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 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.

```
# visited
# min_lat, min_long, max_lat, max_long are got in the first scan
foreach (r <- records)
  if not r in visited
    x = (r.lat - min_lat) / grid_size
    y = (r.long - min_long) / grid_size
    grids(x)(y) += r
```

Listing 1: Grid construction algorithm.

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. A function called `grid_d` (see Figure 2) is to search inside a grid. `k_set` is list with k records ordered by its distance with the location. The full algorithm is showed in Figure 4.

```
def grid_d(grid_records, lat, long, k_set):
  for (r <- grid_record):
    if r < k_set.max or len(k_set) < k
      k_set.insert(r)
  rerurn k_set
```

Listing 2: Function grid_d.

```
def query(lat, long, k):
  x = (lat - min_lat) / grid_size
  y = (lat - min_long) / grid_size
  k_set = grid_search(grids(x)(y), k_set)
  while (true):
    for cell in cells layered around x, y:
      d = dis(cell, lat, long)
      min_d = min(d, min_d)
      if d < k_set.min or len(k_set) < k:
        k_set = grid_search(cell, k_set)
    if (len(k_set) >= k and k_set.max < d_min):
      break
  return k_set
```

Listing 3: K-Nearest Neighbours 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 stops 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. 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 search100 x 100)	22s	3ms	1477
grid search100 x 50)	22s	6ms	799
grid search50 x 100)	22s	1ms	380
linear scan	N/A	32ms	N/A

Table 1: Evaluation result.

Table 1 shows that linear scan has a much larger access time compared with grid search. This is because grid search can avoid unnecessary search. 50 x 100 grid has a better performance as it better suits the range of the train data.