UNIVERSIDADE DE SÃO PAULO - ESCOLA DE ARTES, CIÊNCIA E HUMANIDADES
BACHARELADO DE SISTEMA DE SISTEMA DE INFORMAÇÃO

RAFAEL MACHADO DE MORAES

MÉDIA, MODA E MEDIANA

SALVADOR - BA
2021

I. Sumário

Sumário

1 Introdução.....3

2 Objetivo.....4

3 Resultados.....5

4 Conclusão.....5

1 Introdução

Algoritmos de seleção são de extrema importância hoje em dia, sendo usados em diversos campos do conhecimento, como por exemplo, em finanças, para escolher uma ação específica numa lista, até no dia a dia, para escolher um produto na lista de mercado digital ou na pesquisa do Google.

“Tempo é dinheiro”, esse velho ditado nunca se fez tão presente na nossa sociedade, com a sociedade informatizada em que vivemos com milhares de informações sendo compartilhadas a todo segundo. Por isso, a busca por algoritmos com tempos de execução menores é importante, possibilitando uma melhor eficiência nos processos que servem de base para as comunicações, transações econômicas e diversos outros processos.

Tendo isso em vista, o professor de Introdução a Análise de Algoritmos propôs que fizéssemos como Exercício de Programação(EP) a comparação entre dois algoritmos de seleção e comparasse os resultados, justificando o motivo da diferença de tempo entre um e o outro.

Para o primeiro algoritmo, foi utilizado o quicksort para ordenar a lista. Já no segundo algoritmo foi usado o algoritmo dado pelo professor. A linguagem utilizada foi Python 3.8 e executado no terminal do Linux.

O selection 1 usa um algoritmo de quicksort, um algoritmo que segue a ideia de dividir e conquistar. Ele funciona selecionando um pivô, dividindo a lista em duas partes, uma que vai ser menor que o pivô e a outra maior, a partir disso ele particiona separando os números maiores e os menores que o pivô. Recursivamente ele ordena essas duas partes, repetindo o processo até que a lista esteja organizada. Quando essa lista estiver ordenada, o programa retira o primeiro número da sequência, pois esse vai ser o i-ésimo menor número.

O selection 2 é um algoritmo que utiliza o pivô também, mas a principal diferença é como as listas divididas são utilizadas. As duas listas são divididas pela pivô e ordenada pela partição, uma parte menor que o pivô e a outra maior, se a posição desejada for maior que a posição do pivô ele roda o algoritmo novamente apenas com a lista maior que o pivô e a se for menor com a lista menor, isso continua até achar o número desejado. Devido a alta dependência do pivô, caso um pivô favorável seja achado o tempo de processamento pode ser reduzido, apresentando alta variância.

No selection 1 o tempo de processamento vai ser o mesmo do quicksort, pois devolver o menor número não impacta de maneira significativa no programa, pois o menor

sempre vai ser o primeiro da lista. O tempo médio vai ser Θ(n$log\ n$), no melhor caso ΩΘ(n $log\ n$), no pior caso O(n²)

No selection 2, o melhor caso vai ser quando o nosso pivô da partição for o valor na posição que a função está buscando, por isso o melhor tempo de execução vai ser Ω(n), pois esse é o tempo de execução da partição. Já no caso médio como a partição vai sendo dividida na metade e temos de usar apenas uma das metades, em cada iteração temos: n + ½ n + ¼ n + ⅛ n < 2 n logo o tempo médio vai ser de Θ(n).

2 Objetivo

Após executar os programas obtivemos os tempos de execução de cada algoritmo, com esses dados utilizei a ferramenta de gerar gráficos do google planilhas e obtive o seguinte resultado:

Amostragem*1000	Selection1 Tempo	Amostragem*1000	Selection2 Tempo
100	0,3534	100	0,04124355316
200	0,7784	200	0,01902008057
300	1,187555	300	0,1004328728
400	1,697079	400	0,127194643
500	2,25731	500	0,174705267
600	2,56023	600	0,1364138126
700	3,12401127	700	0,2509181499
800	3,5749428	800	0,3557026386
900	4,0803900	900	0,3462336063
1000	4,5316	1000	0,1801285744
2000	9,918529	2000	0,3963053226
3000	15,40417	3000	0,9432611465
4000	22,517194	4000	0,7444524765
5000	26,6901560	5000	1,313798428
6000	33,412000	6000	1,0023911
7000	38,0937	7000	1,477936029
8000	46,22029	8000	2,583782196
9000	52,625420	9000	2,504620314
10000	57,8486000	10000	1,316442966

11000	64,362453	11000	8,290535688
12000	70,105874	12000	6,361506939
13000	76,75535	13000	5,310325623
14000	85,679253	14000	3,510884523
15000	95,14778	15000	1,725472689
16000	97,82770	16000	2,999705553
17000	102,928465	17000	1,91756916
18000	112,823899	18000	3,63045001
19000	118,6905491	19000	6,899151325
20000	123,542314	20000	2,650235653

3 Resultados

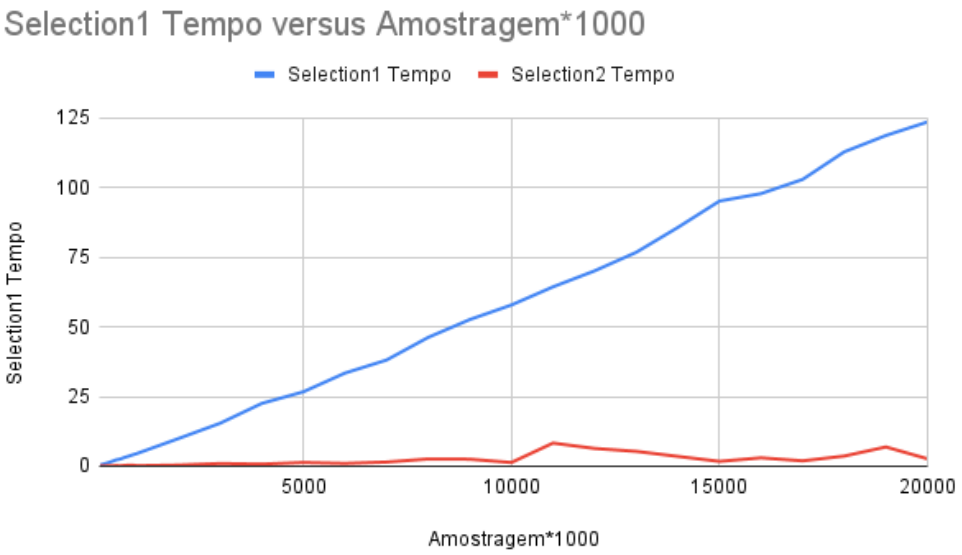

Imagem 1: Comparação do tempo de execução dos algoritmos

4 Conclusão

Com esse gráfico podemos perceber a clara diferença entre os tempos de execução dos algoritmos de seleção, com o Selection 2 performando melhor. A diferença de tempo como essa se dá devido a forma como eles organizam as listas que eles recebem, com o selection 1 ordenando a lista por completo e o selection 2 ordenando apenas o mínimo para retirar o número desejado.

O selection 1 usa um algoritmo de quicksort, um algoritmo que segue a ideia de dividir e conquistar. Ele funciona selecionando um pivô, dividindo a lista em duas partes, uma que vai ser menor que o pivô e a outra maior, a partir disso ele particiona separando os números maiores e os menores que o pivô. Recursivamente ele ordena essas duas partes, repetindo o processo até que a lista esteja organizada. Quando essa lista estiver ordenada, o programa retira o primeiro número da sequência, pois esse vai ser o i-ésimo menor número.

O selection 2 é um algoritmo que utiliza o pivô também, mas a principal diferença é como as listas divididas são utilizadas. As duas listas são divididas pela pivô e ordenada pela partição, uma parte menor que o pivô e a outra maior, se a posição desejada for maior que a posição do pivô ele roda o algoritmo novamente apenas com a lista maior que o pivô e a se for menor com a lista menor, isso continua até achar o número desejado. Isso apresenta uma clara vantagem em relação ao quicksort, que ordena toda a lista, esse algoritmo ordena apenas a parte que interessa para concluir a tarefa, tornando-o mais eficiente.

5 Referências

https://www.ime.usp.br/~pf/algoritmos/aulas/quick.html
https://docs.python.org/3/library/random.html
https://docs.python.org/3/library/time.html