# 程式設計期末考實作題

## 流程圖

#### STEP 1:

根據所提供的城市資訊, 我們可以以建立城市之間 的距離矩陣,並使用最短 路徑演算法來尋找最方便 的旅行方式程序。

首先,讓我們列出城市之 間的距離矩陣:

### STEP 2:

接下來,我們可以使用最 短路徑算法(Dijkstra算 法)找到最方便的旅行程 序。在這裡,我們可以使 用Python來實現此算法:

# 計算兩城市間的直線距離

	Α	В	С	D	Е	F	G↔
Α	0	1200	1175	1200	1200	2230	2580₽
В	1200	0	930	970	1200	2470	2880₽
С	1175	930	0	1225	1280	1880	1350₽
D	1200	970	1225	0	1330	1830	2880₽
E	1200	1200	1280	1330	0	1930	1810⊬
F	2230	2470	1880	1830	1930	0	850₽
G	2580	2880	1350	2880	1810	850	0⊷

# 計算兩城市間的直線距離 def calculate\_distance(city1, city2):

x1, y1 = city1[0], city1[1] x2, y2 = city2[0], city2[1] return ((x2 - x1) \*\* 2 + (y2 - y1) \*\* 2) \*\* 0.5

#### **STEP 3:**

# 計算兩城市間的直線距離

```
#最短路徑演算法
def find_cheapest_path(graph, start):
 num_cities = len(graph)
 distances = [sys.maxsize] * num_cities
 visited = [False] * num_cities
 path = [None] * num_cities
distances[start] = 0
 for _ in range(num_cities):
min_distance = sys.maxsize
   min index = -1
   for i in range(num_cities):
     if not visited[i] and distances[i] <</pre>
min_distance:
min_distance = distances[i]
min_index = i
visited[min_index] = True
   for j in range(num_cities):
     if (
       not visited[j]
       and graph[min_index][j] != sys.maxsize
       and distances[min_index] +
graph[min_index][j] < distances[j]</pre>
     ):
distances[j] = distances[min_index] +
graph[min_index][j]
path[i] = min index
 return distances, path
```

#### **STEP 4:**

# 城市資訊

#### STEP 2:

# 建立城市之間的 距離矩陣

#### # 城市資訊

```
cities = {
    "A": [(0, 0), 2000],
    "B": [(0, 120), 2400],
    "C": [(30, 100), 2400],
    "D": [(80, 0), 1600],
    "E": [(45, 10), 1500],
    "F": [(90, 160), 2200],
    "G": [(120, 200), 1200],
}
```

```
# 建立城市之間的距離矩陣
distances = [
  [0, 1200, 1175, 1200, 1200, 2230, 2580],
  [1200, 0, 930, 970, 1200, 2470, 2880],
  [1175, 930, 0, 1225, 1280, 1880, 1350],
  [1200, 970, 1225, 0, 1330, 1830, 2880],
  [1200, 1200, 1280, 1330, 0, 1930, 1810],
  [2230, 2470, 1880, 1830, 1930, 0, 850],
  [2580, 2880, 1350, 2880, 1810, 850, 0],
]
```

#### **STEP 4:**

# 計算最便宜的 旅遊行程

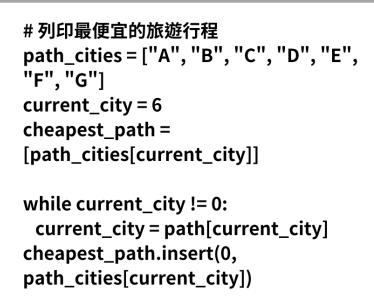
#### **STEP 5:**

# 列印最便宜的 旅遊行程

#### STEP 6:

# 列印結果

# 計算最便宜的旅遊行程 distances, path = find\_cheapest\_path(distances, 0)



# 列印結果
print("最便宜的旅遊行程是:"+"->
".join(cheapest\_path))
print("總共花費的旅費為:"+
str(distances[6] \* 100))
# 油錢以每公里100元計算

最便宜的旅遊行程是: A->G->E->B->D->C->F 總共花費的旅費為:892500