Randomized Search

José Pedro Estima Santos nº 89129

***Resumo* – Ao longo deste relatório são apresentados os algoritmos desenvolvidos, randomized search e linear search, a análise formal de cada um dos algoritmos, bem como os testes computacionais realizados e as conclusões retiradas.**

***Abstract* – Throughout this report are presented the algortihms developed, randomized search and linear search, the formal analysis of each of the algorithms, as well as the computacional teste carried out and the conclusions drawn.**

I. INTRODUÇÃO

O objetivo do trabalho aqui apresentado é analisar o desempenho dos dois algoritmos de pesquisa implementados. Um dos algoritmos envolve pesquisa linear, enquanto que o outro envolve pesquisa aleatória, ou randomized search, ao longo deste relatório vão ser apresentados os algoritmos e uma análise formal sobre cada um, os eventuais testes computacionais realizados e as devidas conclusões retiradas durante o desenvolvimento do trabalho.

II. ALGORITMOS E ANÁLISE FORMAL

Começando no algoritmo de pesquisa linear que pode ser visto na seguinte figura:

```
def linear_seach(array,n):
    global opCount
    for i in range(0,len(array)-1):
        opCount+=1
        if array[i] == n:
            return True;
    return False;
```

Figura 1: algoritmo de pesquisa linear.

O funcionamento deste algoritmo é relativamente simples, ele pesquisa de forma linear, um a um, os elementos que existem no array até encontrar o elemento cujo valor é n. O número de operações representa o número de iterações que o algoritmo faz sobre o array até encontrar o valor pretendido.

Relativamente à sua análise formal, o pior caso é quando o elemento que se procura é o último elemento do array, sendo que nesta situação o algoritmo terá de percorrer todos os seus elementos até encontrar o elemento pretendido. Esta é uma situação que pode ocorrer se o conjunto de números que estiver a ser analisado não contém nenhum elemento repetido. Tal como o seu nome nome indica, este algoritmo cresce de forma linear sendo que a sua complexidade, O(n) equivale ao pior caso.

Observemos agora o algoritmo de pesquisa aleatória que pode ser visto na seguinte figura:

```
def randomized_search(array,n):
    global opCount1
    l = [];
    for i in range(0,len(array)-1):
        test = random.randint(0,len(array)-1)
        while test in l:
            test = random.randint(0,len(array)-1)
        l.append(test)
        opCount1+=1
        if(array[test] == n):
            return True;
    return False;
```

Figura 2: algoritmo de pesquisa aleatória.

O funcionamente deste algoritmo também é relativamente simples, é usado uma lista auxiliar para guardar os valores dos índice que já foram consultados, garantindo assim que o mesmo índice não é consultado mais do que uma vez. O modo de operação é semelhante ao algoritmo anterior, na medida em que se for encontrado o valor que está a ser procurado a função termina e o número de operações consiste na quantidade de índices que foram consultados. Ao contrário do algoritmo anterior a pesquisa não é feita linearmente, são obtidos valores de índicies de forma aleatória e é consultada a posição do índice obtido.

Relativamente à sua análise formal, o pior caso também é aquele onde, num conjunto de números não repetidos, o valor que se procura é exatamente o último valor a ser pesquisado pelo algoritmo. Apesar da sua pesquisa não ser feita de forma linear, o algoritmo cresce de forma linear e portanto a sua complexidade é de O(n).

III. TESTES COMPUTACIONAIS

Para os testes computacionais foram usados três ficheiros, um contém mil valores aleatórios no intervalo [0,100], outro contém cem valores ordenados no intervalo [0,100] e por fim um ficheiro que contém valores não repetidos, não ordenados de [0,100].

Para todos os testes realizados foram feitas as pesquisas pelo elemento 100 vezes e obtida uma média das operações efetuadas.

Procurando pelo número 0, obtemos os seguintes outputs.

```
[Running] python "/home/kanto/Documents/AA/AA_2/RandomizedSearch.py"
Executed 100 times
--------------------------------------------------------------------
        |       |       | Repeated values - 1000 elements
--------------------------------------------------------------------
Average of Linear Search 221 operations!
Average of Random Search 98 operations!
--------------------------------------------------------------------
        |       |       | Non-repeated sorted values - 100 elements
--------------------------------------------------------------------
Average of Linear Search 1 operations!
Average of Random Search 53 operations!
--------------------------------------------------------------------
        |       |       | Non repeated non sorted values - 100 elements
--------------------------------------------------------------------
Average of Linear Search 93 operations!
Average of Random Search 48 operations!
--------------------------------------------------------------------

[Done] exited with code=0 in 0.078 seconds
```

Figura 3 : Outputs para pesquisa pelo número 0.

Como podemos observar neste caso, para valores repetidos, em média, a pesquisa aleatória encontra muito mais rapidamente o valor pretendido, para valores não repetidos e ordenados tal não acontece. Contudo, neste caso, para valores não repetidos não ordenados também é possível ver que a pesquisa aleatória é melhor que a tradicional pesquisa linear.

Procurando agora pelo valor 50, obtemos os seguintes outputs.

```
[Running] python "/home/kanto/Documents/AA/AA_2/RandomizedSearch.py"
Executed 100 times
--------------------------------------------------------------------
        |       |       | Repeated values - 1000 elements
--------------------------------------------------------------------
Average of Linear Search 32 operations!
Average of Random Search 70 operations!
--------------------------------------------------------------------
        |       |       | Non-repeated sorted values - 100 elements
--------------------------------------------------------------------
Average of Linear Search 51 operations!
Average of Random Search 49 operations!
--------------------------------------------------------------------
        |       |       | Non repeated non sorted values - 100 elements
--------------------------------------------------------------------
Average of Linear Search 36 operations!
Average of Random Search 48 operations!
--------------------------------------------------------------------

[Done] exited with code=0 in 0.097 seconds
```

Figura 4 : Outputs para pesquisa pelo número 50.

Como podemos observar, neste caso, em média, os valores obtidos com a pesquisa linear são melhores que aqueles obtidos com a pesquisa aleatória, exceto nos valores ordenados não repetidos onde a quantidade de operações é um valor relativamente próximo um do outro.

Procurando agora pelo número 100, obtemos os seguintes outputs.

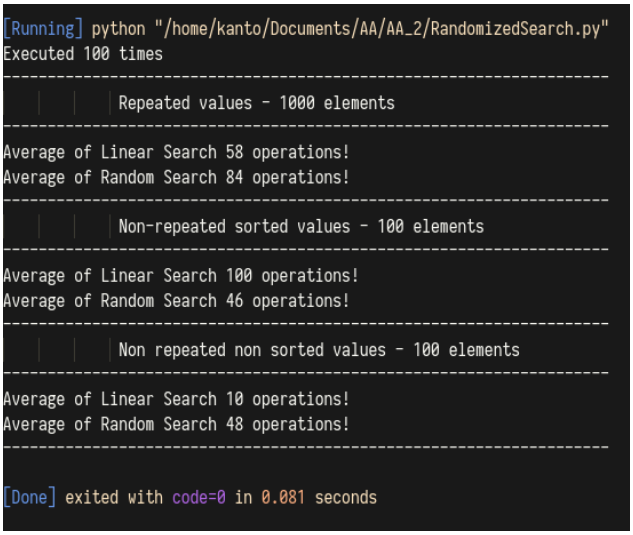

Figura 5 : Outputs para pesquisa pelo número 50.

Neste caso podemos observar, em média, que tanto para valores repetidos, como para valores não repetidos e não ordenados a pesquisa linear encontrou os resultados mais rapidamente, a pesquisa aleatória apenas foi melhor no caso dos elementos não repetidos e ordenados.

Procurando agora por um número aleatório, 77, obtemos os seguintes resultados.

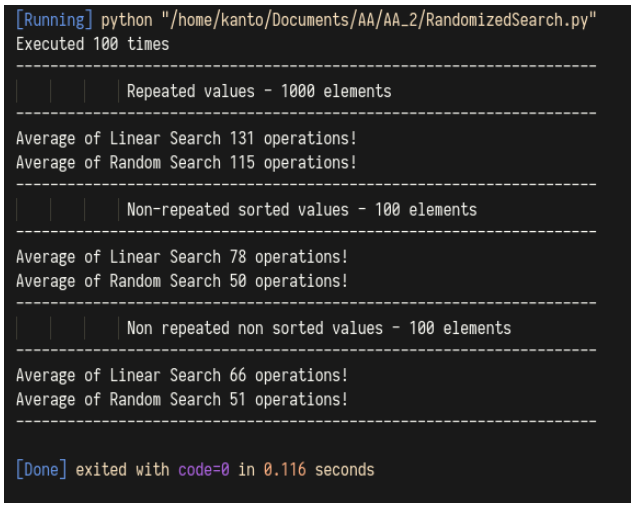

Figura 6 : Outputs de pesquisa para o elemento 77.

Neste caso, em média, a pesquisa aleatória foi sempre superior à pesquisa linear, tanto nos valores repetidos, como nos valores não repetidos ordenados e não ordenados.

IV. CONCLUSÃO

Cada caso é um caso, por vezes podemos obter melhores resultados usando a pesquisa linear, outras vezes usando a pesquisa aleatória, tudo depende dos elementos que queremos pesquisar e da forma como esses elementos estão representados. Nem sempre a pesquisa aleatória é melhor que a linear e vice-versa, ambas se aplicam em diferentes situações, a pesquisa aleatória é melhor se o conjunto de dados tiver muitos elementos repetidos, e para elementos ordenados que se encontrem para cima do meio do espaço de pesquisa isso também se verifica, para elementos repetidos não ordenados, tudo depende da posição no array onde eles se encontram.

A pesquisa linear apenas é melhor nas situações em que a probabilidade da pesquisa aleatória atingir aquele elemento, em média, num número menor ou igual de operações que a pesquisa linear for muito baixa.

Em suma, a pesquisa aleatória pode ser mais rápido do que a pesquisa linear, em média, se o espaço de pesquisa (array) conter múltiplas ocorrências do elemento pelo qual se procura [1] (página 327).

V. REFERÊNCIAS

[1] - Vrajitoru D., Knight W. (2014) Algorithms and Probabilities. In: Practical Analysis of Algorithms. Undergraduate Topics in Computer Science. Springer, Cham.