# Trabalho Computacional
# Algoritmos e Estruturas de Dados II

Davi Juliano Ferreira Alves
Instituto de Ciência e Tecnologia
Bacharelado em Ciência e Tecnologia
Universidade Federal de São Paulo
Av. Cesare Monsueto Giulio Lattes, 1201, 12247-014
Email: davi.juliano@unifesp.br

***Abstract*—A pandemia de COVID-19 foi um evento histórico em 2020. Devido à esse grande evento, foi-se feito uma análise exploratória nos dados dos respiradores enviados para toda a população. A análise consistiu em vislumbrar quais foram os maiores distribuidores dentro de um certo centro de distribuição, quais foram os destinos mais receptíveis com o material, o valor de cada material e sua relação direta com a economia, e também a quantidade de respiradores doados sobre uma determinada situação.**

## I. Introdução

A pandemia de COVID-19 foi uma realidade histórica em 2020, devido à sua grande contaminação. De acordo com a descrição do [1], a transmissão acontece de uma pessoa doente para outra ou por contato próximo por meio de:

- Toque do aperto de mão contaminadas;
- Gotículas de saliva;
- Espirro;
- Tosse;
- Catarro;
- Objetos ou superfícies contaminadas, como celulares, mesas, talheres, maçanetas, brinquedos, teclados de computador etc.

Ou seja, a transmissão se dá de maneira respiratória e de contato direto com líquidos humanos e, portanto, de maneira facilitada para uma alta transmissão. De acordo com o [1] novamente, os sintomas da COVID-19 podem variar de um resfriado, a uma Síndrome Gripal-SG (presença de um quadro respiratório agudo, caracterizado por, pelo menos dois dos seguintes sintomas: sensação febril ou febre associada a dor de garganta, dor de cabeça, tosse, coriza) até uma pneumonia severa. Portanto, os sintomas mais comuns são:

- Tosse
- Febre
- Coriza
- Dor de garganta
- Dificuldade para respirar
- Perda de olfato (anosmia)
- Alteração do paladar (ageusia)
- Distúrbios gastrintestinais (náuseas/vômitos/diarreia)
- Cansaço (astenia)
- Diminuição do apetite (hiporexia)
- Dispnéia ( falta de ar)

Como um dos sintomas é dado pela dificuldade de respirar quando a doença se apresenta de maneira mais aguda, um dos métodos de se ajudar pessoas infectadas que possuem esse quadro de dificuldade respiratória é prover respiradores, como o da imagem abaixo:

Fig. 1. Exemplo de respirador (Imagem de [2])

Nesse trabalho, iremos fazer uma análise exploratória em cima do dado que está no url a seguir (Além do arquivo .csv, há também o Google Colab (Arquivo .ipynb) para que os arquivos possam ser testados devidamente):

- https://drive.google.com/drive/folders/1nfp-CS4R35FYs4vtSw6YkySo1-ZEMFfz?usp=sharing

Além do link acima, pode-se adquirir o arquivo utilizado por meio do link dado pelo próprio banco de dados:

- http://sage.saude.gov.br/dados/repositorio/distribuicao_respiradores.csv

## II. Definição dos Algoritmos Utilizados

### A. Algoritmo do Selection Sort

De acordo com as definições de classe, temos que:

**Definição 1.** *Selection Sort é um algoritmo de ordenação baseado em selecionar o menor item do vetor, trocá-lo com o item da primeira posição do vetor e repitir essas duas*

*operações com os $n-1$ itens restantes, depois com os $n-2$ itens, até que reste apenas um elemento.*

O algoritmo utilizado é dado pelo [3] no seguinte código do Python:

```
#Codigo do Prof. Fabio Kon (prof do IME/USP) no Algoritmo de Selecao

def selecao(lista):
  fim=len(lista)
  for i in range(fim-1):
    posicao_minimo = i
    for j in range(i+1, fim):
      if lista[j] < lista[posicao_minimo]:
        posicao_minimo = j

      lista[i], lista[posicao_minimo] = lista[posicao_minimo], lista[i]
```

### B. Algoritmo do Quick Sort

De acordo com as definições de classe, temos que:

**Definição 2.** *Quick Sort é um algoritmo de dividir para conquistar utilizando os passos a seguir:*

- *Escolha arbitrariamente um pivô x.*
- *Percorra o vetor a partir da esquerda até que A[i] $\geq$ x.*
- *Percorra o vetor a partir da direita até que A[j] $\leq$ x.*
- *Troque A[i] com A[j].*
- *Continue este processo até os apontadores i e j se cruzarem.*
- *Ao final, o vetor A[Esq..Dir] está particionado de tal forma que:*
- *Os itens em A[Esq], A[Esq + 1], . . . , A[j] são menores ou iguais a x.*
- *Os itens em A[i], A[i + 1], . . . , A[Dir] são maiores ou iguais a x.*

O algoritmo utilizado é dado pelo [4] no seguinte código do Python:

```
#Funcao do Prof. Hallison Paz (Infnet) com ordenacao por QuickSort

def quicksort(lista, inicio=0, fim=None):
    if fim is None:
        fim = (len(lista)-1)
    if inicio < fim:
        p = partition(lista, inicio, fim)
        # recursivamente na sublista a esquerda (menores)
        quicksort(lista, inicio, p-1)
        # recursivamente na sublista a direita (maiores)
        quicksort(lista, p+1, fim)

def partition(lista, inicio, fim):
    pivot = lista[fim]
    i = inicio
    for j in range(inicio, fim):
        # j sempre avanca, pois representa o elementa em analise
        # e delimita os elementos maiores que o pivo
        if lista[j] <= pivot:
            lista[j], lista[i] = lista[i], lista[j]
            # incrementa-se o limite dos elementos menores que o pivo
            i = i + 1
    lista[i], lista[fim] = lista[fim], lista[i]
    return i
```

### C. Algoritmo do Counting Sort

De acordo com as definições de classe, temos que:

**Definição 3.** *O Counting Sort é um algoritmo que pressupõe que cada um dos n elementos de entrada é um inteiro no intervalo de 1 a k, pois ele atua com um vetor temporário que será o vetor auxiliar que contará qual valor há dentro, para, posteriormente, retornar ao vetor original com a contagem feita e ordenado.*

O algoritmo utilizado é dado pelo [5] no seguinte código do Python (O código está em C no vídeo anexo nas referências, entretanto, o autor do trabalho tomou a liberdade de transformá-lo em Python, mantendo a referência à lógica do citado Prof. André Backes):

```
#Funcao do Prof. Andre Backes (UFU) com ordenacao por CountingSort
#Adaptada para o Python pelo autor do trabalho

def countingsort(lista):
  fim=len(lista)
  baldes=[0 for i in range(302)]

  for j in range(fim):
    baldes[lista[j]] += 1

  i=0
  for j in range(302):
    k=baldes[j]
    while k>0:
      lista[i]=j
      i+=1
      k-=1
```

### D. Código de Leitura dos Dados

Iremos extrair apenas 5 colunas dentre as 10 colunas dadas no arquivo csv, sendo essas colunas dadas por ”FORNECEDORES”, ”DESTINO”, ”QUANTIDADE”, ”VALOR” e ”DATA”. O código é dado por:

```
#Abertura do arquivo e a obtencao dos valores que serao arquivados em cada lista

fornecedor=[]
destinos=[]
valor_respiradores=[]
data_entrega=[]
qnt_respiradores=[]

with open('distribuicao_respiradores_new.csv', newline='') as dist:
  leitor = csv.DictReader(dist)
  for key in leitor:
    for i in range(len(key)):
      fornecedor.append(key['FORNECEDOR'])
      qnt_respiradores.append(int(key['QUANTIDADE'])
    )
      destinos.append(key['DESTINO'])
```

```
        valor_respiradores.append(float(key['VALOR']))
        data_entrega.append(key['DATA'])
```

## III. Resultados

Agora analisaremos os resultados dos métodos evidenciados acima dentro das listas também evidenciadas e definidas. Teremos perguntar para responder dentro dos dados na análise exploratória:

1) Qual a média dos valores dos respiradores?
2) Quais foram as maiores quantidades de respiradores distribuídos dentro de uma determinada encomenda? Plote o gráfico das quantidades
3) Quais foram os destinos mais escolhidos em encomendas? Mostre por meio de um gráfico.
4) Quais foram os fornecedores que realizaram mais encomendas? Mostre por meio de um gráfico.
5) Quais foram os pontos atenuantes ou anômalos na análise?

### A. *Qual a média dos valores dos respiradores?*

A média dos valores será dada pelo somatório do valor dos respiradores dividido pela quantidade de respiradores, portanto, temos o seguinte código:

```
def media(lst, lista):
    return sum(lst) / sum(lista)

#Utilizando a funcao media, temos

valor_medio = media(valor_respiradores,
    qnt_respiradores)
print(f'O valor medio e {valor_medio}')
```

E o resultado é dado por "O valor medio é 52431.45742149624", ou seja, R$ 52431,45.

### B. *Quais foram as maiores quantidades de respiradores distribuídos dentro de uma determinada encomenda? Plote o gráfico das quantidades*

Para a quantidade de respiradores, utilizaremos os 3 algoritmos de ordenação, o Counting, o Selection e o Quick Sort:

```
quantidade = qnt_respiradores

ini=time.time()
countingsort(quantidade)
fim=time.time()

print(quantidade)
print(len(quantidade))
print(f'Tempo de execucao: {fim-ini} segundos')

quantidade = qnt_respiradores

ini=time.time()
selecao(quantidade)
fim=time.time()

print(quantidade)
print(len(quantidade))
print(f'Tempo de execucao: {fim-ini} segundos')

quantidade = qnt_respiradores

#Para que o quicksort rodasse com a quantidade de
    valores dentro do colab, utilizamos essa funcao
    para aumentar a quantidade de recursoes
    permitidas
sys.setrecursionlimit(max(sys.getrecursionlimit(),
    len(quantidade)+1000))

ini=time.time()
quicksort(quantidade)
fim=time.time()

print(quantidade)
print(f'Tempo de execucao: {fim-ini} segundos')
```

Temos os resultados do Counting Sort dados em um tempo de execução de 0.0072 segundos, com uma quantidade de 18973 valores.

Para os resultados do Selection Sort, temos um tempo de execução de 29.7948 segundos, com a mesma 18973 quantidade de valores.

Para os resultados do Quick Sort, temos um tempo de execução de 4.1201 segundos, com a mesma quantidade de 18973 valores na lista.

O gráfico plotado pela quantidade acumulada dos valores de cada encomenda é dado pelo seguinte comando:

```
dict_quantidade = dict(Counter(quantidade))
lista=[]
lista_valores=[]
lista=list(dict_quantidade.items())
for i in range(len(lista)):
  tupla = lista[i]
  lista_valores.append(tupla[1])

print(lista_valores)

ini=time.time()
quicksort(lista_valores)
fim=time.time()

print(lista_valores)
print(len(lista_valores))
print(f'Tempo de execucao: {fim-ini} segundos')

tuplas = sorted(dict_quantidade.items(), key=
    itemgetter(1), reverse=False)

lista_chaves=[]

for i in range(len(tuplas)):
  tupla = tuplas[i]
  lista_chaves.append(tupla[0])

fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
plt.xticks(rotation=90)
ax.bar(lista_chaves, lista_valores)
plt.savefig('quantidade.png', bbox_inches = "tight")
plt.show()
```

### C. *Quais foram os destinos mais escolhidos em encomendas? Mostre por meio de um gráfico.*

Utilizaremos o mesmo pensamento do plot do gráfico das quantidades, ou seja, teremos:

```
dict_destinos = dict(Counter(destinos))
lista=[]
lista_valores=[]
lista=list(dict_destinos.items())
for i in range(len(lista)):
  tupla = lista[i]
  lista_valores.append(tupla[1])

print(lista_valores)

ini=time.time()
countingsort(lista_valores)
fim=time.time()

print(lista_valores)
print(len(lista_valores))
print(f'Tempo de execucao: {fim-ini} segundos')


#Como eu nao sabia ordenar as keys, utilizei o
    sorted simplesmente para plotar os graficos
tuplas = sorted(dict_destinos.items(), key=
    itemgetter(1), reverse=False)

lista_chaves=[]

for i in range(len(tuplas)):
  tupla = tuplas[i]
  lista_chaves.append(tupla[0])

fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
plt.xticks(rotation=90)
ax.bar(lista_chaves, lista_valores)
plt.savefig('destinos.png', bbox_inches = "tight")
plt.show()
```

O que gerou um gráfico dado na seção de gráficos.

*D. Quais foram os fornecedores que realizaram mais encomendas? Mostre por meio de um gráfico.*

Utilizaremos o mesmo pensamento do plot do gráfico anterior, entretanto ordenaremos utilizando o Selection Sort, ou seja, teremos:

```
dict_fornecedor = dict(Counter(fornecedor))
lista=[]
lista_valores=[]
lista=list(dict_fornecedor.items())
for i in range(len(lista)):
  tupla = tuplas[i]
  lista_valores.append(tupla[1])

print(lista_valores)

ini=time.time()
selecao(lista_valores)
fim=time.time()

print(lista_valores)
print(len(lista_valores))
print(f'Tempo de execucao: {fim-ini} segundos')

#Como eu nao sabia ordenar as keys, utilizei o
    sorted simplesmente para plotar os graficos

tuplas = sorted(dict_fornecedor.items(), key=
    itemgetter(1), reverse=False)

lista_chaves=[]

for i in range(len(tuplas)):
  tupla = tuplas[i]
  lista_chaves.append(tupla[0])

fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
plt.xticks(rotation=90)
ax.bar(lista_chaves, lista_valores)
plt.savefig('fornecedor.png', bbox_inches = "tight")
plt.show()
```

O que gerou um gráfico que está dado na seção de gráficos gerados.

*E. Quais foram os pontos atenuantes ou anômalos na análise?*

Existiram diversos pontos atenuantes diante da análise dos dados, sendo uma delas a distinção entre as strings na análise do fornecedor, e, portanto, a distinção entre LEISTUNG e Leistung, o que gera uma certa dificuldade para a análise dos gráficos e dos dados, visto que são uma mesma empresa escrita de maneira distinta.

Outro ponto distoante é a aparição unicamente do Líbano como um dos destinos sendo que todos os outros destinos eram estados brasileiros.

## IV. Gráficos Plotados

### A. Gráfico das Quantidades

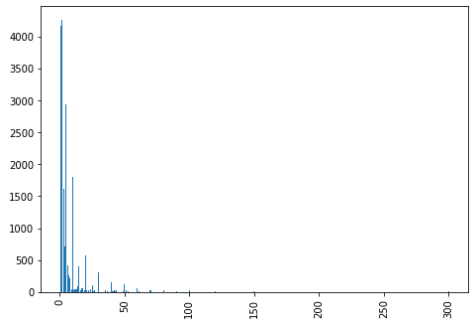


Fig. 2. Gráfico das quantidades acumuladas (Fonte: Autor)

### B. Gráfico dos Destinos

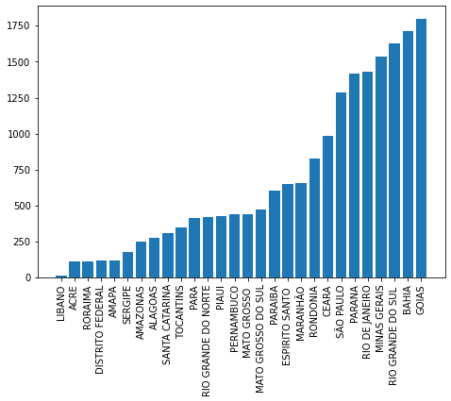


Fig. 3. Gráfico dos destinos das encomendas (Fonte: Autor)

### C. Gráfico dos Fornecedores

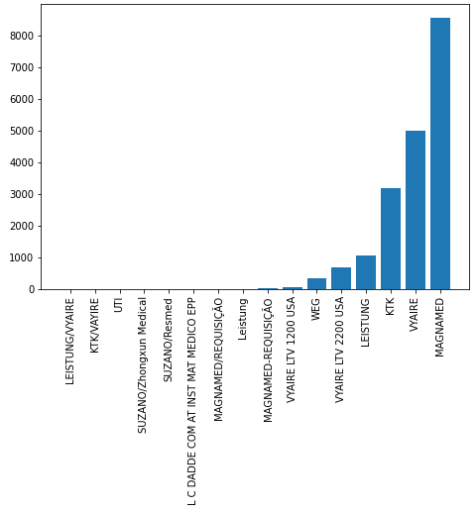


Fig. 4. Gráfico das empresas fornecedoras (Fonte: Autor)

## V. Resultados

### A. Quantidades

Como foi utilizado a função counter e não soube ordenar em cima das keys dentro do dicionário counter, foi-se utilizado um sorted apenas para o plot do gráfico, entretanto, os valores foram ordenados pelo Quick Sort, com um tempo de execução de 0.0001206 segundos, com 47 iterações.

### B. Destinos

Ou seja, o maior destino dos respiradores foi em Goiás (com 1799 encomendas), enquanto o menor destino dos respiradores foi o Líbano (com 11 encomendas). O Counting Sort executou em um total de 0.0002479 segundos, com 28 valores.

### C. Fornecedores

A empresa que mais forneceu respiradores foi a Magnemed (com 8563 encomendas) e a que menos forneceu foi a Leistung (com 11 encomendas), entretanto, a string Leistung que possui 11 encomendas está minúscula, ou seja, ela faz parte da quarta empresa no gráfico. Ou seja, de acordo com o gráfico e com a análise do dicionário do counter, temos que a L C Dadde com at Inst Mat Medico EPP (com 11 encomendas) é a que menos forneceu respiradores, de acordo com o gráfico. O tempo de execução do Selection foi de 5.745887756347656e-05 segundos, para 16 valores.

## VI. Conclusion

A conclusão que foi obtida é sobre a enorme importância dos respiradores diante de uma distribuição emergencial de um aparelho que pode salvar uma vida, se tratando de um cenário pandêmico da COVID-19. Para mais informações sobre o código, ele pode ser acessado diante do link a seguir: https://colab.research.google.com/drive/18NPwR9CHKth4beXgUo8iVQkEwicFCWTj?authuser=1#scrollTo=Zrpx0QLx94xj. Caso haja algum problema com o link disponibilizado, contatar o autor do trabalho.

## References


[1] Ministerio da Saude do Brasil. O que é covid-19. Disponível em: https://coronavirus.saude.gov.br/sobre-a-doenca. Accessed: 2020-12-16.

[2] Frances Jones. Governos e hospitais correm contra o tempo em busca de respiradores. Disponível em: https://www.uol.com.br/vivabem/noticias/redacao/2020/04/06/governos-e-hospitais-correm-contra-o-tempo-em-busca-de-respiradores.htm. Accessed: 2020-12-16.

[3] Fabio Kon. 38 - algoritmos de ordenação - seleção direta. Disponível em: https://www.youtube.com/watch?v=JggjFK5PkMs&ab_channel=CCSLdoIME%2FUSP. Accessed: 2020-12-16.

[4] Halisson Oliveira da Paz. Quicksort — algoritmos #8. Disponível em: https://www.youtube.com/watch?v=wx5juM9bbFo&feature=youtu.be. Accessed: 2020-12-16.

[5] André Backes. [ed] aula 123 - ordenação: Countingsort. Disponível em: https://www.youtube.com/watch?v=En8daEdcpJU&ab_channel=LinguagemCPrograma%C3%A7%C3%A3oDescomplicada. Accessed: 2020-12-16.