Python: a faster way to solve non-trivial problems

Yasser Bashir

PyCon Pakistan 2017

whoami

Programmer for 20 years

- C++
- Java
- Python
- Javascript

Builder of successful (and failed) products

Founder of companies

(Also amateur squash player)

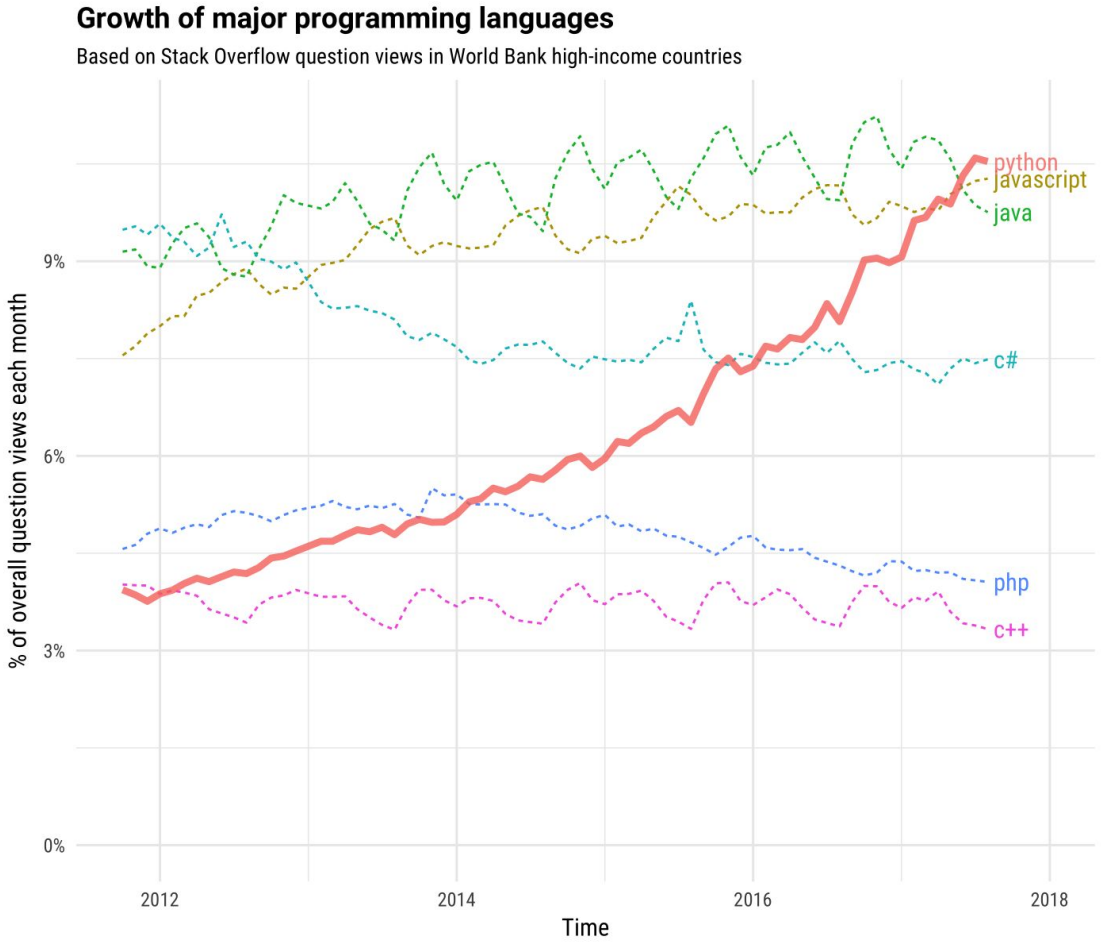
Why Python?

Language of the fourth industrial wave.

Why Python?

Language of the fourth industrial wave.

Because it is very popular and its popularity is increasing.



Why Python?

Language of the fourth industrial wave.

Because it is very popular and its popularity is increasing.

Because it is practised by contributors in both industrial and scientific communities.

Why Python?

Language of the fourth industrial wave.

Because it is very popular and its popularity is increasing.

Because it is practised by contributors in both industrial and scientific communities.

Because it is easy to understand.

Why Python?

Language of the fourth industrial wave.

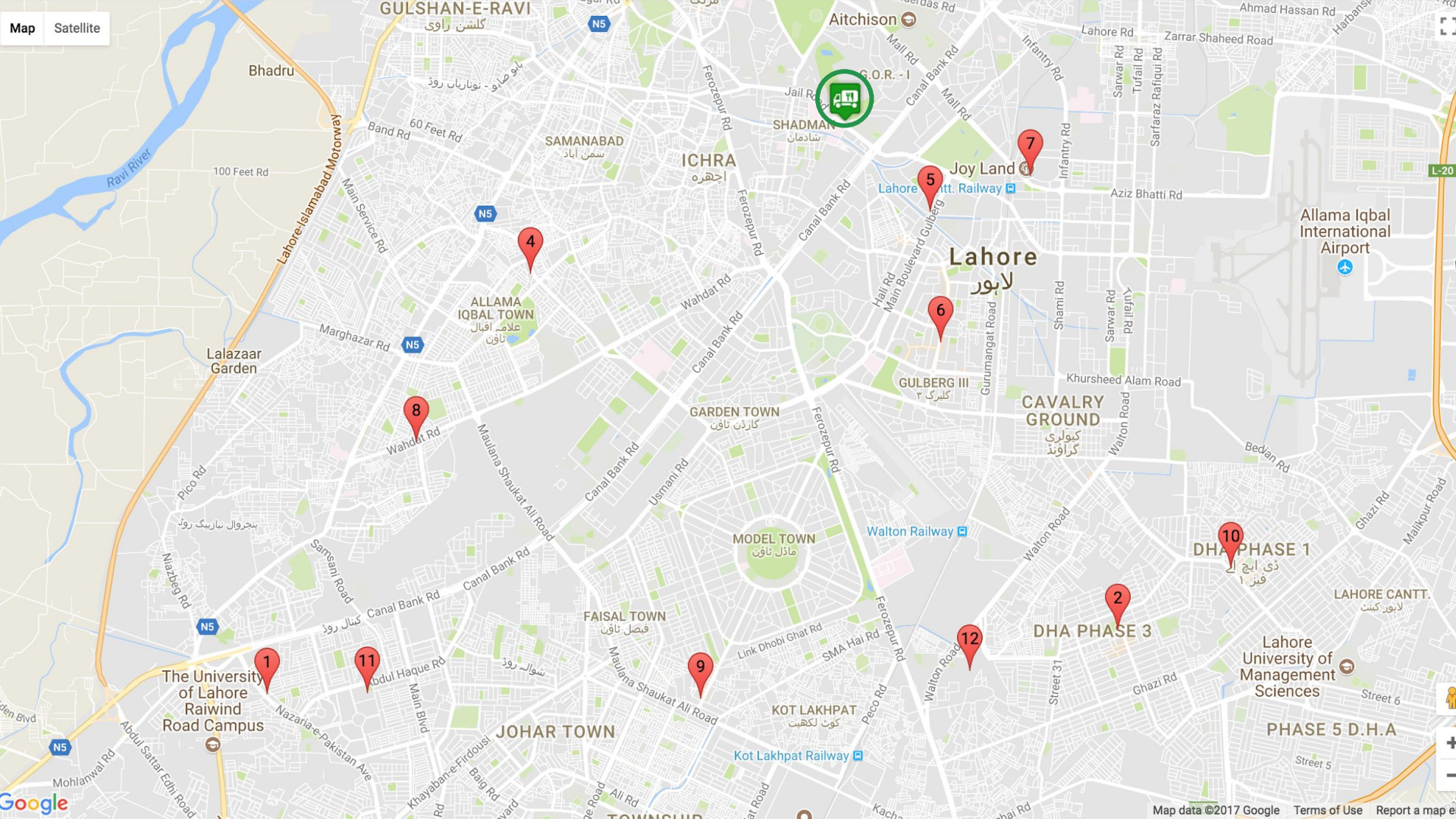
Because it is very popular and its popularity is increasing.

Because it is practised by contributors in both industrial and scientific communities.

Because it is easy to understand.

Because there is very little syntax to learn

Python allows you to solve non-trivial problems faster


Map
Satellite
GULSHAN-E-RAVI
Bhadru
Ravi River
100 Feet Rd
Band Rd
60 Feet Rd
SAMANABAD
ICHRA
SHADMAN
Aitchison
G.O.R. - I
Joy Land
Lahore Cantt. Railway
Lahore-Islamabad Motorway
Main Service Rd
Ferozepur Rd
Canal Bank Rd
Jail Rd
Mall Rd
Lahore Rd
Zarrar Shaheed Road
Ahmad Hassan Rd
Infantry Rd
Sarwar Rd
Tufail Rd
Sarfaraz Rafiqui Rd
Aziz Bhatti Rd
L-20
Allama Iqbal International Airport
Lahore
لاہور
Wahdat Rd
Main Boulevard Gulberg
Hali Rd
Shami Rd
Gurumangat Road
Jinnah Rd
ALLAMA IQBAL TOWN
Marghazar Rd
Lalazaar Garden
GULBERG III
CAVALRY GROUND
Khursheed Alam Road
Walton Road
Bedian Rd
GARDEN TOWN
Maulana Shaukat Ali Road
Usmani Rd
Pico Rd
Wahdat Rd
MODEL TOWN
Walton Railway
DHA PHASE 1
Ghazi Rd
Malikpur Road
Samnani Road
Niazbeg Rd
Canal Bank Rd
FAISAL TOWN
Link Dhobi Ghat Rd
SMA Hai Rd
DHA PHASE 3
LAHORE CANTT.
The University of Lahore Raiwind Road Campus
Abdul Haque Rd
Main Blvd
Nazaria-e-Pakistan Ave
JOHAR TOWN
KOT LAKHPAT
Peco Rd
Kot Lakhpat Railway
Lahore University of Management Sciences
Street 31
Street 6
Street 5
PHASE 5 D.H.A
Mohlanwal Rd
Abdul Sattar Edhi Road
Khayaban-e-Firdousi
Baig Rd
Ali Rd
N5
1
2
4
5
6
7
8
9
10
11
12
Google
Map data ©2017 Google
Terms of Use
Report a map e

The Vehicle Routing Problem (VRP)

How to minimize the time (or distance) required to visit a set of locations given that:

- Vehicles have unlimited capacity
- There may be multiple vehicles
- Vehicles depart from one depot
- Visit each location once only
- Come back to the depot

Input

Locations: [0,1,2,3,4,5,6,7,8,9]

Vehicles: 3 *Distances:* [0→1=4, 0→2=5, 2→3=7 etc]

....

Assume 0th location is the depot

Find all partitions of size k for the input set where k is the number of vehicles

Minimize the route for each subset in each partition

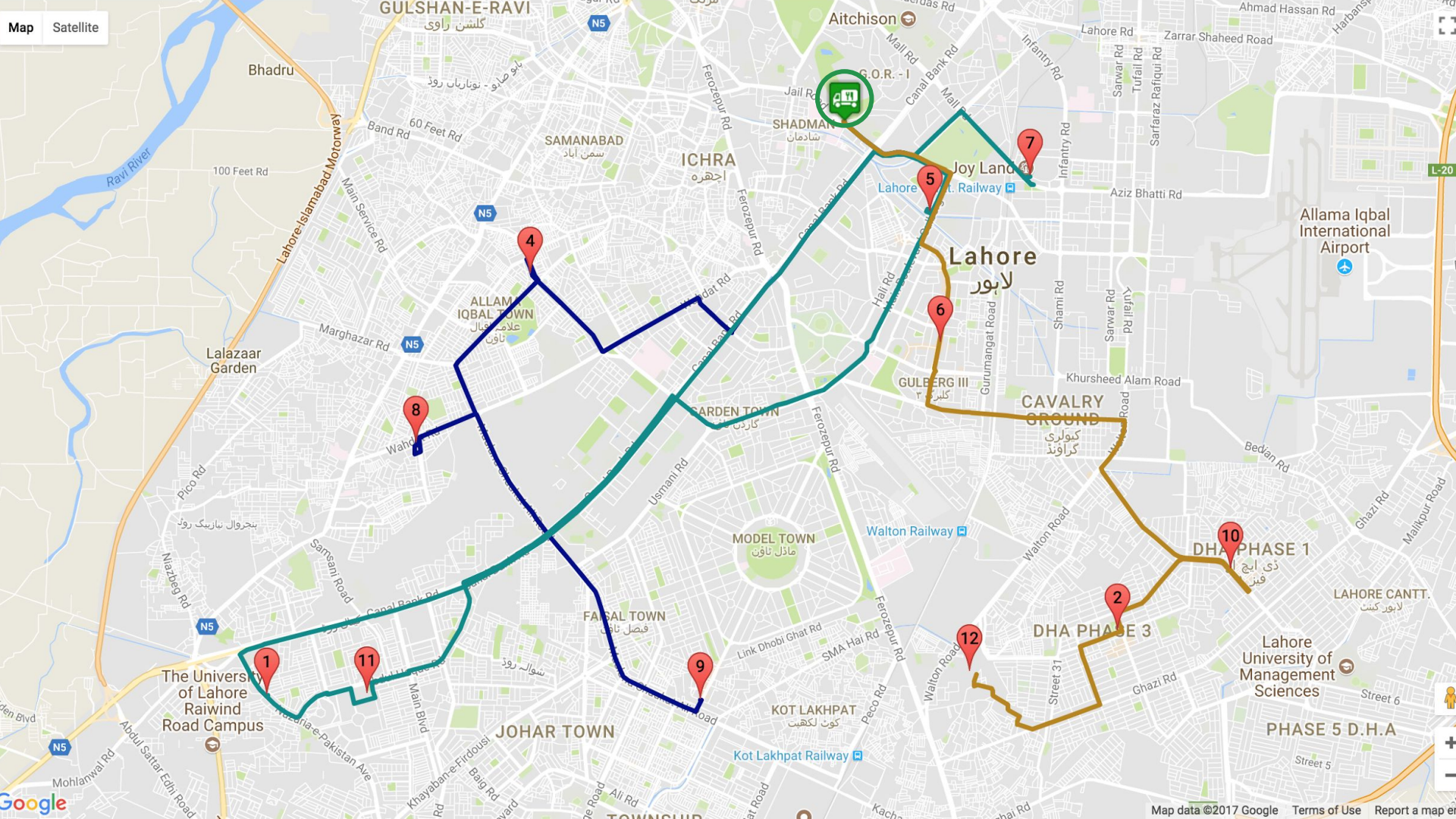
Find the partition with the smallest cost

....

Output

Optimal route: [[0,7,2] , [0,3,5,9], [0,4,1,8,6]]

Optimal time: 72.3 minutes


Map
Satellite
GULSHAN-E-RAVI
Aitchison
Bhadru
Ravi River
SHADMAN
G.O.R. - I
SAMANABAD
ICHRA
Joy Land
Lahore
Allama Iqbal International Airport
ALLAMA IQBAL TOWN
Lalazaar Garden
GULBERG III
CAVALRY GROUND
GARDEN TOWN
MODEL TOWN
Walton Railway
DHA PHASE 1
DHA PHASE 3
LAHORE CANTT.
FAISAL TOWN
JOHAR TOWN
KOT LAKHPAT
Kot Lakhpat Railway
The University of Lahore Raiwind Road Campus
Lahore University of Management Sciences
PHASE 5 D.H.A
Lahore-Islamabad Motorway
N5
L-20
1
2
4
5
6
7
8
9
10
11
12
Map data ©2017 Google
Terms of Use
Report a map er

Inputs

Id	Address	Lat	Lon
0	Al-Hussain Road، Lahore, Pakistan	31.4669244	74.25186409999999
1	Lahore, Pakistan	31.3743935	74.17818310000001
2	Sheeba Park, Y Block Market, DHA Phase 3، Lahore, Pakistan	31.47498589999999	74.3770639
3	13 Main Boulevard Allama Iqbal Town، Lahore 54000, Pakistan	31.5197581	74.29066159999999
4	89-B, Jail Rd، Lahore, Pakistan	31.5386987	74.3369092
5	M.M. Alam Road, Block B2، Lahore 54660, Pakistan	31.5111079	74.3509436
6	63 Main Gulberg، Lahore, Pakistan	31.5274972	74.34939969999999
7	Plot #14 and Plot # 15, Hadayatullah Block, Mustafa Town, Wahdat Road، Lahore, Pakistan	31.4984994	74.2737308
8	23-Phase II, Govt. Employees Cooperative Housing Society Ltd, Model Town Link Road، Lahore 54600, Pakistan	31.4662661	74.31567249999999

start	end	distance	duration
0	1	17.79	32.80
0	2	17.29	34.83
0	3	9.76	22.00
0	4	13.69	23.82

The Pythonic Solution

6

functions

26

lines of (non-spaghetti) code using core libraries

50

seconds running time

3

hours coding time

https://github.com/ybashir/vrpfun

```
def distance(x, y):
  return distances[x][y]

def route_length(route):
  return sum(distance(route[i], route[i - 1]) for i, v in enumerate(route))

def all_routes(seq):
  return [[seq[0]] + list(rest) for rest in itertools.permutations(seq[1:])]

def all_partitions(collection):
  if len(collection) == 1:
    yield [collection]
    return
  first = collection[0]
  for smaller in all_partitions(collection[1:]):
    for n, subset in enumerate(smaller):
      yield smaller[:n] + [[first] + subset] + smaller[n + 1:]
    yield [[first]] + smaller

def k_partitions_with_shortest_routes(ids, k=3):
  for p in filter(lambda x: len(x) == k, all_partitions(ids[1:])):
    yield [min(all_routes([ids[0]] + q), key=route_length) for q in p]

def shortest_partition(ids, k=3):
  return min(k_partitions_with_shortest_routes(ids, k),
             key=lambda x: max(route_length(x[i]) for i in range(k)))
```

Python

```python
def route_length(route):
  return sum(distance(route[i], route[i - 1]) for i, v in enumerate(route))
```

```
>>> route_length([0,3,5,2])
134.85
```

Java

```java
private float routeLength(List<Integer> route) {
    float sum = 0.0f;
    for (int i = 1; i < route.size(); i++) {
        sum += getDistance(route.get(i), route.get(i - 1));
    }
    sum += getDistance(route.get(0), route.get(route.size() - 1));
    return sum;
}
```

Kotlin

```kotlin
fun routeLength(route: List<Int>): Double {
    var sum = 0.0
    for (i in 1 until route.size)
        sum += distance(route[i], route[i - 1])
    sum += distance(route[0], route[route.size - 1])
    return sum
}
```

Python

```python
def all_routes(seq):
    return [[seq[0]] + list(rest) for rest in itertools.permutations(seq[1:])]
```

```
>>> all_routes([0,3,5,2])
[[0, 3, 5, 2], [0, 3, 2, 5], [0, 5, 3, 2], [0, 5, 2, 3], [0, 2, 3, 5], [0, 2, 5, 3]]
```

Java

```java
private List<List<Integer>> allRoutes(List<Integer> seq) {
    Integer depot = seq.get(0);
    List<List<Integer>> allPossibleRoutes = new ArrayList<>();
    Collection<List<Integer>> permutations = Collections2.permutations(seq.subList(1, seq.size()));
    for (List<Integer> permutation : permutations) {
        List<Integer> route = new ArrayList<>();
        route.add(depot);
        route.addAll(permutation);
        allPossibleRoutes.add(route);
    }
    return allPossibleRoutes;
}
```

Kotlin

```kotlin
fun allRoutes(original: ArrayList<Int>): ArrayList<ArrayList<Int>> {
    if (original.size < 2) {
        return arrayListOf(original)
    } else {
        val firstElement = original.removeAt(0)
        return permutations(original).map {
            it.add(0, firstElement)
            return@map it
        } as ArrayList<ArrayList<Int>>
    }
}
```

```python
def all_partitions(collection):
  if len(collection) == 1:
    yield [collection]
    return
  first = collection[0]
  for smaller in all_partitions(collection[1:]):
    for n, subset in enumerate(smaller):
      yield smaller[:n] + [[first] + subset] + smaller[n + 1:]
    yield [[first]] + smaller
```

```
>>> for p in (all_partitions([0,3,5,2])):
...     print(p)

[[0, 3, 5, 2]]
[[0], [3, 5, 2]]
[[0, 3], [5, 2]]
[[3], [0, 5, 2]]
[[0], [3], [5, 2]]
[[0, 3, 5], [2]]
[[3, 5], [0, 2]]
[[0], [3, 5], [2]]
[[0, 5], [3, 2]]
[[5], [0, 3, 2]]
[[0], [5], [3, 2]]
[[0, 3], [5], [2]]
[[3], [0, 5], [2]]
[[3], [5], [0, 2]]
[[0], [3], [5], [2]]
```

https://stackoverflow.com/questions/19368375/set-partitions-in-python

```java
public static List<List<List<Integer>>> getAllPartitions(List<Integer> inputList) throws Exception {
    int[] array = inputList.stream().mapToInt(i -> i).toArray();
    int[][][] partitionArrays = getAllPartitions(array);
    List<List<List<Integer>>> partitionsList = new ArrayList<>();
    for (int[][] partition : partitionArrays) {
        List<List<Integer>> partitionSets = new ArrayList<>();
        for (int[] set : partition) {
            List<Integer> singleSet = Ints.asList(set);
            partitionSets.add(singleSet);
        }
        partitionsList.add(partitionSets);
    }
    return partitionsList;
}
private static int[][][] getAllPartitions(int[] array) throws Exception {
    int[][][] res = new int[0][][];
    int n = 1;
    for (int i = 0; i < array.length; i++) {
        n *= 2;
    }
    for (int i = 1; i < n; i += 2) {
        boolean[] contains = new boolean[array.length];
        int length = 0;
        int k = i;
        for (int j = 0; j < array.length; j++) {
            contains[j] = k % 2 == 1;
            length += k % 2;
            k /= 2;
        }
        int[] firstPart = new int[length];
        int[] secondPart = new int[array.length - length];
        int p = 0;
        int q = 0;
        for (int j = 0; j < array.length; j++) {
            if (contains[j]) {
                firstPart[p++] = array[j];
            } else {
                secondPart[q++] = array[j];
            }
        }
        int[][][] partitions;
        if (length == array.length) {
            partitions = new int[][][] {{firstPart}};
        } else {
            partitions = getAllPartitions(secondPart);
            for (int j = 0; j < partitions.length; j++) {
                int[][] partition = new int[partitions[j].length + 1][];
                partition[0] = firstPart;
                System.arraycopy(partitions[j], 0, partition, 1, partitions[j].length);
                partitions[j] = partition;
            }
        }
        int[][][] newRes = new int[res.length + partitions.length][][];
        System.arraycopy(res, 0, newRes, 0, res.length);
        System.arraycopy(partitions, 0, newRes, res.length, partitions.length);
        res = newRes;
    }
    return res;
}
```

```
20 def k_partitions_with_shortest_routes(ids, k=3):
21   for p in filter(lambda x: len(x) == k, all_partitions(ids[1:])):
22     yield [min(all_routes([ids[0]] + q), key=route_length) for q in p]
```

```
>>> for p in k_partitions_with_shortest_routes([0,3,5,2],k=2):
...     print(p)
...
[[0, 3], [0, 5, 2]]
[[0, 3, 5], [0, 2]]
[[0, 5], [0, 3, 2]]
```

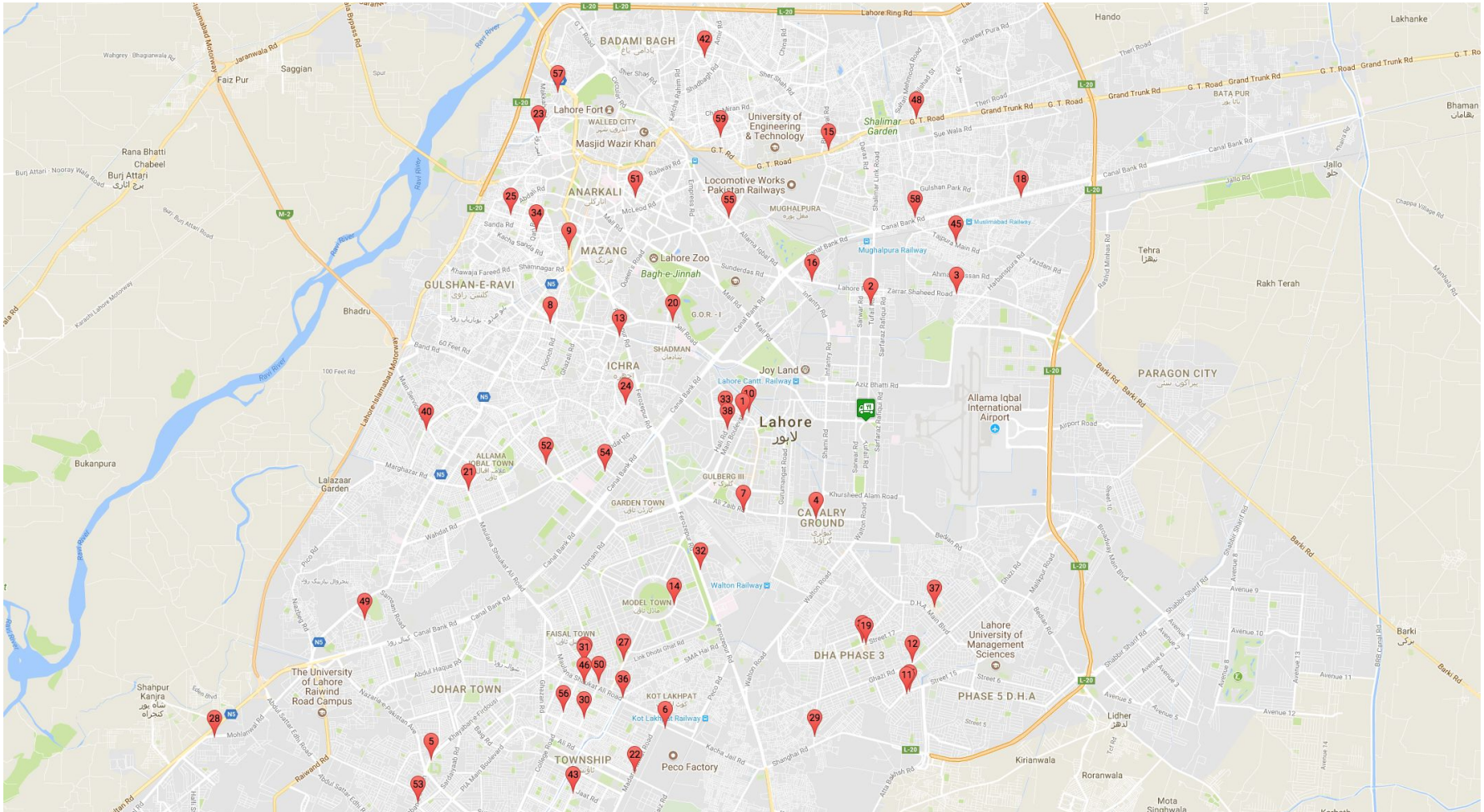
```
24 def shortest_partition(ids, k=3):
25   return min(k_partitions_with_shortest_routes(ids, k),
26              key=lambda x: max(route_length(x[i]) for i in range(k)))
```

```
>>> shortest_partition([0,3,5,2],k=2)
[[0, 3], [0, 5, 2]]
```

```
def distance(x, y):
  return distances[x][y]

def route_length(route):
  return sum(distance(route[i], route[i - 1]) for i, v in enumerate(route))

def all_routes(seq):
  return [[seq[0]] + list(rest) for rest in itertools.permutations(seq[1:])]

def all_partitions(collection):
  if len(collection) == 1:
    yield [collection]
    return
  first = collection[0]
  for smaller in all_partitions(collection[1:]):
    for n, subset in enumerate(smaller):
      yield smaller[:n] + [[first] + subset] + smaller[n + 1:]
    yield [[first]] + smaller

def k_partitions_with_shortest_routes(ids, k=3):
  for p in filter(lambda x: len(x) == k, all_partitions(ids[1:])):
    yield [min(all_routes([ids[0]] + q), key=route_length) for q in p]

def shortest_partition(ids, k=3):
  return min(k_partitions_with_shortest_routes(ids, k),
             key=lambda x: max(route_length(x[i]) for i in range(k)))
```

	Python (PyPy)	Java	Kotlin
Execution time	50 secs	25 secs	19 secs
Coding time	3 hours	12 hours	8 hours
Lines of code	30	150+	100
Memory footprint	136M	346M	887 MB

```
>>> draw_map(get_locations('gourmets in lahore'),[],'gourmet.html')
```



Thank you

Special thanks to:

Aitzaz Mumtaz Khan (for contributing java code)

https://github.com/aitzaz/vehicle_routing_problem

Shahroz Khan (for contributing Kotlin code)

https://github.com/Shahroz16/VRP