho um pouco pior. Mas o Quick teve um desempenho muito ruim nesse caso, pois é um dos piores casos pra ele, tornando a complexidade n^2.

Vetor Crescente

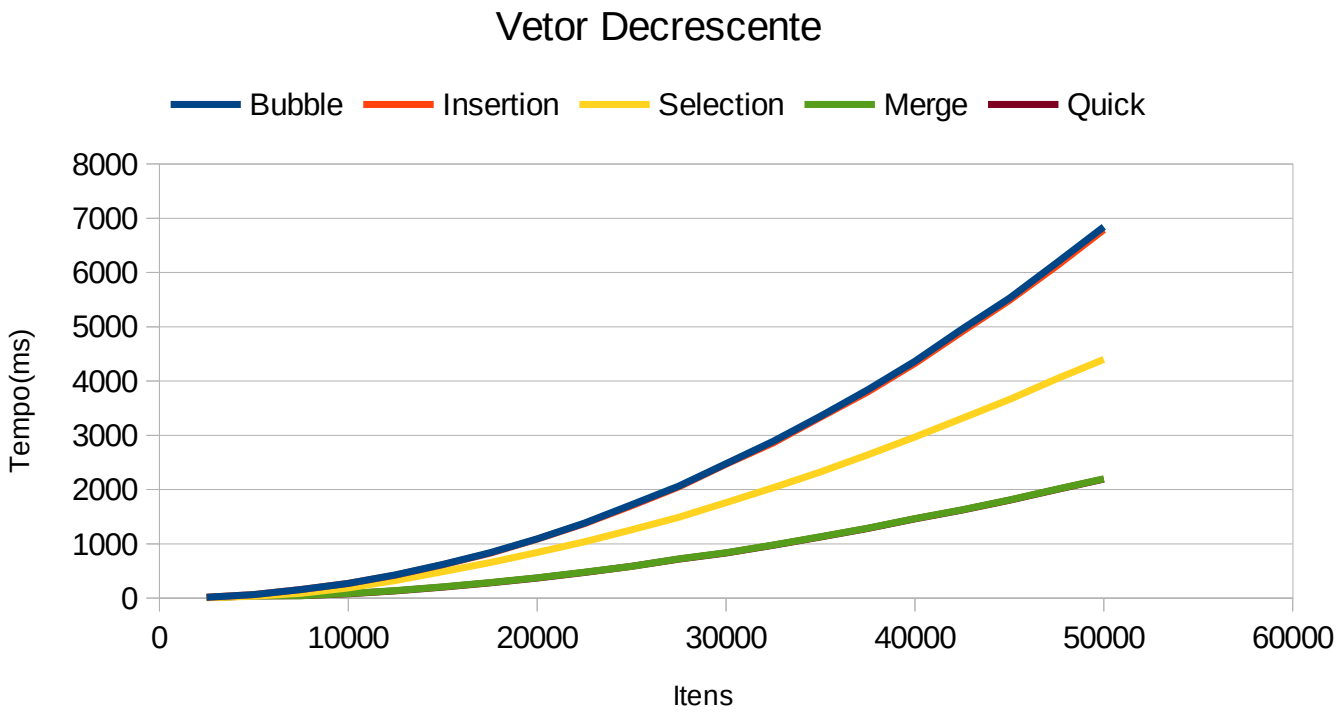
Itens	Bubble	Insertion	Selection	Merge	Quick
2500	0	0	7	0	18
5000	0	0	31	0	73
7500	0	0	71	0	164
10000	0	0	127	1	293
12500	0	0	199	1	459
15000	0	0	287	1	663
17500	0	0	392	1	904
20000	0	0	512	2	1182
22500	0	0	651	2	1496
25000	0	0	800	2	1848
27500	0	0	969	2	2242
30000	0	0	1153	3	2662
32500	0	0	1353	3	3127
35000	0	0	1571	3	3628
37500	0	0	1803	4	4169
40000	0	0	2052	4	4736
42500	0	0	2314	4	5351
45000	0	0	2594	4	5995
47500	0	0	2892	6	6693
50000	0	0	3212	5	7406
Tempo Total	76 seg				



Em um vetor decrescente, os algoritmos elementares tiverem um desempenho ruim de n^2. O Merge independe de como o vetor está ordenado pois sempre fará suas divisões com complexidade nlog(n). E o Quick foi ruim novamente pois é mais um pior caso, juntamente com o vetor crescente, tornando-o n^2 nessa situação.

Vetor Decrescente

Itens	Bubble	Insertion	Selection	Merge	Quick
2500	16	16	7	10	10
5000	68	67	33	42	42
7500	156	156	89	45	44
10000	272	271	188	83	82
12500	427	422	324	137	136
15000	619	614	486	206	205
17500	838	828	653	285	283
20000	1092	1088	842	373	370
22500	1384	1370	1036	478	476
25000	1720	1700	1257	586	583
27500	2065	2050	1492	721	718
30000	2475	2465	1757	834	831
32500	2891	2862	2032	978	974
35000	3350	3330	2324	1130	1125
37500	3838	3801	2641	1286	1281
40000	4364	4326	2970	1464	1459
42500	4960	4910	3311	1628	1623
45000	5530	5488	3661	1811	1806
47500	6180	6117	4038	2006	2000
50000	6841	6789	4400	2198	2192
Tempo Total	147 seg				



Conclusões:

- ✔ Em vetores aleatórios, o Quick é melhor que qualquer um dos outros, com um desempenho médio de nlog(n) e na prática,mais rápido que o Merge. Já os elementares tem na média o desempenho de n^2.

- ✔ Num vetor crescente, os campeões são os elementares com a complexidade de n. Já o Quick fica em último lugar por ser seu pior caso.

- ✔ Já em um vetor decrescente, o vencedor é o Merge que mantém constante a sua complexidade de nlog(n) em qualquer caso, já os outros todos tem a complexidade de n^2.