

# به نام خدا

## پروژه تحلیل و طراحی الگوریتم

استاد : دکتر داریوش نجفی

افراد گروه :

امیرحسین داداش زاده - 97463199

امیرحسین صفری - 97463137

## الگوریتم Traveling salesman

در این الگوریتم سعی بر این است که همه ی راس های گراف با کمترین هزینه پیمایش شوند.

برای اینکار ابتدا تابع `make_row_column_infint` را برای تبدیل سطر ها و ستون های داده شده به `infinite` استفاده شده است، در الگوریتم اصلی از بی نهایت استفاده میشود در این کد به جای یک عدد بزرگ، از `1` استفاده شده است. (در این کد از کتابخانه ی `pickle` برای خواندن و نوشتن در فایل استفاده شده است ولی نیازی به نصب ندارد.)

```
126 def make_row_column_infint(matrix, row, column):
127     i = row
128     for j in range(5):
129         matrix[i][j] = -1
130
131     j = column
132     for i in range(5):
133         matrix[i][j] = -1
134
135     matrix[column][row] = -1
136
```

تابع `print_matrix` وظیفه ی چاپ آرایه ی ورودی را بر عهده دارد.


```
137 def print_matrix(matrix):
138     for i in range (5):
139         print(matrix[i])
140
```

تابع Reducer وظیفه‌ی محاسبه‌ی هزینه‌ی هر راس گراف را بر عهده دارد.

```
141 def reducer(matrix):
142     min_row:int = -1
143     cost = 0
144
145     #-----
146     #   recution in rows
147     #-----
148     for i in range(5):
149
150         for j in range(5):
151             if matrix[i][j] == -1:
152                 continue
153             else:
154                 if min_row == -1:           #first not -1 number
155                     min_row = matrix[i][j]
156                 elif matrix[i][j] < min_row:
157                     min_row = matrix[i][j] #find min
158
159             #print(min_row)
160             for j in range(5):
161                 if matrix[i][j] == -1:
162                     continue
163                 else:
164                     matrix[i][j] = matrix[i][j] - min_row
165
166             if min_row != -1:
167                 cost += min_row
168
169             min_row = -1
170
```

```
171 #-----
172 #   recution in columns
173 #-----
174 min_column:int = -1
175
176 for j in range(5):
177     for i in range(5):
178         if matrix[i][j] == -1:
179             continue
180         else:
181             if min_column == -1:           #first not -1 number
182                 min_column = matrix[i][j]
183             elif matrix[i][j] < min_column:
184                 min_column = matrix[i][j] #find min
185
186     for i in range(5):
187         if matrix[i][j] == -1:
188             continue
189         else:
190             matrix[i][j] = matrix[i][j] - min_column
191
192     if min_column != -1:
193         cost += min_column
194     min_column = -1
195
196 return cost
197
198
```

در تابع `main`، منطق کد پیاده سازی شده است که ماتریس ورودی را از فایل دریافت کرده سپس ماتریس های کمکی `previous_level_matrix` و `reduced_matrix` برای مراحل هر طبقه از درخت استفاده شده است. در کد پایین مقادیر اولیه تعریف و `cast` است.

```
salesman >  salesman.py
1  import pickle
2  def main():
3      #-----
4      # initialization arrays
5      #-----
6      main_matrix = []
7      previous_level_matrix = []
8      reduced_matrix = []
9      main_costs = {
10     }
11     vertex = {
12     }
13     seq = []
14
15     for i in range(5):
16         vertex[i+1] = False
17         main_costs[i+1] = 0
18     #-----
19     # filling array by inputs
20     #-----
21     input_file = open("salesman.txt", "r")
22     for line in input_file:
23         elements = line.split()
24         main_matrix.append(elements)
25
26     main_matrix = [list( map(int,i) ) for i in main_matrix]    #initial all array elements to int
27
```

منطق اصلی در حلقه ی های تو در تو ی کد زیر است، که در هر مرحله هزینه ی راس ها را حساب کرده و با مقایسه با یکدیگر مینیمم را یافته و آن راس را به عنوان راس بعدی انتخاب کرده و از آن راس هزینه ی بقیه ی راس ها به غیر از راس قبلی را محاسبه کرده و این روند تکرار می شود.

```
52     while level_counter < 5:
53         while vertex_counter <= 5:
54             if vertex[vertex_counter] == False:
55
56                 main_matrix = []
57                 input_file = open("salesman.txt", "r")
58                 for line in input_file:
59                     elements = line.split()
60                     main_matrix.append(elements)
61
62                 main_matrix = [list( map(int,i) ) for i in main_matrix]
63                 reducer(main_matrix)
64
65                 #make_row_column_infint(reduced_matrix, last_varified_vertex-1, vertex_counter-1)
66                 i = last_varified_vertex-1
67                 for j in range(5):
68                     reduced_matrix[i][j] = -1
69
70                 j = vertex_counter-1
71                 for i in range(5):
72                     reduced_matrix[i][j] = -1
73
74                 reduced_matrix[vertex_counter-1][last_varified_vertex-1] = -1
75
76                 #print_matrix(reduced_matrix)
77                 cost = reducer(reduced_matrix)
78                 print(cost)
79                 cost += main_matrix[last_varified_vertex-1][vertex_counter-1]
80                 print(main_matrix[last_varified_vertex-1][vertex_counter-1])
81                 cost += main_costs[last_varified_vertex]
82                 print(main_costs[last_varified_vertex])
```

نمونه ی خروجی به همراه ماتریس های هر راس در مراحل

```
////////////////////////////////////
5
1
25
[-1, -1, -1, -1, -1]
[10, -1, 9, 0, -1]
[0, 3, -1, 0, -1]
[12, 0, 9, -1, -1]
[-1, 0, 0, 12, -1]
////////////////////////////////////
level_counter :1
0
3
25
[-1, -1, -1, -1, -1]
[12, -1, 11, -1, 0]
[0, -1, -1, -1, 2]
[-1, -1, -1, -1, -1]
[11, -1, 0, -1, -1]
////////////////////////////////////
0
12
25
[-1, -1, -1, -1, -1]
[12, -1, -1, -1, 0]
[0, 3, -1, -1, 2]
[-1, -1, -1, -1, -1]
[11, 0, -1, -1, -1]
////////////////////////////////////
```

```
PS D:\Uni\5\Algorithm project\salesman> python .\salesman.py
0
10
25
[-1, -1, -1, -1, -1]
[-1, -1, 11, 2, 0]
[0, -1, -1, 0, 2]
[15, -1, 12, -1, 0]
[11, -1, 0, 12, -1]
////////////////////////////////////
11
17
25
[-1, -1, -1, -1, -1]
[1, -1, -1, 2, 0]
[-1, 3, -1, 0, 2]
[4, 3, -1, -1, 0]
[0, 0, -1, 12, -1]
////////////////////////////////////
0
0
25
[-1, -1, -1, -1, -1]
[12, -1, 11, -1, 0]
[0, 3, -1, -1, 2]
[-1, 3, 12, -1, 0]
[11, 0, 0, -1, -1]
////////////////////////////////////
```

```

////////////////////////////////////
level_counter :3
0
0
28
[-1, -1, -1, -1, -1]
[-1, -1, -1, -1, -1]
[0, -1, -1, -1, -1]
[-1, -1, -1, -1, -1]
[-1, -1, -1, -1, -1]
////////////////////////////////////
level_counter :4
[-1, 10, 17, 0, 1]
[12, -1, 11, 2, 0]
[0, 3, -1, 0, 2]
[15, 3, 12, -1, 0]
[11, 0, 0, 12, -1]
#####
Final sequence:
[1, 4, 2, 5, 3]
costs of vertexes:
{1: 25, 2: 28, 3: 28, 4: 25, 5: 28}
total cost: 134

```

```

////////////////////////////////////
11
0
25
[-1, -1, -1, -1, -1]
[1, -1, 0, -1, -1]
[0, 3, -1, -1, -1]
[-1, -1, -1, -1, -1]
[11, 0, 0, -1, -1]
////////////////////////////////////
level_counter :2
13
11
28
[-1, -1, -1, -1, -1]
[-1, -1, -1, -1, -1]
[0, -1, -1, -1, 0]
[-1, -1, -1, -1, -1]
[0, -1, -1, -1, -1]
////////////////////////////////////
0
0
28
[-1, -1, -1, -1, -1]
[-1, -1, -1, -1, -1]
[0, -1, -1, -1, -1]
[-1, -1, -1, -1, -1]
[11, -1, 0, -1, -1]
////////////////////////////////////

```

جواب نهایی:

```

#####
Final sequence:
[1, 4, 2, 5, 3]
costs of vertexes:
{1: 25, 2: 28, 3: 28, 4: 25, 5: 28}
total cost: 134

```

نحوه ی خروجی گرفتن: ابتدا وارد فایل salesman شوید سپس در ترمینال عبارت python salesman.py را وارد کنید.