UNIVERSIDADE DE SÃO PAULO

INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

INTRODUÇÃO À CIÊNCIA DE COMPUTAÇÃO II

LEANDRO SATOSHI DE SIQUEIRA

GABRIEL SILVA FONTES

ALGORITMOS DE ORDENAÇÃO

SÃO CARLOS - SÃO PAULO

2018

LEANDRO SATOSHI DE SIQUEIRA
GABRIEL SILVA FONTES

ALGORITMOS DE ORDENAÇÃO

Trabalho apresentado ao Curso de Bacharelado em Sistema de Informação do Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo.

Orientador: Adenilso da Silva Simão

SÃO CARLOS - SÃO PAULO
2018

SUMÁRIO

1 INTRODUÇÃO

Este trabalho refere-se aos algoritmos de ordenação estudados e implementados durante as aulas da disciplina de Introdução à ciência de computação II.

Dividido em 3 partes, primeiramente serão abordados todos os algoritmos aqui analisados, em seguida os casos testes utilizados, e por fim o resultado de performance de cada algoritmo.

1.1 OBJETIVOS

O objetivo é analisar e comparar o desempenho de diferentes algoritmos de ordenação em cenários diversos, variando o tamanho e o tipo das entradas.

2 METODOLOGIA

A metodologia utilizada foi uma simples implementação dos algoritmos utilizando pequenas modificações para o obter o número de comparações e o número de atribuições envolvendo elementos do vetor ao termino da execução.

Cada cenário, variando o número de elementos do vetor e o tipo destes, foi testado ao menos cinco vezes, e os resultados aqui apresentados são uma média dos obtidos durante a realização desta análise.

Os resultados de cada algoritmo nos casos analisados estão explicitados posteriormente nas formas de gráficos e tabelas para uma melhor comparação e exibição.

3 ALGORITMOS

Os algoritmos aqui utilizados foram os oito primeiros a serem estudados e implementados nas aulas de introdução à ciência da computação II, sendo destes, cinco do tipo simples ou intuitivos, e outros três mais complexos e sofisticados.

Abaixo estão um breve resumo de cada algoritmo bem como seus respectivos custos de tempo e memória na notação de O-grande(mais comumente conhecido como Big-O)

Mais a frente os Big-O's dos algoritmos serão verificados com os dados dos testes.

3.1 BUBBLE SORT

O algoritmo de bolha(ordenação por flutuação), mais conhecido como Bubble sort, é o algoritmo de ordenação mais simples de todos, e funciona trocando elementos adjacentes repetidamente de estiverem na ordem errada. Sempre iniciando no primeiro elemento do vetor e indo até o ultimo. (GEEKS FOR GEEKS, a)
Pior caso de tempo:O(n^2)
Melhor caso de tempo:O(n)
Gasto auxiliar de memoria:O(1)

3.2 BUBBLE SORT COM SENTINELA

Bubble sort com sentinela é uma otimização do algoritmo citado anteriormente, e funciona adicionando um elemento sentinela para verificar onde houve a primeira interação de troca, assim sabe-se que até aquele ponto o vetor ja esta ordenado e não é necessario verificar todos elementos anteriores a ele.
Pior caso de tempo:O(n^2)
Melhor caso de tempo:O(n)
Gasto auxiliar de memoria:O(1)

3.3 BUBBLE SORT COCKTAIL SHAKER

Assim como a sentinela, Bubble com coquetel, conhecido também como cocktail sort, é uma variação/otimização do Bubble. Enquanto o tradicional sempre atravessa o vetor da esquerda para direita, o cocktail sort o faz em ambas as direções alternadamente.
Pior caso de tempo:O(n^2)
Melhor caso de tempo:O(n)
Gasto auxiliar de memoria:O(1)

3.4 INSERCTION SORT

Insertion sort, ou ordenação por seleção, é um algoritmo que funciona da maneira que ordenamos cartas na mão, ou seja, pega um elemento e o insere na sua posição correta, arrastando outros para abrir espaço. (GEEKS FOR GEEKS, b)
Pior caso de tempo:O(n^2)
Melhor caso de tempo:O(n)
Gasto auxiliar de memoria:O(1)

3.5 SELECTION SORT

Selection sort, ou seleção, funciona buscando o menor elemento e o colocando no inicio do vetor não ordenado, assim ele mantém dois subvetores, um de ordenados e outro de não-ordenados.
Pior caso de tempo:O(n^2)
Melhor caso de tempo:O(n^2)
Gasto auxiliar de memoria:O(1)

3.6 MERGE SORT

Merge Sort, ou intercalação, funciona dividindo o problema em problemas menores, trabalhando com duas funções, uma que divide um vetor em dois menores e uma que intercala dois vetores ordenados em um unico. Como um vetor unitario é sempre ordenado, chamando recursivamente essas funções é possivel ordenar o vetor.

Pior caso de tempo:O($n\log(n)$)
Melhor caso de tempo:O($n\log(n)$)
Gasto auxiliar de memoria:O(n)

3.7 HEAP SORT

Heap sort é um algoritmo de ordenação baseado na estrutura binaria de Heap, uma vez com essa estrutura, é semelhante ao selection sort em encontrar o maior elemento e ordenar.
Pior caso de tempo:O($n\log(n)$)
Melhor caso de tempo:O($n\log(n)$)
Gasto auxiliar de memoria:O(1)

3.8 QUICK SORT

Quick sort, semelhante ao merge, divide o problema em parcelas menores, pegando um elemento como pivô, faz todos antes dele serem menores, e todos depois dele serem maiores, então repete o processo recursivamente.
Pior caso de tempo:O(n^2)
Melhor caso de tempo:O($n\log(n)$)
Gasto auxiliar de memoria:O(1)

4 CENÁRIOS E CASOS TESTE

Para a avaliação dos algoritmos, serão utilizados varios cenários com diversos casos teste. Primeiramente serão utilizados vetores com 10^k elementos, sendo que $2 \leqslant k \leqslant 6$. Além disso, os elementos desses vetores poderão estar distribuidos de quatro formas diferentes, sendo elas: Completamente aleatória, Quase ordenada, Quase inversamente ordenada e com muitos elementos repetidos.

Sendo assim utilizaremos cinco tamanhos de vetores diferentes e quatro distribuições de elementos, totalizando uma quantia de vinte cenários diferentes para ser analisado ao menos cinco vezes em cada um dos oito algoritmos aqui avaliados.

Em cada cenário analisado serão registrados dados sobre o número de atribuições e de comparações feitas no vetor, que em seguida serão exibidos nos resultados.

5 RESULTADOS

Aqui estão os resultados dos testes realizados nos cenários descritos anteriormente.

5.1 BUBBLE SORT

5.1.1 Comparações

	Aleatorio	Quase Ordenado	Inv. Ordenado	Repetidos
10^2	9306	693	9603	7722
10^3	991008	7992	993006	932067
10^4	99140085	89991	99950004	97450254
10^5	9906200937	999990	9999800001	9944000559
10^6	NULL	NULL	NULL	NULL

5.1.2 Atribuições

	Aleatorio	Quase Ordenado	Inv. Ordenado	Repetidos
10^2	8043	423	10371	7068
10^3	762264	4224	1451094	749040
10^4	74312238	42030	149484591	75605889
10^5	7512681453	419583	14994853377	7473100884
10^6	NULL	NULL	NULL	NULL

5.2 BUBBLE SORT COM SENTINELA

5.2.1 Comparações

	Aleatorio	Quase Ordenado	Inv. Ordenado	Repetidos
10^2	1900	1908	2059	1788
10^3	184629	254349	205795	184017
10^4	18453753	11542759	20361953	18413768
10^5	1842168146	3067032680	2041626972	1843218397
10^6	NULL	NULL	NULL	NULL

5.2.2 Atribuições

	Aleatorio	Quase Ordenado	Inv. Ordenado	Repetidos
10^2	4420	395	5503	4030
10^3	449876	4191	610524	446453
10^4	44947304	42026	61017085	45054740
10^5	4510622030	419210	6124189875	4510572345
10^6	NULL	NULL	NULL	NULL

5.3 BUBBLE SORT COCKTAIL SHAKER

5.3.1 Comparações

	Aleatorio	Quase Ordenado	Inv. Ordenado	Repetidos
10^2	5940	594	7623	5940
10^3	476523	6993	980019	498501
10^4	51444855	89991	99720027	50404959
10^5	5028849711	899991	9997600023	5007749922
10^6	NULL	NULL	NULL	NULL

5.3.2 Atribuições

	Aleatorio	Quase Ordenado	Inv. Ordenado	Repetidos
10^2	7512	435	11163	7008
10^3	720564	4050	1449165	728427
10^4	76059789	42339	149484621	74803839
10^5	7523061132	418974	14994848034	7492000080
10^6	NULL	NULL	NULL	NULL

5.4 INSERCTION SORT

5.4.1 Comparações

	Aleatorio	Quase Ordenado	Inv. Ordenado	Repetidos
10^2	2524	135	3783	2325
10^3	250319	1400	483088	248614
10^4	24925870	14013	49828319	24947032
10^5	2496725225	139740	4998482442	2505577578
10^6	NULL	1402844	NULL	NULL

5.4.2 Atribuições

	Aleatorio	Quase Ordenado	Inv. Ordenado	Repetidos
10^2	2722	333	3981	2523
10^3	252317	3398	485086	250612
10^4	24945868	34011	49848317	24967030
10^5	2496925223	339738	4998282444	2505777576
10^6	NULL	3402842	NULL	NULL

5.5 SELECTION SORT

5.5.1 Comparações

	Aleatorio	Quase Ordenado	Inv. Ordenado	Repetidos
10^2	5049	5049	5049	5049
10^3	500499	500499	500499	500499
10^4	50004999	50004999	50004999	50004999
10^5	5000049999	5000049999	5000049999	5000049999
10^6	NULL	NULL	NULL	NULL

5.5.2 Atribuições

	Aleatorio	Quase Ordenado	Inv. Ordenado	Repetidos
10^2	120	162	99	99
10^3	1029	1623	999	999
10^4	10026	16068	9999	9999
10^5	100035	161430	99999	99999
10^6	NULL	NULL	NULL	NULL

5.6 MERGE SORT

5.6.1 Comparações

	Aleatorio	Quase Ordenado	Inv. Ordenado	Repetidos
10^2	536	404	509	530
10^3	8603	5637	7062	8562
10^4	119562	73892	87970	119582
10^5	1526319	901733	1045371	1525984
10^6	18560858	10638710	12086103	18561303

5.6.2 Atribuições

	Aleatorio	Quase Ordenado	Inv. Ordenado	Repetidos
10^2	1344	1344	1344	1344
10^3	19952	19952	19952	19952
10^4	267232	267232	267232	267232
10^5	3337856	3337856	3337856	3337856
10^6	39902848	39902848	39902848	39902848

5.7 HEAP SORT

5.7.1 Comparações

	Aleatorio	Quase Ordenado	Inv. Ordenado	Repetidos
10^2	1344	1344	1344	1344
10^3	19952	19952	19952	19952
10^4	267232	267232	267232	267232
10^5	3337856	3337856	3337856	3337856
10^6	39902848	39902848	39902848	39902848

5.7.2 Atribuições

	Aleatorio	Quase Ordenado	Inv. Ordenado	Repetidos
10^2	1767	1884	1662	1668
10^3	27213	29025	25146	27153
10^4	372204	393885	349341	372357
10^5	4724376	4947645	4489104	4725738
10^6	57149343	59328996	54917466	57142287

5.8 QUICK SORT

5.8.1 Comparações

	Aleatorio	Quase Ordenado	Inv. Ordenado	Repetidos
10^2	579	614	531	746
10^3	10657	9943	9979	12684
10^4	146034	146075	151098	170136
10^5	1959313	1945109	1925342	2090274
10^6	24141819	24552279	24502073	26236616

5.8.2 Atribuições

	Aleatorio	Quase Ordenado	Inv. Ordenado	Repetidos
10^2	828	678	891	1085
10^3	10816	6624	10136	13097
10^4	131680	67288	102770	153558
10^5	1538152	671870	1034338	1776925
10^6	17707539	6718560	10345663	20002381

6 CONCLUSÃO

Como pode-se verificar com os testes realizados, não existe um algoritmo perfeito para todos os possiveis cenários, até mesmo os mais sofisticados e complexos são ultrapassados pelos simples em alguns casos específicos.

Apesar deste resultado ambíguo, algumas ressalvas são importantes: Em um caso geral com vetores de tamanhos maiores 10^5 a eficácia do Merge Sort, Heap Sort e Quick Sort sobre os demais é gigantesca, deixando extremamente frisado a diferença de um Big-O de O(n^2) para o O($n\log(n)$) desses algoritmos.

Dissertando mais especificamente sobre os resultos obtidos por cada algoritmo, primeiramente, sobre os algoritmos intuitivos:

- Bubble sort e suas otimizações, bubble com sentinela e cocktail sort, todos funcionam de maneira semelhante, apesar de os dois ultimos trabalharem mais eficientemente, o resultado final não varia muito. Funciona muito bem para vetores quase ordenados, e não muito bem nos demais, possui um declinio muito grande na eficiencia com o aumento do vetor.
- O inserction sort em um caso geral possui em desempenho semelhante ao bubble, porém no caso de vetor quase ordenado possui o melhor desempenho entre todos aqui avaliados.
- Selection sort possui a mesma ordem que os demais intuitivos, porém possui o pior desempenho entre eles, para todos os casos. Seu unico ponto positivo é o baixo número de atribuições.

Por fim, os algoritmos mais sofisticados e seus pontos positivos e negativos:

- Merge sort, o primeiro dos complexos aqui avaliados, possui O($n\log(n)$) sendo muito mais eficiente que os anteriores e possuindo um desempenho constante para todos os casos, se destacando um pouco mais no caso quase ordenado. Seu principal ponto negativo, apesar de não ser exibido pelos testes, é seu gasto de memória ser O(n), contra O(1) dos demais algoritmos.
- Heap sort, organiza o vetor no mesmo Big O que o merge, porém sem o ponto negativo de gasto extra de espaço. Algoritmo muito eficiente, tendo apenas um grande defeito, que é ser contante no tempo independente do tipo de vetor, o gasto é o mesmo.
- E por ultimo, quick sort, assim como os demais aqui avaliados, possui um desem-

pelho excelente em seu caso geral, não necessitando de espaço extra como o merge, e se beneficiando mais da situação do vetor que o heap. Possui um pouco de dificuldade em vetores com muitos elementos repetidos, porém seu defeito esta no pior caso, que apesar de raro, o torna um algoritmo com complexidade de tempo O(n^2).

REFERÊNCIAS

GEEKS FOR GEEKS. **Geeks for Geeks - a computer science portal for geeks**. Disponível em: <https://www.geeksforgeeks.org/bubble-sort/>. Acesso em: 19 set. 2018.

GEEKS FOR GEEKS. **Geeks for Geeks - a computer science portal for geeks**. Disponível em: <https://www.geeksforgeeks.org/insertion-sort/>. Acesso em: 19 set. 2018.