

# 1 Problem Definition

Implement a nearest neighbor search technique on grid-indexed data. The algorithm should take as input: a query location  $q$  and an integer  $k$  and finds the  $k$ -nearest neighbors of  $q$  using the grid index.

## 2 Design and Implementation

### 2.1 Grid construction

Each data record is stored in a data structure named `Record`. The grids are stored in a multi-array for fast accesses. The grid construction algorithm is showed in Listing 2.1. The construction needs two scans of the dataset. In the first scan, it finds the maximum latitude and longitude of all location, determines the grid size and create the grid storage. Then in the second scan, put the record to its corresponding grid.

---

```
1  # visited
2  # min_lat, min_long, max_lat, max_long are got in the first scan
3  grid_size_lat = (max_lat - min_lat) / n
4  grid_size_long = (max_long - min_long) / n
5  grid_size = max(grid_size_lat, grid_size_long)
6
7  for r in records:
8      if not r in visited:
9          x = (r.lat - min_lat) / grid_size
10         y = (r.long - min_long) / grid_size
11         grids(x)(y) += r
```

---

Listing 1: grid construction.

### 2.2 k-nearest neighbour search on grids

A  $k$ -nearest neighbour search query accepts a parameter of location (latitude and longitude). It first finds the corresponding grid of the location, and then find a  $k$ -nearest locations in the grid. After that, it searches the layered cells. The algorithm is showed in Figure 2.2.

---

```
1  def grid_search(grid_records, lat, long, k_set):
2      for (r <- grid_record):
3          if r < k_set.max or len(k_set) < k
4              k_set.insert(r)
5      rerurn k_set
```

---

Listing 2: Single grid search.

---

```

1 def query(lat, long, k):
2     x = (lat - min_lat) / grid_size
3     y = (lat - min_long) / grid_size
4     k_set = grid_search(grids(x)(y), k_set)
5     while (true):
6         for cell in cells layered around x, y:
7             d = dis(cell, lat, long)
8             min_d = min(d, min_d)
9             if d < k_set.min or len(k_set) < k:
10                 k_set = grid_search(cell, k_set)
11             if (len(k_set) >= k and k_set.max < d_min):
12                 break
13     return k_set

```

---

Listing 3: knn search algorithm.

In this algorithm, if the minimum distance between the cell and the location is larger than the maximum value in `k_set`, then the cell does not need to be access. This algorithm stop when the minimum distance between the location and all cells in the layer (none of the cells in the layer has been accessed).

## 2.3 benchmark

Evaluation of the algorithm is focused of the time of constructing the grids and the average query time. Random locations were generated randomly and these locations are served as queries to the knn search system. The average query time for 10k queries is recorded. A simple linear scan algorithm is used for comparison. The result is showed in Table 1.

name	construction time	query time	cell accessed
grid search	10s	43ms	7800
linear scan	N/A	43ms	10000

Table 1: Evaluation result.