MS515 - Métodos Probabilísticos em Pesquisa Operacional

Prof.Lúcio Tunes Santos

Projeto - Programação Dinâmica

Bryan Prado - 195171
Fábio Paiva - 215570
Flávia Tartare - 108175
Renata Watanabe - 186543

Instituto de Matemática, Estatística e Computação Científica (IMECC)
Universidade Estadual de Campinas (UNICAMP)
Brasil, Novembro 2021

Conteúdo

1 Introdução:

Neste projeto, nosso objetivo era implementar e avaliar o funcionamento e eficiência de um algoritmo de programação dinâmica para um problema onde, dada uma sequência de n números $\in \mathbb{R}$, $(a_1, a_2, a_3..., a_n)$, devemos encontrar uma subsequência adjacente $(a_k, a_{k+1}, a_{k+2}, a_{k+3}..., a_m)$, $k \leq m$, que possui soma máxima.

Como um "extra", analisamos também o problema para se encontrar um caminho máximo dentro de uma matriz m X n a partir de sua primeira linha, retornando o valor da soma total e o percurso ótimo.

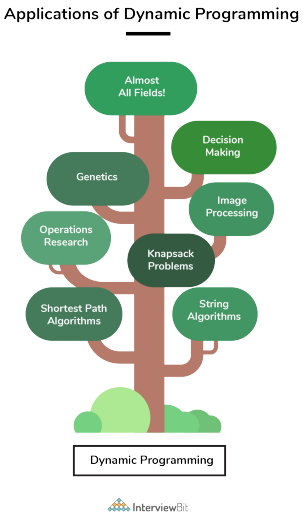
2 Programação Dinâmica: História e Ideias

Sendo um método desenvolvido na década de 1950 pelo matemático americano Richard Bellman, a programação dinâmica sofreu um grande processo de mudanças, aprimoramentos e abertura para novas aplicações durante o passar dos anos. Tendo sua essência no "Princípio de Otimalidade de Bellman", a programação dinâmica, também conhecida como otimização recursiva, ao contrário de outros ramos da programação matemática, não possui um único algoritmo que resolve todos os problemas possíveis, pelo contrário, é necessário uma análise e modelagem correta do problema proposto a se resolver, porém, sempre se baseando no "Princípio de Bellman".

Richard Bellman[1]

De maneira geral, o que a programação dinâmica faz é "quebrar"um problema baseado em possibilidades/decisões em problemas menores que geram um resultado para o problema original.

Desenvolvida em um momento histórico em que programação tinha como sinônimo "Método Tabular", suas primeiras aplicações foram em problemas relacionados a organização e logística, como o planejamento de estoques e entre outros. Em seguida, dada a versatilidade do método, problemas como a sequência de Fibonacci, o algoritmo de Djikstra e problemas relacionados a obtenção de subsequências máximas ou mínimas, também foram solucionados através de programação dinâmica.



Algumas Aplicações de Programação Dinâmica[2]

[1]Imagem retirada de https://int8.io/bellman-equations-reinforcement-learning/
[2]Imagem retirada de https://www.interviewbit.com/courses/programming/topics/dynamic-programming/

3 O Problema da Subsequência Adjacente Máxima:

3.1 Concepção Geral da Resolução:

Para a resolução de tal problema proposto, a partir de um algoritmo de programação dinâmica, concebemos que, dada uma sequência S qualquer de números reais, temos uma subsequência $A_{(j,i)}$, que se inicia no elemento j e termina no elemento i, sendo assim, temos dois caminhos possíveis para lidar com o elemento $i+1$; denotemos como $B_{(j,i)}$ a soma dos elementos da subsequência $A_{(j,i)}$, se a soma de $B_{(j,i)}$ com o elemento $i+1$ resultar em um valor maior que o anterior, o elemento $i+1$ entra na subsequência $A_{(j,i)}$, caso contrário, $i+1$ não entra na sequência, assim, $B_{(j,i+1)} = max(B_{(j,i)} + i + 1, 0)$

Com isso, basta definirmos uma variável qualquer que guardará a subsequência com maior soma possível até dado ponto, e assim, percorrer a subsequência (uma única vez) comparando continuamente tal variável com uma nova possível subsequência, se a nova possibilidade possuir uma soma maior que a da variável de armazenamento, a mesma se torna a nova subsequência adjacente máxima.

Como é possível observar, quebramos o problema de analisar todas as subsequências possíveis dentro da sequência S em pequenos problemas onde progressivamente percorrendo a sequência S nos utilizamos dos resultados dos mesmos para encontrar a solução ótima do problema original.

3.2 Algoritmo em Python:

Para a implementação do algoritmo, tivemos como pilares principais um código que permitisse uma ampla gama de interações por parte do usuário, executasse o proposto através de programação dinâmica e de forma rápida.

A seguir, nosso código em Python desenvolvido para resolução do problema separado função a função:

```
import numpy as np
import pandas as pd
import time
```

Primeiro, temos as bibliotecas que utilizamos para o problema; como trabalhamos extensamente com uma lógica de vetores, a utilização do *numpy* se fez necessária, enquanto que, o *pandas* foi utilizado para ofertarmos a possibilidade do usuário dar como entrada para o programa um arquivo do tipo *CSV*, e *time* foi utilizado para conseguirmos registrar o tempo de execução do programa para um dado problema.

```
Control=int(input("Voce gostaria que uma sequencia fosse gerada
randomicamente ou prefere inserir uma manualmente? 1 para geracao
randomica e 2 para manual: "))
Sum,initial_value,final_value,vector,total_time=Initialization(Control)
print("\nPara a sequencia: ",vector)
print("\nTemos como resultado de soma maxima: ",Sum)
print("\nTal soma ocorre na subsequencia que se inicia no elemento numero
",initial_value+1,"e termina no elemento numero",final_value+1)
print("\nSubsequencia que maximiza a soma: ",vector[initial_value:
final_value+1])
if total_time>0:
    print("\nO processo todo levou um tempo total de:",
    "{:.5f}".format(total_time),"segundos")
else:
    print("\nO processo todo foi tao rapido que a precisao da maquina
    nao conseguiu registrar o tempo corretamente")
```

Em seguida, temos nossa "central de controle", trecho do código onde ocorre a interação inicial do usuário com o programa, e após a execução de todo processo para obtenção do resultado ótimo, ocorrem os prints para visualização da subsequência ótima, a soma da mesma e do tempo decorrido durante a execução do código.

```
def Initialization(Control):
    if Control==1:
       j=int(input("\nQual seria o tamanho da sequencia? "))
       r=int(input("\nOs valores da sequencia sao de quais tipo? 1 para
       int e 2 para float: "))
       if r==1:
```

```
            t=np.array([int(item) for item in input("\nQuais seriam os
            valores limites da sequencia? Inserir limite inferior e
            superior na ordem: ").split()])
            v=np.random.randint(t[0],t[1]+1,size=j)
        if r==2:
            t=np.array([float(item) for item in input("\nQuais seriam os
            valores limites da sequencia? Inserir limite inferior e
            superior na ordem: ").split()])
            v=np.random.uniform(t[0],t[1]+2.3e-308,size=j)
        return Max(v)
    if Control==2:
        x=int(input("\nA sequencia se encontra em um arquivo CSV? 1 para
        sim e 2 para nao: "))
        if x==2:
            r=int(input("\nOs valores da sequencia sao de quais tipo? 1
            para int e 2 para float: "))
            if r==1:
                v=np.array([int(item) for item in input("\nInsira a
                sequencia : ").split()])
            if r==2:
                v=np.array([float(item) for item in input("\nInsira a
                sequencia : ").split()])
            return Max(v)
        if x==1:
            b=input("\nInsira aqui o endereco de seu arquivo (sem o .csv)
            : ")
            v=pd.read_csv(b+".csv", sep=';', header=None)
            v=np.array(v.values).flatten()
            return Max(v)
```

A função *Initialization* é a responsável por iniciar os parâmetros da sequência que se busca analisar, além disso, é onde ocorre o processo para escolha de qual meio de "entrada"será utilizado pelo programa, no caso, são três possibilidades; uma geração aleatória de sequência com n termos com limite de valores definidos pelo usuário, inserção manual de cada elemento, e por fim, leitura automática de um arquivo CSV. Um ponto interessante a se observar que para o caso de geração aleatória, utilizamos a função *np.random.randint*, que recebe como parâmetros um número máximo e um número mínimo, além do tamanho desejado do vetor, porém, o valor máximo não é incluso nas possibilidades de valores, assim, para contornar este "problema", para o caso dos inteiros nós adicionamos um no valor máximo, e para o caso dos floats, adicionamos um número extremamente pequeno ($2.3e^{-308}$), de forma que essa adição interferisse o mínimo possível no resultado buscado.

```
def Max(v):
    start_time=time.time()
    a=0
    b=0
    att_value=0
    initial_value=0
    final_value=0
    if max(v)<=0:
        end_time=time.time()
        return max(v), np.where(v==np.amax(v))[0][0],np.where
        (v==np.amax(v))[0][0],v,end_time-start_time
    for i in range(0,len(v)):
        a=max(0,a+v[i])
        if a==0:
            att_value=i+1
        if max(a,b)==a:
            b=a
            initial_value=att_value
            final_value=i
    end_time=time.time()
    return b,initial_value,final_value,v,end_time-start_time
```

Por fim, temos a função Max, a versão em Python do algoritmo relatado no tópico de concepção geral. Tal função é responsável por obter o valor da soma máxima, o elemento de início e o elemento de fim da subsequência ótima, e também o tempo de execução de todo o processo.

3.3 Aplicações Possíveis Para o Problema Proposto:

Através de pesquisas acerca de possíveis aplicações para a busca por uma subsequência adjacente máxima, encontramos alguns problemas onde tal resolução é útil e parte da solução, podemos citar a busca pelo melhor alinhamento entre sequências de DNA, RNA ou proteínas e estudo de sua similaridade através de somas, a aplicação em modelos de planejamento de negócios para averiguar em qual período do ano ocorre o maior número de vendas e entre outros. Assim, de forma geral, constatamos que se trata de um problema que, além de interessante para um estudo acerca de programação dinâmica, também retrata situações e eventos úteis em uma situação real.

Fita de DNA[3]

3.4 Teste Utilizando o Algoritmo Desenvolvido:

Para nossa "bateria de testes", separamos alguns exemplos que pudessem testar o algoritmo desenvolvido em diferentes aspectos, e assim, nos possibilitar a verificação de seu funcionamento pleno e de forma correta. Importante ressaltar que, para averiguarmos a veracidade do resultado encontrado, utilizamos o site https://ostermiller.org/calc/sum.html, uma calculadora automática de subsequências adjacentes máximas.

Assim, para um teste com uma sequência randômica de 10 elementos do tipo inteiro, tivemos o seguinte resultado:

```
Você gostaria que uma sequência fosse gerada randomicamente ou prefere inserir
uma manualmente? 1 para geração randômica e 2 para manual: 1


Qual seria o tamanho da sequência? 10


Os valores da sequência são de quais tipo? 1 para int e 2 para float: 1


Quais seriam os valores limites da sequência?
Inserir limite inferior e superior na ordem: -10 10

Para a sequência:  [-7  2 -5 -3  2 -3  6  5 -4 -1]

Temos como resultado de soma máxima:  11

Tal soma ocorre na subsequência que se inicia no elemento número 7 e termina
no elemento número 8

Subsequência que maximiza a soma:  [6 5]

O processo todo foi tão rápido que a precisão da máquina não
conseguiu registraro tempo corretamente
```

[3]Imagem retirada de https://dglab.com.br/blog/sequenciamento-do-dna/

Com um teste de uma sequência de 100 elementos do tipo float gerados randomicamente, temos o seguinte resultado:

```
Você gostaria que uma sequência fosse gerada randomicamente ou prefere inserir
uma manualmente? 1 para geração randômica e 2 para manual: 1


Qual seria o tamanho da sequência? 100


Os valores da sequência são de quais tipo? 1 para int e 2 para float: 2


Quais seriam os valores limites da sequência?
Inserir limite inferior e superior na ordem: -1 1

Para a sequência:
[-0.16660212 -0.22182652 -0.293259   -0.50525662  0.19955868 -0.25377744
 -0.7913347  -0.41277132  0.73595656 -0.65373665  0.46478545  0.54419921
 -0.06393744 -0.29950873 -0.98656465  0.87532156 -0.50326603 -0.68373717
  0.46750925  0.83917158  0.50801982 -0.79180387 -0.88955336 -0.98873273
 -0.675185   -0.88022838 -0.31209704  0.60898657 -0.15207644  0.7250476
  0.30164623 -0.17017216 -0.26855158 -0.16336951 -0.23037232 -0.35438376
  0.0467722  -0.75315146 -0.34727055 -0.26423106  0.89901842  0.55840444
  0.76818818 -0.51112938  0.9069174  -0.27582277 -0.53600248  0.81729411
  0.55270586 -0.84137039 -0.84578161  0.91621226  0.37909822 -0.39136072
 -0.11628958  0.17846089 -0.31589828  0.36157663 -0.50895223 -0.0219341
  0.63788619 -0.11191349  0.43650194 -0.44277474 -0.15215365 -0.40842935
  0.33582526  0.23403858 -0.64090068 -0.35692329 -0.04883871 -0.88471245
 -0.27502855  0.62470601 -0.40885035 -0.67315885 -0.04953514 -0.01916049
 -0.76709618 -0.13523892  0.98672974 -0.89035172 -0.44362653  0.2643584
  0.6391804  -0.01678172 -0.38629213 -0.19333624  0.53203256 -0.25233678
  0.42060786  0.61694419  0.57629172  0.27205384 -0.26482045  0.12649985
  0.65733425  0.71183306 -0.17992915  0.66485104]

Temos como resultado de soma máxima:  4.18849070596235

Tal soma ocorre na subsequência que se inicia no elemento número 84 e termina
no elemento número 100

Subsequência que maximiza a soma:
[ 0.2643584   0.6391804  -0.01678172 -0.38629213 -0.19333624  0.53203256
 -0.25233678  0.42060786  0.61694419  0.57629172  0.27205384 -0.26482045
  0.12649985  0.65733425  0.71183306 -0.17992915  0.66485104]

O processo todo foi tão rápido que a precisão da máquina não
conseguiu registrar o tempo corretamente
```

Os dois testes acima foram resolvidos de forma extremamente rápida ao ponto de não conseguirmos aferir o tempo, assim, resolvemos executar testes com uma maior quantidade de elementos.

Para um teste em uma sequência com 10000 elementos do tipo float gerados randomicamente, obtivemos o seguinte resultado:

```
Você gostaria que uma sequência fosse gerada randomicamente ou prefere inserir
uma manualmente? 1 para geração randômica e 2 para manual: 1


Qual seria o tamanho da sequência? 10000


Os valores da sequência são de quais tipo? 1 para int e 2 para float: 2


Quais seriam os valores limites da sequência?
Inserir limite inferior e superior na ordem: -100 100

Para a sequência:
[ 29.71135102  42.50796561 -79.42673057 ...  44.59463348  72.11889092
  83.46818254]

Temos como resultado de soma máxima:  6723.346027438372

Tal soma ocorre na subsequência que se inicia no elemento número 6 e termina
no elemento número 4173

Subsequência que maximiza a soma:
[84.6951689  -0.1315653  98.12729703 ... 46.24954249 32.79277313
 94.8769944 ]

O processo todo levou um tempo total de: 0.01562 segundos
```

Para um teste com uma sequência de 50000 elementos do tipo float gerados randomicamente:

```
Você gostaria que uma sequência fosse gerada randomicamente ou prefere inserir
uma manualmente? 1 para geração randômica e 2 para manual: 1


Qual seria o tamanho da sequência? 50000


Os valores da sequência são de quais tipo? 1 para int e 2 para float: 2


Quais seriam os valores limites da sequência?
Inserir limite inferior e superior na ordem: -100 100

Para a sequência:
[ 86.80263237 -30.80948265  67.36413881 ... -14.49234073  95.85800701
  41.78332125]

Temos como resultado de soma máxima:  13357.177415169896

Tal soma ocorre na subsequência que se inicia no elemento número 29106 e
termina no elemento número 38282

Subsequência que maximiza a soma:
[ 74.21760644  94.7988352   77.93207546 ... -49.79460305  96.18706001
  17.19482925]

O processo todo levou um tempo total de: 0.07810 segundos
```

Para um exemplo de sequência com 1000000 de elementos do tipo float gerados randomicamente:

```
Você gostaria que uma sequência fosse gerada randomicamente ou prefere inserir
uma manualmente? 1 para geração randômica e 2 para manual: 1


Qual seria o tamanho da sequência? 1000000


Os valores da sequência são de quais tipo? 1 para int e 2 para float: 2


Quais seriam os valores limites da sequência?
Inserir limite inferior e superior na ordem: -100 100

Para a sequência:
[-96.3852335  -49.47529575  62.59039807 ...  87.71615665 -27.82191716
  97.26641133]

Temos como resultado de soma máxima:  104992.53823852376

Tal soma ocorre na subsequência que se inicia no elemento número 268670 e
termina no elemento número 889892

Subsequência que maximiza a soma:
[ 73.94376917 -70.44070767  79.16300201 ...  59.15161675  27.1660067
  59.98090745]

O processo todo levou um tempo total de: 1.56896 segundos
```

Para um exemplo de sequência com 10000000 de elementos do tipo float gerados randomicamente:

```
Você gostaria que uma sequência fosse gerada randomicamente ou prefere inserir
uma manualmente? 1 para geração randômica e 2 para manual: 1


Qual seria o tamanho da sequência? 10000000


Os valores da sequência são de quais tipo? 1 para int e 2 para float: 2


Quais seriam os valores limites da sequência?
Inserir limite inferior e superior na ordem: -100 100

Para a sequência:
[-49.41275363 -39.49740922 -24.88701938 ...  76.96899746 -91.11850947
  52.08025419]

Temos como resultado de soma máxima:  202789.59070365826

Tal soma ocorre na subsequência que se inicia no elemento número 3590272 e
termina no elemento número 6345932

Subsequência que maximiza a soma:
[82.03491227 40.31932896 36.39133504 ... 89.1956947  37.98501107
 47.30927633]

O processo todo levou um tempo total de: 16.04589 segundos
```

Um teste "extremo"com uma sequência com 500000000 elementos do tipo float gerados randomicamente teve como resultado:

```
Você gostaria que uma sequência fosse gerada randomicamente ou prefere inserir
uma manualmente? 1 para geração randômica e 2 para manual: 1


Qual seria o tamanho da sequência? 500000000


Os valores da sequência são de quais tipo? 1 para int e 2 para float: 2


Quais seriam os valores limites da sequência?
Inserir limite inferior e superior na ordem: -100 100

Para a sequência:
[ -8.16096911  63.66695632  86.03656203 ... -98.86569809  46.57142387
  27.75219957]

Temos como resultado de soma máxima:  1842037.1252040102

Tal soma ocorre na subsequência que se inicia no elemento número 182142019 e
termina no elemento número 499597815

Subsequência que maximiza a soma:
[ 95.2384995  -68.89342445  69.55348756 ...  -2.65881169 -42.83779894
  56.86510679]

O processo todo levou um tempo total de: 783.32848 segundos
```

Com isso, conseguimos verificar que o código implementado teve um desempenho satisfatório, gerando o resultado buscado em um tempo relativamente baixo para o tamanho das sequências avaliadas.

Em seguida, realizamos alguns testes com inserção manual da sequência, como por exemplo, uma sequência de 20 elementos do tipo inteiro:

```
Você gostaria que uma sequência fosse gerada randomicamente ou prefere inserir
uma manualmente? 1 para geração randômica e 2 para manual: 2
A sequência se encontra em um arquivo CSV? 1 para sim e 2 para não: 2

Os valores da sequência são de quais tipo? 1 para int e 2 para float: 1

Insira a sequência :
10 9 20 -50 -500 1000 -20 -35 60 19 98 -1000 200 1400 10000 -10000 99
1000000 20 -1000

Para a sequência:
[     10       9      20     -50    -500    1000     -20     -35      60
      19      98   -1000     200    1400   10000  -10000      99 1000000
      20   -1000]

Temos como resultado de soma máxima:  1001841

Tal soma ocorre na subsequência que se inicia no elemento número 6 e
termina no elemento número 19

Subsequência que maximiza a soma:
[   1000     -20     -35      60      19      98   -1000     200
1400   10000  -10000      99 1000000      20]

O processo todo levou um tempo total de: 0.01562 segundos
```

Para testarmos a importação de dados de arquivos *CSV* por parte do código, geramos sequências aleatórias através do site https://www.omnicalculator.com/statistics/random-number, em seguida, salvamos a sequência aleatória em uma arquivo CSV através do Excel.

Os dados podem ser dispostos em colunas ou em linhas.

	A
1	36
2	-44
3	45
4	-41
5	8
6	-50
7	-82
8	1
9	74
10	32
11	-58
12	29
13	66
14	-37
15	54
16	82
17	19
18	37
19	-15
20	-97
21	47
22	59

Disposição em Coluna

	A	B	C	D	E	F	G	H	I	J	K
1	100	-100	200	300	-1000	1000	800	200	-50	400	1200
2											
3											
4											
5											
6											
7											
8											
9											
10											
11											
12											
13											

Disposição em Linha

Assim, para o CSV com os elementos dispostos em uma linha:

```
Você gostaria que uma sequência fosse gerada randomicamente ou prefere inserir
uma manualmente? 1 para geração randômica e 2 para manual: 2

A sequência se encontra em um arquivo CSV? 1 para sim e 2 para não: 1

Insira aqui o endereço de seu arquivo (sem o .csv):
C:\Users\USUARIO\Desktop\progdinâmica\Teste_2

Para a sequência:
[     100     -100      200      300    -1000     1000      800      200
      -50      400     1200      -90      100     1000  -100000   900000
    -8000    82000  1000000     3000     4999 -1000000]

Temos como resultado de soma máxima:  1981999

Tal soma ocorre na subsequência que se inicia no elemento número 16 e
termina no elemento número 21

Subsequência que maximiza a soma:
[ 900000   -8000   82000 1000000    3000    4999]

O processo todo foi tão rápido que a precisão da máquina não conseguiu
registrar o tempo corretamente
```

E para o CSV com os elementos dispostos em uma coluna:

```
Você gostaria que uma sequência fosse gerada randomicamente ou prefere inserir
uma manualmente? 1 para geração randômica e 2 para manual: 2


A sequência se encontra em um arquivo CSV? 1 para sim e 2 para não: 1


Insira aqui o endereço de seu arquivo (sem o .csv):
C:\Users\USUARIO\Desktop\progdinâmica\Teste_1

Para a sequência:
[  36  -44   45  -41    8  -50  -82    1   74   32  -58   29   66  -37
   54   82   19   37  -15  -97   47   59  -57  -66  -70  -43  -24  -62
  -56  -87   21  -98   89  -84   28   85  100  -93   49   95  -48  -60
  -90  -38   99   61   33  -10  -76   96  -80  -22   75  -34  -35  -28
   40  -63  -18  -86  -95   90  -96   48   18   53  -78  -85   72    4
  -16  -26  -27  -99   77  -36  -13   62   79   67   30  -20   -3   27
  -83  -77  -88   69  -33  -79  -47  -64  -74  -91  -71  -59   88 -100
   35    2   70   55  -69  -54   -4   -8    3   44   58   63   84  -21
   52  -45  -67  -73    7  -40    9   64  -53   60  -12   -9   -1  -52
   98   13   51   16    0   23  -68   -2   41   86   -6  -30  -61  -17
   22   56   46    5  -51   73  -31  -92  -25  -32  -19   11  -65   12
   94   10   25   76  -94    6   15   57   81   17  -46  -75  -49  -23
  -11   80   68   24   91   34   83   65   71   43   92  -81   20  -72
  -14   39   38  -42   -5   78   50  -29  -39   -7   14   26   97   42
  -89   87  -55   93   31]

Temos como resultado de soma máxima:  1068

Tal soma ocorre na subsequência que se inicia no elemento número 99 e
termina no elemento número 201

Subsequência que maximiza a soma:
[ 35   2  70  55 -69 -54  -4  -8   3  44  58  63  84 -21  52 -45 -67 -73
   7 -40   9  64 -53  60 -12  -9  -1 -52  98  13  51  16   0  23 -68  -2
  41  86  -6 -30 -61 -17  22  56  46   5 -51  73 -31 -92 -25 -32 -19  11
 -65  12  94  10  25  76 -94   6  15  57  81  17 -46 -75 -49 -23 -11  80
  68  24  91  34  83  65  71  43  92 -81  20 -72 -14  39  38 -42  -5  78
  50 -29 -39  -7  14  26  97  42 -89  87 -55  93  31]

O processo todo foi tão rápido que a precisão da máquina não conseguiu
registrar o tempo corretamente
```

Com isso, conseguimos concluir que o código funciona plenamente para todos os tipos de entrada e problemas desenvolvidos, permitindo uma maior gama de interação e versatilidade para o usuário.

Importante ressaltar também que damos uma opção ao usuário de salvar os resultados obtidos, isso ocorre através do seguinte segmento de código:

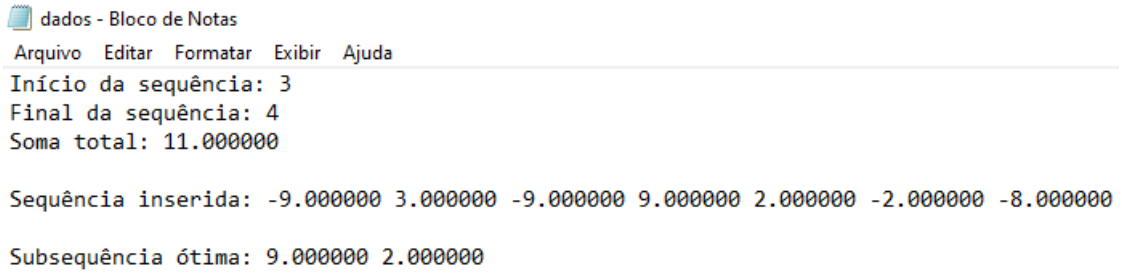
```
1  texto=int(input("Gostaria de salvar um arquivo txt com os dados obtidos?
2  1 para sim e 2 para nao: "))
3  if texto==1:
4      salvar=input("Endereco para salvar o arquivo
5      (sem o nome do mesmo): ")
6      salvar=r"%s\dados.txt" %salvar
7      f=open(salvar,"w+")
8      f.write("Inicio da sequencia: %d" %initial_value)
9      f.write("\nFinal da sequencia: %d" %final_value)
10     f.write("\nSoma total: %d" %Sum)
11     f.write("\n")
12     f.write("\nSequencia inserida:")
13     f.writelines([" %d" %c for c in vector])
```

```
14      f.write("\n")
15      f.write("\nSubsequencia otima:")
16      f.writelines([" %d" %d for d in vector[initial_value:final_value+1]])
17      f.close()
```

Com um exemplo de sequência com 20 elementos do tipo inteiro gerados randomicamente:

```
Você gostaria que uma sequência fosse gerada randomicamente ou prefere inserir
uma manualmente? 1 para geração randômica e 2 para manual: 1


Qual seria o tamanho da sequência? 20


Os valores da sequência são de quais tipo? 1 para int e 2 para float: 1


Quais seriam os valores limites da sequência?
Inserir limite inferior e superior na ordem: -10 10

Para a sequência:
[-9  3 -9  9  2 -2 -8  5 -3 -9  3  7 -8  6 -9 -8  6  3  1 -8]

Temos como resultado de soma máxima:  11

Tal soma ocorre na subsequência que se inicia no elemento número 4 e
termina no elemento número 5

Subsequência que maximiza a soma:  [9 2]

O processo todo foi tão rápido que a precisão da máquina não conseguiu
registrar o tempo corretamente

Gostaria de salvar um arquivo txt com os dados obtidos?
1 para sim e 2 para não: 1

Endereço para salvar o arquivo (sem o nome do mesmo):
C:\Users\USUARIO\Desktop\progdinâmica
```

Geramos o seguinte arquivo “dados.txt”:



Arquivo de Texto Gerado Pelo Programa

4 O Problema do Caminho Máximo em Uma Matriz:

4.1 Concepção Geral da Resolução:

Como um extra, resolvemos abordar como seria o problema da subsequência máxima em uma espécie de possibilidade "2D", ou seja, se as subsequências possíveis fossem dispostas em uma matriz de dimensão m X n. Assim, concebemos que a partir de uma dada coordenada (i, j), existem três possibilidades de passo seguinte, $(i+1, j)$,$(i+1, j+1)$ e $(i+1, j-1)$, com $j \leq n$ e $i \leq m$, ou seja, o algoritmo sempre está em um movimento de descida na matriz, reta ou em diagonal. Importante lembrar que a ideia é percorrer todas as linhas da matriz, traçando um caminho ótimo, assim, o algoritmo sempre irá se iniciar na primeira linha da matriz de entrada.

A base do nosso algoritmo é a definição de variáveis que recebem os possíveis passos seguintes, e, a partir disso, chamamos a função recursivamente, gerando cada vez problemas menores onde a solução converge para o buscado pelo problema maior. Em dado momento, percebemos que alguns pontos eram utilizados em mais de um caminho possível, com isso, tivemos a ideia de armazenar os pontos em um dicionário conforme o algoritmo era executado, o que economizou boa parte do processamento do problema e tornou o código extremamente mais rápido. Importante ressaltar que a escolha pelo armazenamento em um dicionário teve também como motivação a ideia de que, atualmente, em um geral armazenamento é extremamente mais barato que processamento.

4.2 Algoritmo em Python:

Assim como no código proposto anteriormente, visamos implementar um código que cumprisse de forma satisfatória a solução do problema proposto, permitindo também uma ampla gama de interações e possibilidades para o usuário, gerando uma ampla variedade de análises e aplicações.

A seguir, nosso código implementado em Python separado função a função:

```
import math as mt
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import copy as cp
import time
```

Primeiramente, as bibliotecas que utilizamos para o desenvolvimento do programa, começando pela *math*, de onde conseguimos extrair o conceito de infinito para utilizarmos em algumas comparações importantes durante o algoritmo, em seguida, a *numpy*, essencial para que conseguissemos trabalhar plenamente com matrizes; a biblioteca *time* novamente foi utilizada para nos auxiliar na obtenção do tempo de execução do programa; a *copy* nos permitiu realizar cópia de variáveis de forma adequada a manter as mesmas sem vínculos diretos de alteração entre si, por fim, *pandas* foi utilizado para o carregamento de arquivos *CSV* e *matplotlib* foi utilizado para a plotagem do caminho na matriz de forma visual.

```
A,Result,total_time,Sum=Initialize()
print("\nPara a seguinte matriz: ")
print("\n")
print(A)
if Sum==0:
    print("\nA soma para o melhor caminho e menor ou igual a zero, sendo
    assim, nao existe nenhuma escolha otima.")
else:
    print("\nTemos como caminho que maximiza a soma o seguinte: ")
    if len(Result)>60:
        for i in range(0,30):
            print((i+1,Result[i]+1),"= ",A[i,Result[i]])
        print("\n    .")
        print("    .")
        print("    .")
        print("\n")
        for k in range(len(Result)-1-30,len(Result)):
            print((k+1,Result[k]+1),"= ",A[k,Result[k]])

    else:
        for i in range(len(Result)):
            print((i+1,Result[i]+1),"= ",A[i,Result[i]])
```

```
    print("\nCom a seguinte soma maxima: ",Sum)
    Show(A,Result)
    if total_time!=0:
        print("\nTempo total para a execucao completa do processo:
        ",total_time,"segundos")
    else:
        print("\nO Processo foi tao rapido que a precisao da maquina
        nao conseguiu registrar o tempo")
```

Ao contrário do primeiro problema, nossa central de controle nesse funciona mais como um meio de "apresentação"dos resultados do que de interação por parte do usuário. Aqui é onde configuramos tudo de forma a gerar uma visualização completa e adequada dos resultados obtidos.

```
def Initialize():
   h=int(input("Gostaria que a matriz com os valores fosse gerada
   randamicamente ou atraves da insercao do usuario?
   1 para geracao randomica e 2 para insercao manual: "))
   C={}
   if h==1:
      b=int(input("\nOs valores da matriz seriam de qual tipo?
      1 para int e 2 para float: "))
      k=[int(item) for item in
      input("\nInsira as dimensoes da matriz na ordem linha e depois
      coluna: ").split()]
      w=[float(item) for item in input("\nQuais seriam os
      valores limites?
      Inserir o limite inferior e superior na ordem: ").split()]
      if b==1:
          A=np.random.randint(w[0],w[1]+1,size=(k[0],k[1]))
      else:
          A=np.random.uniform(w[0],w[1]+2.3e-308,size=(k[0],k[1]))
      B=0
      aux=0
      for j in range(0,A.shape[1]):
          D=Max(A,0,j,C)[0]
          if D>B:
              B=D
              aux=j
      Result,total_time=Travel(A,aux,C)
      return A,Result,total_time,B
   else:
       g=int(input("Os dados se encontram em um arquivo CSV?
       1 para sim e 2 para nao: "))
       if g==1:
           h=input("Endereco para o arquivo (sem o .csv): ")
           h=h+".csv"
           l=pd.read_csv(h,sep=';',header=None)
           A=np.array(l.values)
           B=0
           aux=0
           for j in range(0,A.shape[1]):
               D=Max(A,0,j,C)[0]
               if D>B:
                   B=D
                   aux=j
           Result,total_time=Travel(A,aux,C)
           return A,Result,total_time,B
       else:
           b=int(input("\nOs valores da matriz seriam de qual tipo?
           1 para int e 2 para float: "))
           k=[int(item) for item in
           input("\nInsira as dimensoes da matriz na ordem linha
           e depois coluna: ").split()]
           if b==1:
               v=np.array(list(map(int,
               input("\nInsira os valores da matriz
               (Ao escrever o numero de termos correspondente ao de
```

```
                colunas, a linha ja e trocada automaticamente):
                ").split())))
            else:
                v=np.array(list(map(float,
                input("\nInsira os valores da matriz
                (Ao escrever o numero de termos correspondente ao de
                colunas, a linha ja e trocada automaticamente):
                ").split())))
            A=v.reshape(k[0],k[1])
            B=0
            aux=0
            for j in range(0,A.shape[1]):
                D=Max(A,0,j,C)[0]
                if D>B:
                    B=D
                    aux=j
            return A,Travel(A,aux,C),B
```

Nossa função *Initialize* é responsável por todo processo de preparo da matriz da forma mais adequada a ser utilizada no programa, é onde também oferecemos ao usuário as diferentes possibilidades de entradas para os dados, permitindo uma ampla versatilidade e aplicabilidade. Importante ressaltar que é também onde escolhemos a coluna j inicial de forma automatizada, tal coluna é a que permite obtermos o melhor caminho possível entre todos os existentes dentro da matriz.

```
def Max(X,i,j,C):
    if (i,j) in C:
        return C[(i,j)]
    a=X
    m=X.shape[0]
    n=X.shape[1]
    if i==m-1:
      return (a[i,j],j)
    else:
      k=j-1
      l=j+1
      x=Max(a,i+1,j,C)
      if k>=0:
          y=Max(a,i+1,k,C)
      else:
          y=(-mt.inf,k)
      if l<(n):
          z=Max(a,i+1,l,C)
      else:
          z=(-mt.inf,l)
      c=a[i,j]+max(x[0], y[0], z[0])
      if max(x[0], y[0], z[0])==x[0]:
          C[(i,j)]=(c,j)
          return (c,j)
      elif max(x[0], y[0], z[0])==y[0]:
          C[(i,j)]=(c,k)
          return (c,k)
      else:
          C[(i,j)]=(c,l)
          return (c,l)
```

Em seguida, nossa função Max, a mais importante e responsável por a partir de uma dada coordenada (i, j), obter o passo seguinte e a soma máxima obtida no caminho ótimo. Como é possível visualizar, a base de funcionamento dela é recursiva, o que gera problemas menores que possuem resultados que levam ao resultado ótimo do problema original.

```
def Travel(X,j,C):
    start_time=time.time()
    m=X.shape[0]
    A=[j]
    k=0
    while k<m-1:
        A.append(Max(X,k,j,C)[1])
        j=Max(X,k,j,C)[1]
        k=k+1
    end_time=time.time()
    return A,end_time-start_time
```

A função *Travel* é a responsável por, a partir do resultado obtido na função *Max*, construir o caminho ótimo da matriz oferecida para análise.

```
def Show(A,Result):
    B=cp.deepcopy(A)
    k=0
    for i in range(B.shape[0]):
        a=Result[k]
        for j in range(B.shape[1]):
                if j!=a:
                    B[i,j]=0
                else:
                    B[i,j]=1000
        k=k+1
    plt.matshow(B)
    plt.show()
```

Por fim, temos a função *Show*, responsável por gerar uma plotagem do caminho ótimo que pode ser visualizada pelo usuário.

Assim, conseguimos concluir que a implementação do algoritmo foi plenamente satisfatória, gerando um código versátil, intuitivo, extremamente rápido para o que se propõe, e principalmente, gera resultados corretos.

4.3 Aplicações Possíveis Para o Problema Proposto:

Durante nossa pesquisa para a realização do projeto, nos deparamos com inúmeros problemas que tem como solução um algoritmo como o que implementamos; na área de processamento de imagens por exemplo, tal algoritmo é muito utilizado para a realização de cortes e compressões em imagens de forma a minimizar a perda de informação (a única diferença para o nosso código, é que em um geral, se prefere retirar os pixels menos "luminosos", ou seja, se busca o caminho mínimo), em planejamento de negócios que podem ser modelados em uma matriz e o resultado ótimo é um caminho (Um trajeto de vendas buscando um lucro máximo, por exemplo), mas, o mais interessante em nossa visão, foi a aplicabilidade desse modelo para o aprimoramento de inteligências artificiais em jogos; algumas usam um parâmetro de buscar "rastros"do personagem do usuário de forma a derrotá-lo, sendo que, os caminhos possíveis são estabelecidos em um mapeamento 2D que remete a um caminho dentro de uma matriz.

Exemplo de Aplicação do Problema Proposto[4]

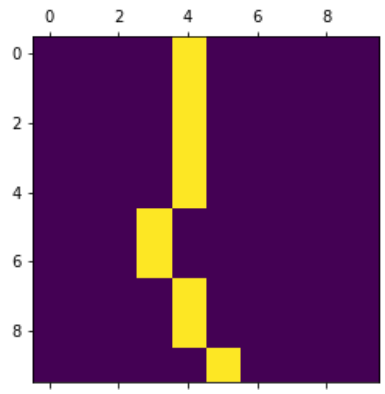
[4]Imagem retirada de https://www.revide.com.br/blog/sheldon-assis/e-possivel-conciliar-jogos-eletronicos-e-educacao/

4.4 Teste Utilizando o Algoritmo Desenvolvido:

Assim como no programa anteriormente abordado, fizemos uma "bateria de teste"que nos permitisse analisar a versatilidade, eficiência e velocidade do algoritmo implementado na resolução do problema proposto. Para a verificação dos resultados, utilizamos o site tutorialspoint.dev/algorithm/dynamic-programming-algorithms/maximum-sum-path-matrix-top-bottom.

Assim, para um primeiro caso de matriz quadrada 10 X 10 gerada randomicamente com elementos do tipo inteiro, obtemos:

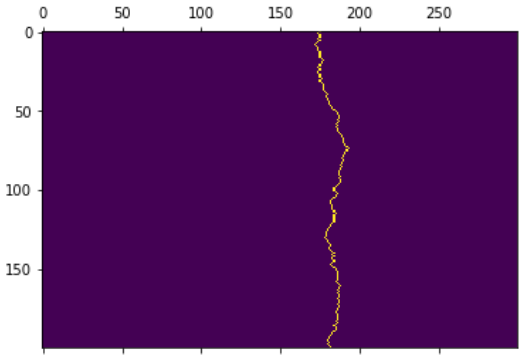
```
Gostaria que a matriz com os valores fosse gerada randômicamente ou através
da inserção do usuário? 1 para geração randômica e 2 para inserção manual: 1

Os valores da matriz seriam de qual tipo? 1 para int e 2 para float: 1

Insira as dimensões da matriz na ordem linha e depois coluna: 10 10

Quais seriam os valores limites?
Inserir o limite inferior e superior na ordem: -100 100

Para a seguinte matriz:

[[ 66   9 -14 -82  36  31 -20  92 -48   0]
 [-72   4 -55 -96  71 -78 -95  -6  30 -43]
 [-41  75 -13 -17  29 -44  14 -23 -76 -22]
 [-94 -49 -69  88  93  72  59 -46  14 -93]
 [-71 -58 -75 -89  16  77 -15   9 -17 -84]
 [ 54 -69  25  99  -6  -2  18  42  29  51]
 [-68 -72 -95  44 -11  10 -16  89  17  -2]
 [ 71  55 -98  20  24 -22 -52 -83 -31 -18]
 [-32  80  57  94  77 -28  17  72  25   5]
 [-17  93 -69  31 -59  72  -5  87 -20  25]]

Temos como caminho que maximiza a soma o seguinte:
(1, 5) =  36
(2, 5) =  71
(3, 5) =  29
(4, 5) =  93
(5, 5) =  16
(6, 4) =  99
(7, 4) =  44
(8, 5) =  24
(9, 5) =  77
(10, 6) =  72
Com a seguinte soma máxima:  561

O Processo foi tão rápido que a precisão da máquina não conseguiu
registrar o tempo
```



Caminho Gerado na Matriz 10 X 10

Para uma matriz de elementos do tipo float gerada randomicamente com dimensões 200 X 300:

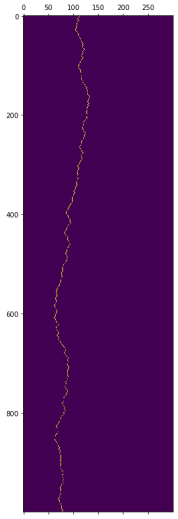
```
Gostaria que a matriz com os valores fosse gerada randômicamente ou através
da inserção do usuário? 1 para geração randômica e 2 para inserção manual: 1

Os valores da matriz seriam de qual tipo? 1 para int e 2 para float: 2

Insira as dimensões da matriz na ordem linha e depois coluna: 200 300

Quais seriam os valores limites?
Inserir o limite inferior e superior na ordem: -100 100

Para a seguinte matriz:

[[-75.16638514  64.81866317  -5.12949879 ...  50.36225386  72.36440372
  -33.93188951]
 [-32.74075576  11.17643436  10.75953347 ... -22.64959871  80.77976547
   74.8588221 ]
 [ -8.41038558 -39.55448938  95.07893341 ...   2.57500337  85.04914292
  -89.54806537]
 ...
 [-41.52147829  13.6305947   28.66164406 ... -14.56784447 -10.18012685
   97.91900562]
 [ 39.44950166  35.35987347  31.33054316 ... -50.32432932  19.49868673
  -57.38661566]
 [ 39.9313197  -21.07706321  15.20829143 ...  65.28930776  41.96003474
   91.48983568]]

Temos como caminho que maximiza a soma o seguinte:
(1, 177) =  92.42601540309877
(2, 176) =  72.23324930292787
(3, 175) =  93.07278453205328
(4, 176) =  40.26681425741657
(5, 176) =  99.42228621156875
    .
    .
    .
(195, 182) =  37.00136963005599
(196, 181) =  92.64013266278138
(197, 182) =  41.96807758338946
(198, 181) =  87.81090417539014
(199, 182) =  99.24281258887882
(200, 183) =  66.46668455850931
Com a seguinte soma máxima:  12918.69174296123

O Processo foi tão rápido que a precisão da máquina não conseguiu
registrar o tempo
```



Caminho Gerado na Matriz 200 X 300

Para uma matriz de elementos do tipo inteiro gerada randomicamente com dimensões 1000 X 300:

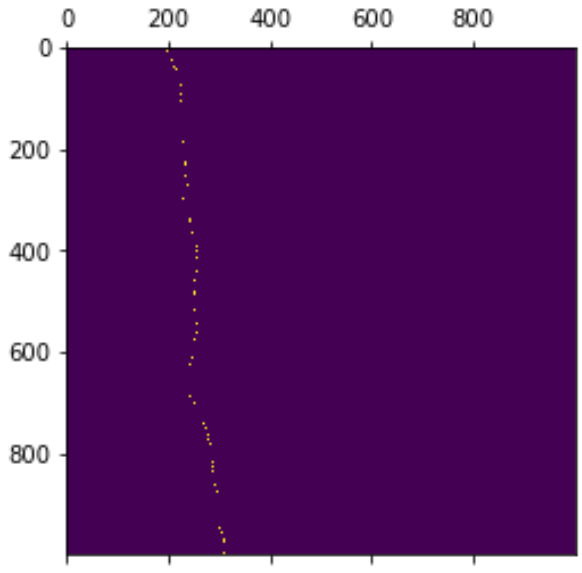
```
Gostaria que a matriz com os valores fosse gerada randômicamente ou através
da inserção do usuário? 1 para geração randômica e 2 para inserção manual: 1

Os valores da matriz seriam de qual tipo? 1 para int e 2 para float: 1

Insira as dimensões da matriz na ordem linha e depois coluna: 1000 300

Quais seriam os valores limites?
Inserir o limite inferior e superior na ordem: -100 100

Para a seguinte matriz:

[[-13  -9 -34 ...  24 -46  -8]
 [-20 -27 -27 ...  59  32   9]
 [ 78   0  39 ... -89 -84 -54]
 ...
 [ 82  78 -35 ...  87  68  36]
 [-96 -28   5 ...  61 -22 -38]
 [ 64 -14  89 ... -51 -83  -8]]

Temos como caminho que maximiza a soma o seguinte:
(1, 114) =  97
(2, 113) =  9
(3, 112) =  78
(4, 113) =  81
(5, 112) =  42
    .
    .
    .
(995, 79) =  74
(996, 80) =  53
(997, 79) =  100
(998, 79) =  96
(999, 79) =  87
(1000, 80) =  62

Com a seguinte soma máxima:  64392

O Processo foi tão rápido que a precisão da máquina não conseguiu
registrar o tempo
```



Caminho Gerado na Matriz 1000 X 300

Para uma matriz de elementos do tipo inteiro gerada randomicamente com dimensões 1000 X 1000:

```
Gostaria que a matriz com os valores fosse gerada randômicamente ou através
da inserção do usuário? 1 para geração randômica e 2 para inserção manual: 1

Os valores da matriz seriam de qual tipo? 1 para int e 2 para float: 1

Insira as dimensões da matriz na ordem linha e depois coluna: 1000 1000

Quais seriam os valores limites?
Inserir o limite inferior e superior na ordem: -100 100

Para a seguinte matriz:

[[ 38  16  67 ...  92  28   2]
 [ 97 -67 -63 ...  72  40 -49]
 [ 87  30  10 ... -25  63  56]
 ...
 [ 60 -14 -33 ... -29  94   6]
 [  5 -73   5 ... -98   6  76]
 [ 50  51   1 ...  35  24  54]]

Temos como caminho que maximiza a soma o seguinte:
(1, 199) =  95
(2, 199) =  69
(3, 200) =  35
(4, 201) =  16
(5, 200) =  61
    .
    .
    .
(995, 309) =  67
(996, 309) =  72
(997, 310) =  86
(998, 311) =  68
(999, 310) =  99
(1000, 311) =  73

Com a seguinte soma máxima:  64577

Tempo total para a execução completa do processo:  0.015625 segundos
```

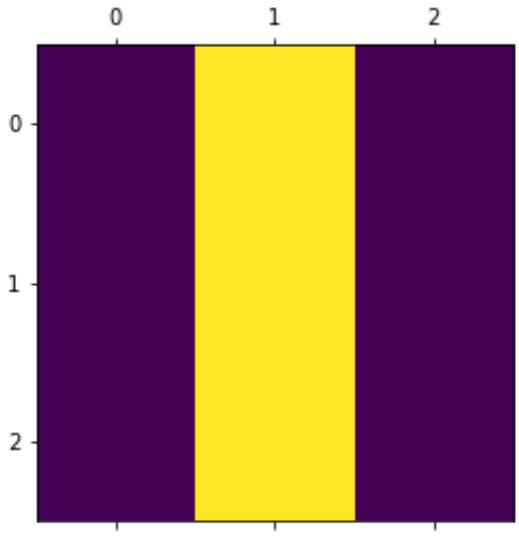


Caminho Gerado na Matriz 1000 X 1000

Com êxito no teste para matrizes geradas randomicamente, decidimos então testar as matrizes inseridas manualmente.

Considerando uma matriz 3 X 3 de elementos do tipo float inserida manualmente:

```
Gostaria que a matriz com os valores fosse gerada randômicamente ou através
da inserção do usuário? 1 para geração randômica e 2 para inserção manual: 2

Os dados se encontram em um arquivo CSV? 1 para sim e 2 para não: 2

Os valores da matriz seriam de qual tipo? 1 para int e 2 para float: 2

Insira as dimensões da matriz na ordem linha e depois coluna: 3 3

Insira os valores da matriz (Ao escrever o número de termos correspondente
ao de colunas, a linha já é trocada automaticamente):
1.1 5.6 -9.3 4.5 6.9 -100 7.8 79 9.4

Para a seguinte matriz:

[[   1.1    5.6   -9.3]
 [   4.5    6.9 -100. ]
 [   7.8   79.     9.4]]

Temos como caminho que maximiza a soma o seguinte:
(1, 2) =  5.6
(2, 2) =  6.9
(3, 2) =  79.0

Com a seguinte soma máxima:  91.5

O Processo foi tão rápido que a precisão da máquina não conseguiu
registrar o tempo
```



Caminho Gerado na Matriz 3 X 3

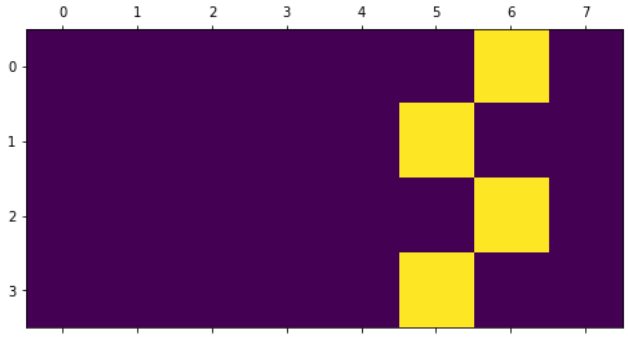
Em seguida, testamos um arquivo CSV que possui a seguinte disposição de elementos:

	A	B	C	D	E	F	G	H
1	1	2	-10	2	4	20	122	-10
2	1	3	6	8	6	5	4	3
3	2	-5	2	1	-6	2	122	4
4	100	4	3	2	1	-100	-200	-200

Exemplo de Elementos Dispostos em um Arquivo CSV

Assim, tomando o arquivo CSV representado acima:

```
Gostaria que a matriz com os valores fosse gerada randômicamente ou através
da inserção do usuário? 1 para geração randômica e 2 para inserção manual: 2

Os dados se encontram em um arquivo CSV? 1 para sim e 2 para não: 1

Endereço para o arquivo (sem o .csv):
C:\Users\USUARIO\Desktop\progdinâmica\Teste_1_matriz

Para a seguinte matriz:

[[   1    2  -10    2    4   20  122  -10]
 [   1    3    6    8    6    5    4    3]
 [   2   -5    2    1   -6    2  122    4]
 [ 100    4    3    2    1 -100 -200 -200]]

Temos como caminho que maximiza a soma o seguinte:
(1, 7) =  122
(2, 6) =  5
(3, 7) =  122
(4, 6) =  -100

Com a seguinte soma máxima:  149

O Processo foi tão rápido que a precisão da máquina não conseguiu
registrar o tempo
```



Caminho Gerado na Matriz 4 X 8

Com isso, concluímos que todos os meios de interação do usuário com o código foram efetivos, e principalmente, o código gerou resultados corretos e em um tempo extremamente pequeno até mesmo para casos com dimensões gigantescas. Sendo assim, podemos dizer que se trata de uma boa implementação com um ótimo desempenho.

Importante ressaltar que, assim como no algoritmo anterior, é possível salvar os resultados em um arquivo de texto, isso é possibilitado pela seguinte segmento de código:

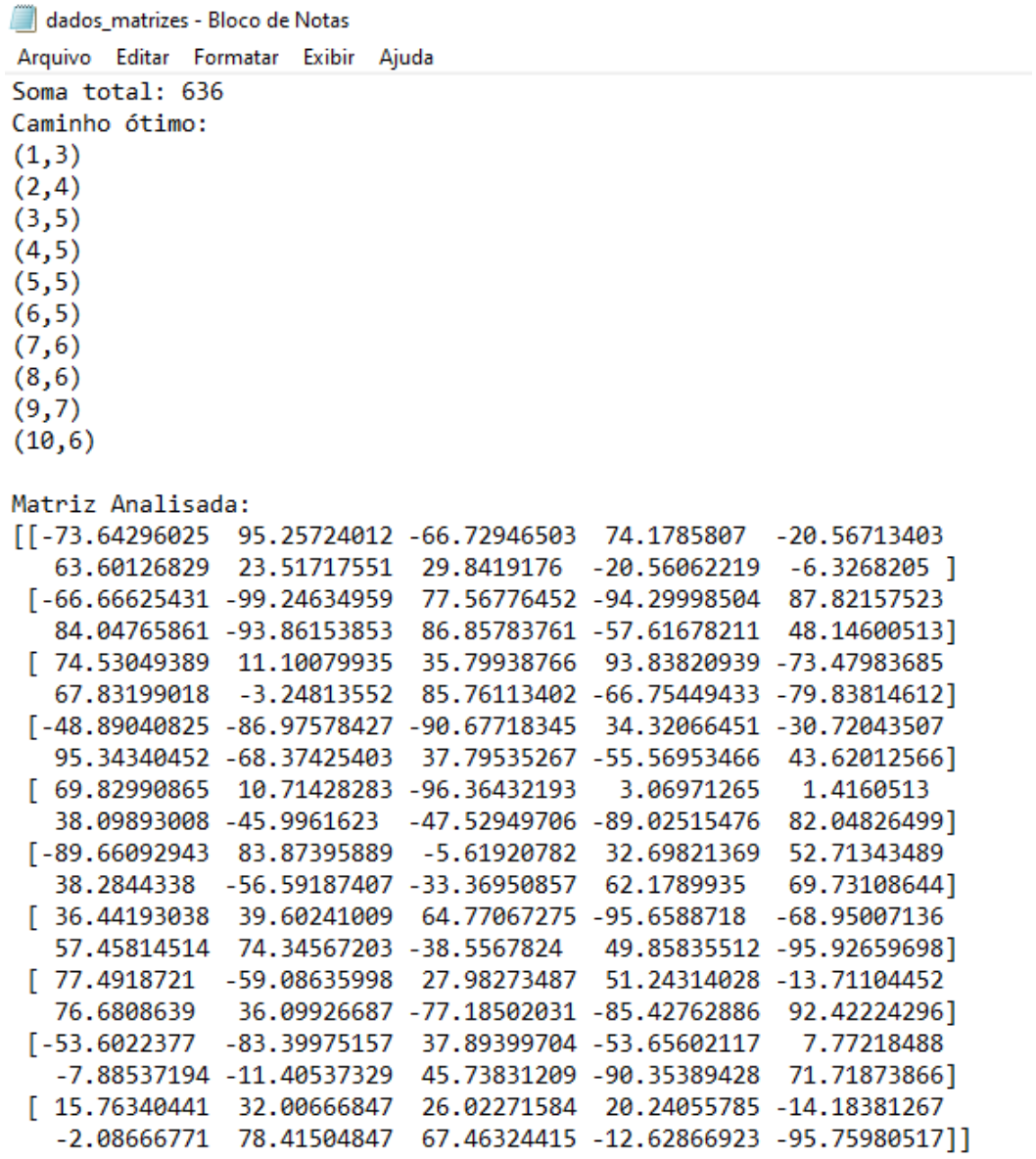
```
texto=int(input("Gostaria de salvar um arquivo txt com os dados obtidos?
1 para sim e 2 para nao: "))
    if texto==1:
        salvar=input("Endereco para salvar o arquivo
        (sem o nome do mesmo): ")
        salvar=r"%s\dados_matrizes.txt" %salvar
        f=open(salvar,"w+")
        f.write("Soma total: %d" %Sum)
        f.write("\nCaminho otimo:")
        f.writelines(["\n(%d,%d)"
        %(d+1,Result[d]) for d in range(len(Result))])
        f.write("\n")
        f.write("\nMatriz Analisada:")
        f.write("\n")
        f.write(str(A))
```

```
16          f.close()
17          print("\nSeus dados foram salvos como dados_matrizes.txt :)")
```

Assim, para uma matriz $10X10$ com elementos do tipo float gerados randomicamente:

```
Gostaria que a matriz com os valores fosse gerada randômicamente ou através
da inserção do usuário? 1 para geração randômica e 2 para inserção manual: 1

Os valores da matriz seriam de qual tipo? 1 para int e 2 para float: 2

Insira as dimensões da matriz na ordem linha e depois coluna: 10 10

Quais seriam os valores limites? Inserir o limite inferior e superior na ordem:
-100 100

Para a seguinte matriz:

[[-73.64296025  95.25724012 -66.72946503  74.1785807  -20.56713403
   63.60126829  23.51717551  29.8419176  -20.56062219  -6.3268205 ]
 [-66.66625431 -99.24634959  77.56776452 -94.29998504  87.82157523
   84.04765861 -93.86153853  86.85783761 -57.61678211  48.14600513]
 [ 74.53049389  11.10079935  35.79938766  93.83820939 -73.47983685
   67.83199018  -3.24813552  85.76113402 -66.75449433 -79.83814612]
 [-48.89040825 -86.97578427 -90.67718345  34.32066451 -30.72043507
   95.34340452 -68.37425403  37.79535267 -55.56953466  43.62012566]
 [ 69.82990865  10.71428283 -96.36432193   3.06971265   1.4160513
   38.09893008 -45.9961623  -47.52949706 -89.02515476  82.04826499]
 [-89.66092943  83.87395889  -5.61920782  32.69821369  52.71343489
   38.2844338  -56.59187407 -33.36950857  62.1789935   69.73108644]
 [ 36.44193038  39.60241009  64.77067275 -95.6588718  -68.95007136
   57.45814514  74.34567203 -38.5567824   49.85835512 -95.92659698]
 [ 77.4918721  -59.08635998  27.98273487  51.24314028 -13.71104452
   76.6808639   36.09926687 -77.18502031 -85.42762886  92.42224296]
 [-53.6022377  -83.39975157  37.89399704 -53.65602117   7.77218488
   -7.88537194 -11.40537329  45.73831209 -90.35389428  71.71873866]
 [ 15.76340441  32.00666847  26.02271584  20.24055785 -14.18381267
   -2.08666771  78.41504847  67.46324415 -12.62866923 -95.75980517]]

Temos como caminho que maximiza a soma o seguinte:
(1, 4) =  74.1785806957144
(2, 5) =  87.82157522607912
(3, 6) =  67.83199017585991
(4, 6) =  95.34340451605308
(5, 6) =  38.0989300799167
(6, 6) =  38.284433802412394
(7, 7) =  74.34567203254625
(8, 7) =  36.09926687471517
(9, 8) =  45.73831209068945
(10, 7) =  78.41504846606844
Com a seguinte soma máxima:  636.1572139600548

O Processo foi tão rápido que a precisão da máquina não conseguiu
registrar o tempo

Gostaria de salvar um arquivo txt com os dados obtidos?
1 para sim e 2 para não: 1

Endereço para salvar o arquivo (sem o nome do mesmo):
C:\Users\USUARIO\Desktop\progdinâmica

Seus dados foram salvos como dados_matrizes.txt :)
```

Geramos o seguinte arquivo de texto:



Exemplo de Arquivo de Texto

5 Conclusão:

Através deste projeto conseguimos ter uma grande compreensão de como a programação dinâmica é extremamente importante para inúmeros problemas, facilitando processos que demandariam tempo e esforço em níveis exaustivos. Durante as pesquisas que realizamos, encontramos inúmeras aplicações para os problemas tratados, o que expandiu nossa compreensão e noção de como a programação dinâmica é um campo grande e demasiadamente útil em vários âmbitos da sociedade.

Importante também ressaltar que a implementação de tais algoritmos aumentam nosso contato e experiência com programação, instigando a busca por diferentes soluções, bibliotecas que podem ser aplicadas e entre outros.

Em um geral, foi extremamente satisfatória fazer o projeto e obter êxito em nossas metas.

6 Bibliografia:

[1] Richard Bellman. Dynamic Programming.

[2] https://www.hackerearth.com/practice/algorithms/dynamic-programming/introduction-to-dynamic-programming-1/tutorial/

[3] https://www.tutorialspoint.com/data_structures_algorithms/dynamic_programming.htm

[4] https://leetcode.com/tag/dynamic-programming/

[5] https://blog.gft.com/br/2021/08/31/dynamic-programming-memoization/

[6] Anotações e aulas do curso MS515 (Métodos Probabilísticos em Pesquisa Operacional) ministrado pelo professor Lúcio Tunes do Santos