Universidade Estadual de Maringá
Departamento de Informática
Curso: Informática
Disciplina: 5184 – Projeto e Análise de Algoritmos
Professor: Daniel Kikuti

UVA

116 – Uniderecional TSP

Discente:

Nome: Eduardo H. K. Shibukawa RA: 62003

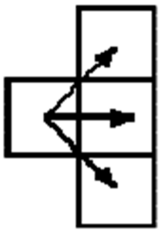
Maringá

Sumário

1. Solução 2
2. Demonstração de sobreposição de problemas. 3
3. Análise de complexidade do algoritmo sem usar programação dinâmica e usando programação dinâmica. 4
4. Proposta de uma escolha gulosa para o problema. 5
5. Tempos de execução dos algoritmos. 6

Background

Problemas que exigem caminhos mínimos em algum domínio aparecem em diversas áreas da Ciência da Computação. Por exemplo, uma das restrições em problemas de roteamento de VLSI é minimizar o comprimento das trilhas. O Problema do Caixeiro Viajante (TSP) - descobrir se todas as cidades na rota de um vendedor podem ser visitadas apenas uma vez com um limite especificado no tempo de viagem - é um dos exemplos canônicos de um problema NP-completo; soluções parecem exigir uma enorme quantidade de tempo para serem geradas, mas são fáceis de verificar. Esse problema lida com encontrar um caminho mínimo através de uma grade de pontos durante a viagem somente de esquerda para a direita.

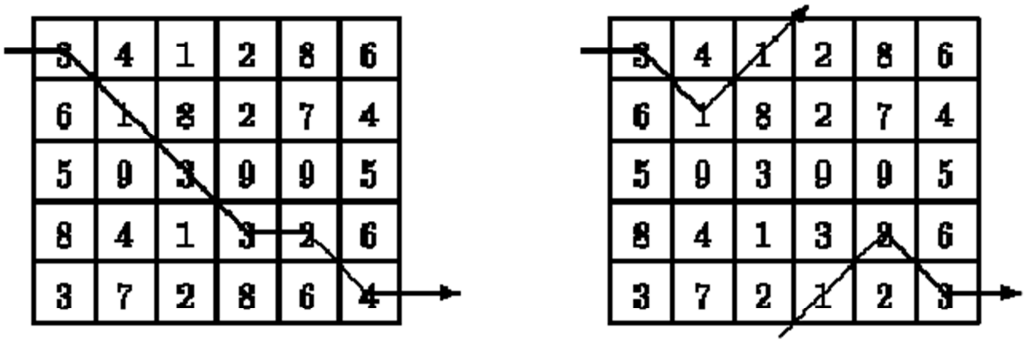


O Problema

Dada uma matriz *m x n* de inteiros, você deve escrever um programa que calcula um caminho de peso mínimo. Um caminho começa em qualquer lugar na coluna 1 (primeira coluna) e consiste de uma sequência de passos que terminam na coluna n (última coluna). Um passo consiste em viajar de coluna i para coluna i + 1 em uma linha (horizontal ou diagonal) adjacente. As primeiras e últimas linhas (linhas 1 e m) de uma matriz são considerados adjacentes, ou seja, a matriz se "enrola" tal que ela representa
um cilindro horizontal. Passos legais são ilustrados abaixo.
O peso de um caminho é a soma dos números inteiros em cada uma das n células

da matriz que são visitadas. Por exemplo, duas matrizes 5x6 ligeiramente diferentes são

mostradas abaixo (a única diferença são os números da linha inferior).



1. Solução

Para solucionarmos este problema temos que procurar **todos os caminhos possiveis** entre as **colunas 1 e m** e então pegar o com menor a soma dos pesos.

$$S(n,m) = \begin{cases} 0, & n = 0 \\ M[n,m] + Min\big(S(n \quad 1,m), S(n \quad 1,m+1), S(n \quad 1,m \quad 1)\big) & n > 0 \end{cases}$$

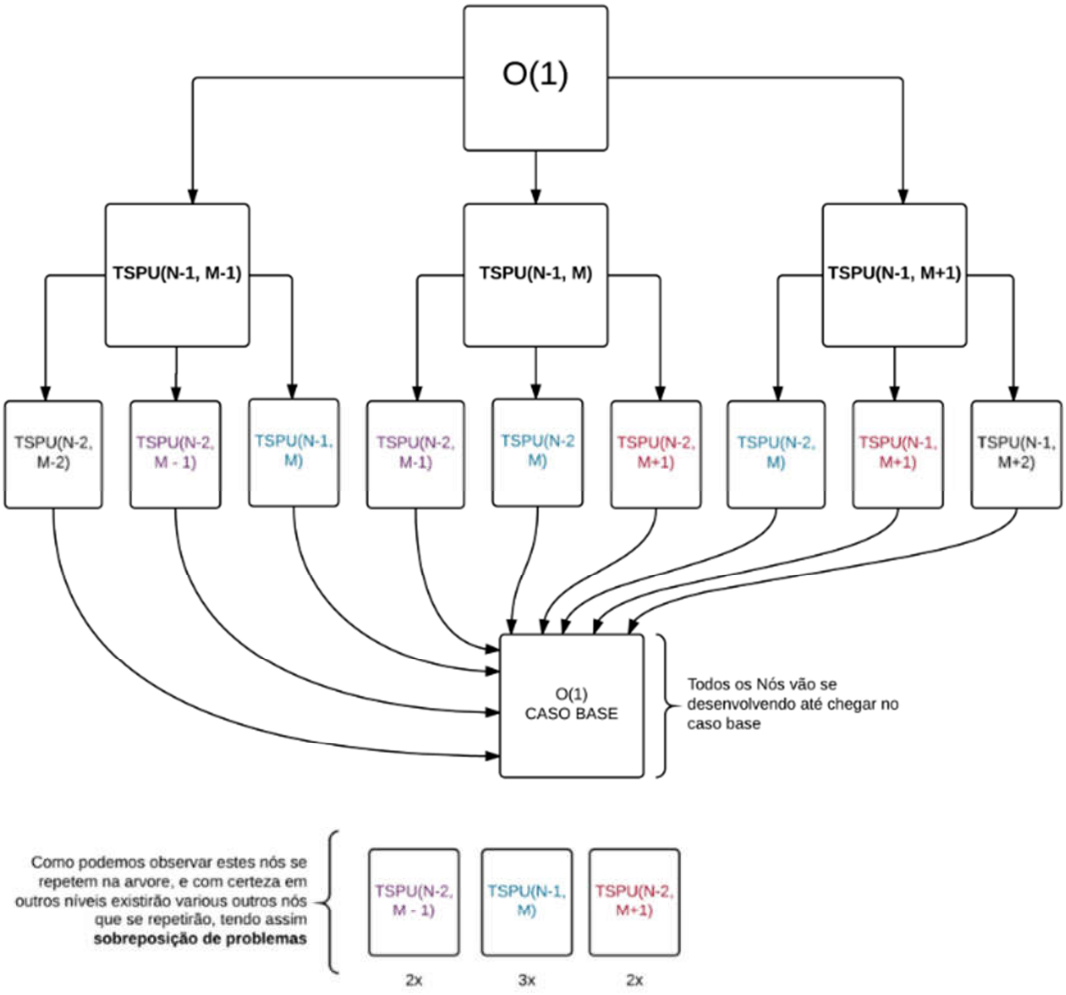
Algortimo recursivo de força bruta

```
def                    TSPU(M,                    n,                    m)
 if                      n                     ==                    0:
     return                                                            0
 else M[n,m]  + Min(TSPU(n-1,  m),TSPU(n-1,  m+1),TSPU(n-1,  m-1))
```

Agora o mesmo algoritmo utilizando programação dinamica

```
def TSPU(n,  m):
    if                memo[n,m]                ==                INFINITO:
        if                    n                    ==                    0:
            memo[n,m]  = 0
        else:
            memo[n,m]  = M[n,m]
                +   Min(TSPU(n-1,   m),TSPU(n-1,   m+1),TSPU(n-1,   m-1))
    return memo[n,m]
```

2. Demonstração de sobreposição de problemas.



3. Análise de complexidade do algoritmo sem usar programação dinâmica e usando programação dinâmica.

Sem usar a programação dinamica temos que pela arvore de sobreposição demonstrada acima

$$TSPU(n,m) = TSPU(n \quad 1,m) + TSPU(n \quad 1,m \quad 1) + TSPU(n \quad 1,m+1) + O(1)$$

Podemos ver que pra cada chamada o algoritmo irá chamar ele mesmo 3 vezes até chegar o caso base onde n = 0, então vamos simplificar a solução para

$$TSPU(n) = 3TSPU(n \quad 1) + O(1)$$

Com isso podemos chegar a complexidade

$$O(TSPU) = 3^n$$

Levando em conta que temos contar o M também chegamos a algo como $O(TSPU) = 3^{n+m}$

Utilizando a programação dinamica nós vamos ignorar os casos repetidos, que como pudemos verificar na arvore de sobreposição são varios, e assim no final resolveriamos apenas os casos onde eles não foram resolvidos ficaremos com uma complexidade parecida com

$$O(TSPU) = O(NxM)$$

4. Proposta de uma escolha gulosa para o problema.

Uma proposta para a solução do problema por escolha gulosa seria, ir para o o caminho de menor custos dos 3 próximos possíveis, pois é a ideia para resolvermos o problema na PD.

Porém como podemos provar com um contra exemplo citado abaixo, está proposta não é valida:

Escolha Gulosa

```
5 6
3 4 1 2 8 6
6 1 8 2 7 4
5 9 3 9 9 5
8 4 1 3 2 6
3 7 2 8 6 4
```

Custo da solução = 17

Programação Dinamica

```
5 6
3 4 1 2 8 6
6 1 8 2 7 4
5 9 3 9 9 5
8 4 1 3 2 6
3 7 2 8 6 4
```

Custo da solução = 16

Como podemos ver o algoritmo da solução gulosa não trouxe a solução otima.

5. Tempos de execução dos algoritmos.

Abaixo temos os tempos de execuções dos algoritmos e suas respectivas saídas

```
------------------------
Arquivo: tsp1.txt
Tempo Exec: 0.000499725341796875
---------SAIDA----------
1 2 3 4 4 5
16
---------ESPERADO-------
1 2 3 4 4 5
16
------------------------
Arquivo: tsp2.txt
Tempo Exec: 0.0005002021789550781
---------SAIDA----------
1 2 1 5 4 5
11
---------ESPERADO-------
1 2 1 5 4 5
11
------------------------
Arquivo: tsp3.txt
Tempo Exec: 0.0005004405975341797
---------SAIDA----------
2 1
19
---------ESPERADO-------
1 1
19
------------------------
Arquivo: tsp4.txt
Tempo Exec: 0.000499725341796875
---------SAIDA----------
2 1 3
3
---------ESPERADO-------
2 1 3
3
------------------------
Arquivo: tsp5.txt
Tempo Exec: 0.000499725341796875
---------SAIDA----------
3 3 4
19
---------ESPERADO-------
3 3 4
19
------------------------
Arquivo: tsp6.txt
Tempo Exec: 0.000499725341796875
---------SAIDA----------
1 1 1 1 1
15
---------ESPERADO-------
1 1 1 1 1
15
------------------------
Arquivo: tsp7.txt
Tempo Exec: 0.0005004405975341797
---------SAIDA----------
1 1 1 2 2
9
---------ESPERADO-------
1 1 1 2 2
9
------------------------
Arquivo: tsp8.txt
Tempo Exec: 0.0005004405975341797
---------SAIDA----------
1
1
---------ESPERADO-------
1
1
------------------------
Arquivo: tsp9.txt
Tempo Exec: 0.000499725341796875
---------SAIDA----------
1 4 1 4
4
---------ESPERADO-------
1 4 1 4
4
```

```
------------------------
Arquivo: tsp10.txt
Tempo Exec: 0.0005021095275878906
---------SAIDA----------
4 3 2 3 3 4
-49
---------ESPERADO-------
4 3 2 3 3 4
-49
------------------------
Arquivo: tsp11.txt
Tempo Exec: 0.001999378204345703
---------SAIDA----------
7 6 6 6 6 5 4 4 3 3 2 1 1 1 1 2 3 3 2 3
357
---------ESPERADO-------
7 6 6 6 6 5 4 4 3 3 2 1 1 1 1 2 3 3 2 3
357
------------------------
Arquivo: tsp12.txt
Tempo Exec: 0.001999378204345703
---------SAIDA----------
9 9 8 9 10 9 9 8 7 7 6 6 7 7 6 5 4 4 5 4
-1391
---------ESPERADO-------
9 9 8 9 10 9 9 8 7 7 6 6 7 7 6 5 4 4 5 4
-1391
------------------------
Arquivo: tsp13.txt
Tempo Exec: 0.011002063751220703
---------SAIDA----------
4 4 5 6 7 6 6 7 7 8 7 7 7 8 8 7 8 8 7 6 6 5 5 4 3 2 3 4 3 3 3 3 3 3 2 2 1 10 10 1 10 10 1 2 2 3 2 2 3
2 3 3 4 5 6 7 6 6 6 5 6 5 5 4 4 4 4 4 4 4 3 2 3 4 5 5 4 5 5 6 5 6 7 6 5 5 6 6 5 6 5 6 7 7 6 6 7 6 5 5 6
19706
---------ESPERADO-------
4 4 5 6 7 6 6 7 7 8 7 7 7 8 8 7 8 8 7 6 6 5 5 4 3 2 3 4 3 3 3 3 3 3 2 2 1 10 10 1 10 10 1 2 2 3 2 2 3
2 3 3 4 5 6 7 6 6 6 5 6 5 5 4 4 4 4 4 4 4 3 2 3 4 5 5 4 5 5 6 5 6 7 6 5 5 6 6 5 6 5 6 7 7 6 6 7 6 5 5 6
19706
------------------------
Arquivo: tsp14.txt
Tempo Exec: 0.010003328323364258
---------SAIDA----------
9 8 8 7 6 5 6 7 7 7 6 6 7 6 7 7 6 6 6 5 6 5 5 4 4 4 3 4 5 5 6 6 5 5 4 3 2 1 1 10 1 2 1 10 1 2 3 2 3 2
1 10 10 10 9 10 9 10 10 10 1 1 2 2 3 4 4 5 5 4 4 3 3 4 4 3 2 2 3 3 2 3 3 4 3 2 3 3 2 2 3 2 1 2 3 3 2
1 10 10
-59457
---------ESPERADO-------
9 8 8 7 6 5 6 7 7 7 6 6 7 6 7 7 6 6 6 5 6 5 5 4 4 4 3 4 5 5 6 6 5 5 4 3 2 1 1 10 1 2 1 10 1 2 3 2 3 2
1 10 10 10 9 10 9 10 10 10 1 1 2 2 3 4 4 5 5 4 4 3 3 4 4 3 2 2 3 3 2 3 3 4 3 2 3 3 2 2 3 2 1 2 3 3 2
1 10 10
-59457
------------------------
Arquivo: tsp15.txt
Tempo Exec: 0.005001068115234375
---------SAIDA----------
2 1 1 2 1 10 9 10 10 10 1 2 3 4 5 6 7 8 8 8 9 10 9 8 7 7 7 8 8 9 10 9 9 10 1 2 1 10 1 10 9 9 10 1 1
10 10 9 8 7
933
---------ESPERADO-------
2 1 1 2 1 10 9 10 10 10 1 2 3 4 5 6 7 8 8 8 9 10 9 8 7 7 7 8 8 9 10 9 9 10 1 2 1 10 1 10 9 9 10 1 1
10 10 9 8 7
933
------------------------
Arquivo: tsp16.txt
Tempo Exec: 0.0050046443939208984
---------SAIDA----------
2 1 1 2 1 10 9 10 10 10 1 2 3 4 5 6 7 8 8 8 9 10 9 8 7 7 7 8 8 9 10 1 2 3 4 4 4 3 3 3 3 3 4 4 3 3 3 4
5 5
19
---------ESPERADO-------
2 1 1 2 1 10 9 10 10 10 1 2 3 4 5 6 7 8 8 8 9 10 9 8 7 7 7 8 8 9 10 1 2 3 4 4 4 3 3 3 3 3 4 4 3 3 3 4
5 5
19
```