TSP & Shortest path

HongSumin

Agenda

- Problem
- TSP / Shortest path
- Code Shortest path
- Code TSP – Permutation
- Dynamic Programming
- Code TSP – Dynamic Programming
- Q&A

Problem

Problem

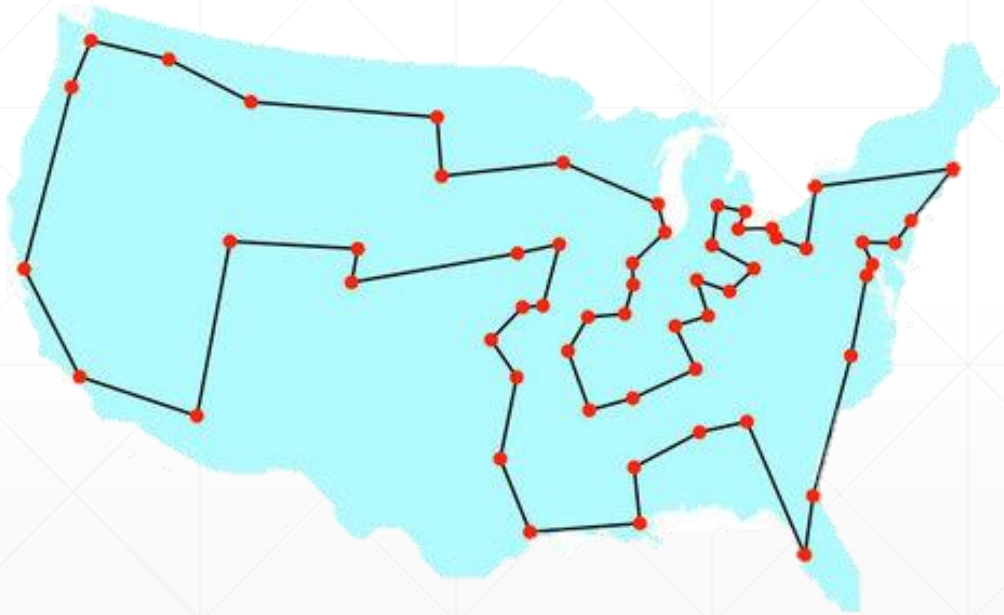
출발 \ 도착	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	0	3	5	48	48	8	8	5	5	3	3	-1	3	5	8	8	5
2	3	0	3	48	48	8	8	5	5	-1	-1	3	-1	3	8	8	5
3	5	3	0	72	72	48	48	24	24	3	3	5	3	-1	48	48	24
4	48	48	74	0	-1	6	6	12	12	48	48	48	48	74	6	6	12
5	48	48	74	-1	0	6	6	12	12	48	48	48	48	74	6	6	12
6	8	8	50	6	6	0	-1	8	8	8	8	8	8	50	-1	-1	8
7	8	8	50	6	6	-1	0	8	8	8	8	8	8	50	-1	-1	8
8	5	5	26	12	12	8	8	0	-1	5	5	5	5	26	8	8	-1
9	5	5	26	12	12	8	8	-1	0	5	5	5	5	26	8	8	-1
10	3	-1	3	48	48	8	8	5	5	0	-1	3	-1	3	8	8	5
11	3	-1	3	48	48	8	8	5	5	-1	0	3	-1	3	8	8	5
12	-1	3	5	48	48	8	8	5	5	3	3	0	3	5	8	8	5
13	3	-1	3	48	48	8	8	5	5	-1	-1	3	0	3	8	8	5
14	5	3	-1	72	72	48	48	24	24	3	3	5	3	0	48	48	24
15	8	8	50	6	6	-1	-1	8	8	8	8	8	8	50	0	-1	8
16	8	8	50	6	6	-1	-1	8	8	8	8	8	8	50	-1	0	8
17	5	5	26	12	12	8	8	-1	-1	5	5	5	5	26	8	8	0

- Path size provided
- Find Shortest Path
  - Choose start & end city
  - Find shortest path each city
- TSP
  - Travel Salesman Problem
  - Uses a two-dimensional array of numbers entered

TSP / Shortest Path(1/3)

- TSP
  - several cities and the cost
  - moving from one city to another
  - visit all cities only once
  - seek the order of minimum cost travel back to the original starting point

TSP / Shortest Path(2/3)



- Permutation Algorithm
  - List the number of all possible cases
  - Calculate the distance of all cases and select the small one
- DP (Dynamic Programming)
  - solve complex problems by dividing them into simple multiple problems

Code

Code : Main

```
if __name__ == '__main__':
    Cmap = cityMap()
    mn = menu()
    if mn == 1:
        start, end = shortestMenu()
        st = time.time()
        print(findShortestCase(start, end, Cmap))
        print(time.time()-st)

    elif mn == 2:
        n = TSPMenu()
        mp = TSPMap(n)
        st = time.time()
        print("%d의 최단경로" % n, TSP(mp))
        # permute(5)
        # print(distanceCalc(permute(10), Cmap))
        print(time.time() - st)
```

- Main
  - Make city map
  - Get user's input and move to right menu

Code : Menu (1/3)

```
def cityMap():
    f = open("city.txt", 'r', encoding='UTF8')
    lines = f.readlines()
    cityMap = [[int(lines[i].split()[n]) for n in range(17)] for i in range(17)]

    return cityMap
```

- City Map
  - Open file and read lines
  - Make 2-D list

Code : Menu (2/3)

```python
def menu():
    while True:
        try:
            mn = int(input("Shortest path(1) or Tsp(2)? Please input the m
            if mn == int(1):
                return mn
                break
            elif mn == int(2):
                return mn
                break
            else:
                raise ValueError

        except:
            print("Please input the right value")
```

```python
def shortestMenu():
    while True:
        try:
            ans = list(map(int, input("Please input the start & end. ex) 7 11").split()))
            return ans
            break
        except:
            print("Please input the right value")
```

- Menu
  - Get menu value
  - Get start and end city of shortest path

Code : Menu (3/3)

```
def TSPMenu():
    while True:
        try:
            num = int(input("Please input the TSP size(3~17)"))
            if num < 3 or num > 17:
                raise ValueError
            else:
                return num
                break
        except:
            print("Please input the right value")
```

```
def TSPMap(num):
    mparr = np.array(cityMap())
    mp = list(map(list, mparr[:num, :num]))
    return mp
```

- TSP Menu
  - Get TSP size
  - By the size value, determine the map's size

Code : Shortest path (1/3)

```
def findShortestCase(start, end, distance):
    shortest = ["", 999]

    def searchCase(start, end, length,tmplist):

return searchCase(start, end, 0, "")
```

- Find Shortest Case
  - Get start, end, city map(distance)
  - Return searchCase
  - Start searchCase recursion

Code : Shortest path (2/3)

```
def searchCase(start, end, length,tmplist):
    start -= 1
    end -= 1
    # print("시작:", start, "끝:", end, "현재경로:", tmplist, "길이
    if tmplist == "":
        tmplist += "%d" % (start+1)
    # print("현재경로", tmplist, "길이", length)

    if distance[start][end] == -1 or distance[start][end] ==0:
        # print("경로가 없거나 자신일때", start, end)
        return

    if distance[start][end] != -1:
        cmprLength =  length + distance[start][end]
        cmprList = tmplist + ",%d" % (end+1)
        # print("도착지까지의 길이:", cmprLength, "도착지까지의 경로:
        if cmprLength < shortest[1]:
            # print("Shortest에 저장", "현재 길이:",cmprLength, "
            shortest[0] = cmprList
            shortest[1] = cmprLength
```

- SearchCase
  - Tmplist = path to now
  - Start -1, end -1 to index
  - When start recursion, add tmplist to start city
  - If no path between start & end city or start city is same with end, stop searching
  - if there's path, Creates a path that reaches the end and compares it to the currently stored minimum path

Code : Shortest path (3/3)

```
for i in range(17):
    # print("현재 길이:", length, "현재 경로", tmplist, "현재 개수", i+1)
    if start != i and distance[start][i] != -1 and length+distance[start][i] < shortest[1] and distance[start][i] != 0:
        nlength = length + distance[start][i]
        nlist = tmplist+ ",%d" % (i+1)
        searchCase(i+1, end+1, nlength, nlist)
    else:
        continue
return shortest
```

- Searching different route
  - Ex) start 3 end 4, 3,4 -> 3,1,4 ->, 3,1,2,4 -> …..
  - Go to next searchCase when using i case is right
    - The start and the new point are different
    - path is existed
    - now distance is shorter than now shortest route
  - Return shortest route

Code : TSP using permutation

```python
import itertools

def permute(num):
    # n<=10 일 때만 사용가능
    inpList = [x + 1 for x in range(num)]
    permutation = list(itertools.permutations(inpList))
    # print(permutation)
    # permutation = list(map(lambda x: ''.join(x), perm
    return permutation
```

- Make permutation
  - Using itertools, put components in inpList
  - If num =6 , inpList = [1,2,3,4,5,6]
  - Make permutations

Code : TSP using permutation

```
def distanceCalc(list, distance):
    shortest = [" ", 999999]
    for prmt in list:
        flg = True
        tmpdis = 0
        # print("reset",tmpdis)
        for idx, i in enumerate(prmt):
            # print("i", i)
            if i == prmt[-1]:
                if distance[int(i)-1][int(prmt[0])-1] == -1:
                    flg = False
                else:
                    tmpdis += distance[int(i)-1][int(prmt[0])-1]
                break
            # print("distance",cityDist(int(i),int(prmt[idx+1])),
            if distance[int(i)-1][int(prmt[idx + 1])-1] == -1:
                flg = False
                break
            tmpdis += distance[int(i)-1][int(prmt[idx + 1])-1]
        if tmpdis < shortest[1] and flg:
            shortest[0] = prmt
            shortest[1] = tmpdis

    return shortest
```

- Calculate distance
  - Set shortest arbitrarily
  - Calculates the distance for each permutation
  - If permutation is end
    - End -> Start path is existed, pass
  - path exists and is shorter than the current shortest path
    - renew the shortest path

Limit of Permutation Method

N=10

```
Shortest path(1) or Tsp(2)? Please input the menu number.2
[(1, 2, 3, 10, 8, 4, 6, 5, 7, 9), 57]
8.416694402694702
```

N=11

```
Shortest path(1) or Tsp(2)? Please input the menu number.2
[(1, 2, 3, 10, 8, 4, 6, 5, 7, 9, 11), 60]
89.18377304077148
```

- Time Complexity
  - N<=10 , permutation method is okay
  - N>10 , running time is too long
- To solve this -> Dynamic Programming

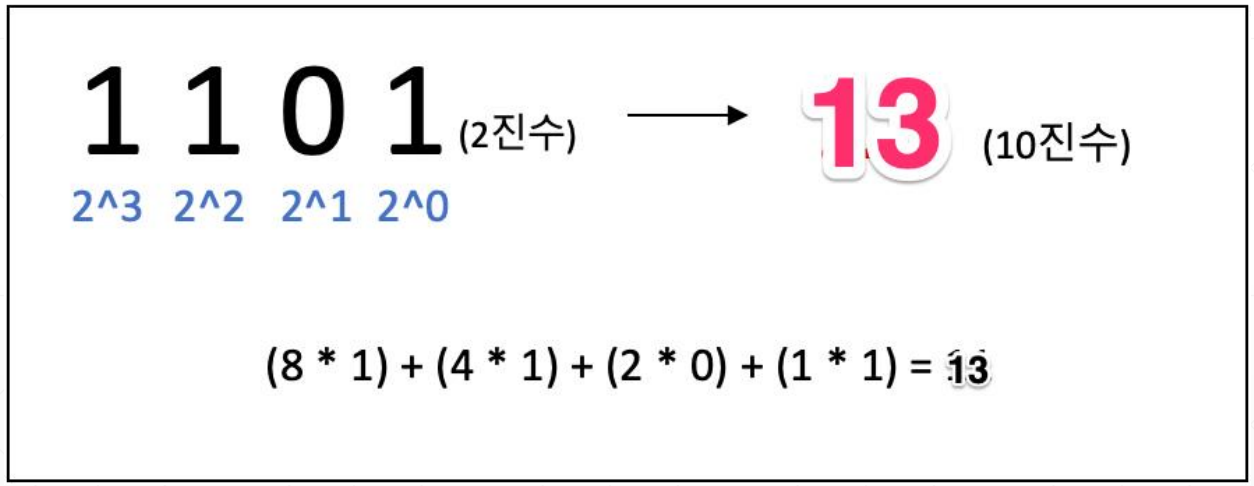
Dynamic Programming

Dynamic Programming

- solve complex problems by dividing them into simple multiple problems
- Conditions
  - A small problem is repeated.
  - The same question has the same answer every time it is asked.
- Memoization
  - Save a small calculated problem and use it again
- Top-Down vs Bottom-Up
  - Top-down: Solve small problems if big problems are not solved
  - Bottom-up: Solving small problems step by step

https://galid1.tistory.com/507

Bit Mask



- Techniques using binary representations of integers
  - { 0, 1, 2, 3, 4 } => 11111
  - { 1, 2, 3, 4 } => 11110
  - { 1, 2, 4 } => 10110
  - { 2, 4 } => 10100
  - { 1 } => 00010
- Express a collection of cities passed by

Code : TSP using dynamic programming

```python
def TSP(distance):
    N = len(distance)
    VISITED_ALL = (1 << N) - 1 #if n이 5일때, 11111(전부 다녀간 경우) 만들어줌
    cache = [[None] * (1 << N) for _ in range(N)]
    INF = float('inf')

    def find_path(last, visited): #last: 중간(현재까지 방문한) 경로의 마지막도시, #visited
        if visited == VISITED_ALL:
            if distance[last][0] ==-1:
                return INF, []
            return [last+1], distance[last][0]

        if cache[last][visited] is not None: #cache에 있으면 그거 재사용
            return cache[last][visited]

        tmp = INF
        tmplist = []

        for city in range(N):
            # print("Come in for statement")
            if visited & (1 << city) == 0 and distance[last][city] != -1: #visited &

                ncity, ntmp = find_path(city, visited | (1 << city))
                ntmp += distance[last][city]
                ntmplist = [last+1] + ncity
                if ntmp < tmp:
                    tmplist = ntmplist
                    tmp = min(tmp, ntmp) #visited | (1 << city): city가 포함된 경로 갱

        cache[last][visited] = tmplist, tmp
        return tmplist, tmp

    return find_path(0, 1 << 0)
```

- Using Bit mask
  - VISITED_ALL = 11111(all visited)
- Cache list to store small question's minimum answer
- Top-down Method
  - 0001 -> 0011 -> 0111 -> 1111(smallest)

Time Complexity : Dynamic Programing

N=10

```
Shortest path(1) or Tsp(2)? Please input the menu number.2
[(1, 2, 3, 10, 8, 4, 6, 5, 7, 9), 57]
8.416694402694702

Please input the TSP size(3~17)10
10의 최단경로 ([1, 2, 3, 10, 8, 4, 6, 5, 7, 9], 57)
0.02278304100036621
```

N=11

```
Shortest path(1) or Tsp(2)? Please input the menu number.2
[(1, 2, 3, 10, 8, 4, 6, 5, 7, 9, 11), 60]
89.18377304077148

Please input the TSP size(3~17)11
11의 최단경로 ([1, 2, 3, 10, 8, 4, 6, 5, 7, 9, 11], 60)
0.042912721633911113
```

- Big difference in time
  - compared to permutations
- 9nn times difference in n=10

Q&A