

## [공간데이터 관리 및 응용 프로젝트 2]

20161575 김연요

### 1. 프로젝트 파일 분류:

README 파일에도 기술해놓았지만, 폴더를 압축 해제하시면 20161575 Code 폴더 안에 해당 보고서 외에도 3 가지 dataset 들과 BruteForce, KDTree, RTree 폴더 세 개가 존재합니다. 각 폴더에서 각 해당하는 방법의 Query 를 실행해보실 수 있습니다. 컴파일 방법과 실행 방법 역시 각 폴더안의 README 파일을 확인하시면 좋을 것 같습니다.

### 2. 각 질의 처리 구현 함수와 자료구조

세 개의 방법 모두에서 main 함수에서는 Query 를 입력받고, dataset 종류를 입력 받고, 각 각 선택한 종류의 Query 에 따라 해당 query 들을 실행합니다.

#### 1) BruteForce.c

브루트포스의 경우, 이름과 마찬가지로 전체 탐색을 통해 질의 조건에 맞는 객체를 탐색합니다.

사용한 자료구조의 경우 2 차원 좌표값을 저장할 수 있는 **point** 구조체와 knn query 에서 배열을 top-k 개 저장할 수 있는 구조체 **knn** 또한 정의했습니다. 그 외에도 두 개의 point 구조체들 사이의 유클리디언 distance 를 return 하는 함수인 **euclid\_dist** 또한 정의했습니다.

#### 2) KDTree.c

KDTree 에서는 기본적으로 제공된 구조체와 함수들을 적극 활용했습니다.

그 외에도, knn query 를 할 때 사용한 **knn\_node** 구조체와 resultQ 라는 priority queue 를 직접 구현해서 그 기능들을 활용하여 kNN 의 후보가 되는 오브젝트 값들을 정렬해서 저장했습니다. priority queue 구조체를 선언함과 동시에 해당 priority queue 가 사용하는 함수들 (push, pop, empty 등) 을 직접 구현했습니다.

각 코드를 구현한 방식에 대해서는 주석을 통해 상세히 적어놨습니다.

### 3. 실험 결과 분석

#### 1) Bruteforce

##### A) Range Query

```
ginyeon-yo ~/Desktop/yoProjects/SpatialData/Project 2/Codes/BruteForce > main
~/BruteForce
BruteForce Query!!!
Select Query Type
1. Range Query
2. KNN Query
1
Select Dataset
1. clustered_dataset.txt
2. gaussian_dataset.txt
3. uniform_dataset.txt
1
[Range Query]
Test Query Point (x,y) : (250.000000, 250.000000)
Range query for range : 10.00
Query Object Count: 0
Query Time : 14737.000000

Range query for range : 20.00
Query Object Count: 0
Query Time : 7485.000000

Range query for range : 40.00
Query Object Count: 177
Query Time : 7254.000000

Range query for range : 60.00
Query Object Count: 18933
Query Time : 8897.000000

Range query for range : 80.00
Query Object Count: 19917
Query Time : 8632.000000

Range query for range : 100.00
Query Object Count: 20000
Query Time : 7541.000000

ginyeon-yo ~/Desktop/yoProjects/SpatialData/Project 2/Codes/BruteForce > main
~/BruteForce
BruteForce Query!!!
Select Query Type
1. Range Query
2. KNN Query
1
Select Dataset
1. clustered_dataset.txt
2. gaussian_dataset.txt
3. uniform_dataset.txt
2
[Range Query]
Test Query Point (x,y) : (250.000000, 250.000000)
Range query for range : 10.00
Query Object Count: 88215
Query Time : 17580.000000

Range query for range : 20.00
Query Object Count: 33896
Query Time : 12097.000000

Range query for range : 40.00
Query Object Count: 88815
Query Time : 8142.000000

Range query for range : 60.00
Query Object Count: 97362
Query Time : 10290.000000

Range query for range : 80.00
Query Object Count: 99841
Query Time : 6000.000000

Range query for range : 100.00
Query Object Count: 99990
Query Time : 8690.000000

ginyeon-yo ~/Desktop/yoProjects/SpatialData/Project 2/Codes/BruteForce > main
~/BruteForce
BruteForce Query!!!
Select Query Type
1. Range Query
2. KNN Query
2
Select Dataset
1. clustered_dataset.txt
2. gaussian_dataset.txt
3. uniform_dataset.txt
1
[KNN Query]
Test Query Point (x,y) : (250.000000, 250.000000)
KNN query for K : 1
Query Object Count: 1
Query Time : 17200.000000

KNN query for K : 10
Query Object Count: 10
Query Time : 9964.000000

KNN query for K : 20
Query Object Count: 20
Query Time : 8684.000000

KNN query for K : 30
Query Object Count: 30
Query Time : 8072.000000

KNN query for K : 40
Query Object Count: 40
Query Time : 8204.000000

KNN query for K : 50
Query Object Count: 50
Query Time : 8144.000000

KNN query for K : 60
Query Object Count: 60
Query Time : 8150.000000

KNN query for K : 70
Query Object Count: 70
Query Time : 8444.000000

KNN query for K : 80
Query Object Count: 80
Query Time : 8420.000000

KNN query for K : 90
Query Object Count: 90
Query Time : 8745.000000

KNN query for K : 100
Query Object Count: 100
Query Time : 10840.000000
```

Brute Force 를 각각 Clustered, Gaussian, Uniform 한 dataset 에 대해 Range query 를 질의한 결과입니다.

##### B) KNN Query

```
ginyeon-yo ~/Desktop/yoProjects/SpatialData/Project 2/Codes/BruteForce > main
~/BruteForce
BruteForce Query!!!
Select Query Type
1. Range Query
2. KNN Query
2
Select Dataset
1. clustered_dataset.txt
2. gaussian_dataset.txt
3. uniform_dataset.txt
1
[KNN Query]
Test Query Point (x,y) : (250.000000, 250.000000)
KNN query for K : 1
Query Object Count: 1
Query Time : 17200.000000

KNN query for K : 10
Query Object Count: 10
Query Time : 9964.000000

KNN query for K : 20
Query Object Count: 20
Query Time : 8684.000000

KNN query for K : 30
Query Object Count: 30
Query Time : 8072.000000

KNN query for K : 40
Query Object Count: 40
Query Time : 8204.000000

KNN query for K : 50
Query Object Count: 50
Query Time : 8144.000000

KNN query for K : 60
Query Object Count: 60
Query Time : 8150.000000

KNN query for K : 70
Query Object Count: 70
Query Time : 8444.000000

KNN query for K : 80
Query Object Count: 80
Query Time : 8420.000000

KNN query for K : 90
Query Object Count: 90
Query Time : 8745.000000

KNN query for K : 100
Query Object Count: 100
Query Time : 10840.000000

ginyeon-yo ~/Desktop/yoProjects/SpatialData/Project 2/Codes/BruteForce > main
~/BruteForce
BruteForce Query!!!
Select Query Type
1. Range Query
2. KNN Query
2
Select Dataset
1. clustered_dataset.txt
2. gaussian_dataset.txt
3. uniform_dataset.txt
1
[KNN Query]
Test Query Point (x,y) : (250.000000, 250.000000)
KNN query for K : 1
Query Object Count: 1
Query Time : 17200.000000

KNN query for K : 10
Query Object Count: 10
Query Time : 9964.000000

KNN query for K : 20
Query Object Count: 20
Query Time : 8684.000000

KNN query for K : 30
Query Object Count: 30
Query Time : 8072.000000

KNN query for K : 40
Query Object Count: 40
Query Time : 8204.000000

KNN query for K : 50
Query Object Count: 50
Query Time : 8144.000000

KNN query for K : 60
Query Object Count: 60
Query Time : 8150.000000

KNN query for K : 70
Query Object Count: 70
Query Time : 8444.000000

KNN query for K : 80
Query Object Count: 80
Query Time : 8420.000000

KNN query for K : 90
Query Object Count: 90
Query Time : 8745.000000

KNN query for K : 100
Query Object Count: 100
Query Time : 10840.000000
```

Brute Force 를 각각 Clustered, Gaussian, Uniform 한 dataset 에 대해 KNN query 를 질의한 결과입니다.

**결과 분석** : Bruteforce 의 경우 모든 data 들에 대해 전체 탐색을 진행하기 때문에 Query 종류와 관계 없이 시간이 오래걸리는 비효율성을 보여준다. 그렇기 때문에 당연히 Range 값이나 K 값에도 시간 영향은 받지 않는 모습을 확인할 수 있었다.

이후에 KD tree 와의 비교를 통해 시간 효율성을 더 깊이 비교, 확인해보겠다.

## 2) KD Tree

### A) Range Query

```
glsysen-yo ~/Desktop/Projects/SpatialData/Project 2/Codes/KDTree 7 main 3
./KDTree
KDTree Query!!!
Select Query Type
1. Range Query
2. KNN Query
3.
Select Dataset
1. clustered_dataset.txt
2. gaussian_dataset.txt
3. uniform_dataset.txt
4.
[Range Query]
Test Query Point (x,y) : (250.000000, 250.000000)
Range query for range : 10.00
Query Object Count: 0
Query Time : 26.0000 ms

Range query for range : 20.00
Query Object Count: 0
Query Time : 9.0000 ms

Range query for range : 40.00
Query Object Count: 127
Query Time : 101.0000 ms

Range query for range : 60.00
Query Object Count: 1893
Query Time : 710.0000 ms

Range query for range : 80.00
Query Object Count: 19917
Query Time : 716.0000 ms

Range query for range : 100.00
Query Object Count: 20000
Query Time : 700.0000 ms

glsysen-yo ~/Desktop/Projects/SpatialData/Project 2/Codes/KDTree 7 main 3
./KDTree
KDTree Query!!!
Select Query Type
1. Range Query
2. KNN Query
3.
Select Dataset
1. clustered_dataset.txt
2. gaussian_dataset.txt
3. uniform_dataset.txt
4.
[Range Query]
Test Query Point (x,y) : (250.000000, 250.000000)
Range query for range : 10.00
Query Object Count: 0
Query Time : 533.0000 ms

Range query for range : 20.00
Query Object Count: 33000
Query Time : 1639.0000 ms

Range query for range : 40.00
Query Object Count: 80010
Query Time : 3210.0000 ms

Range query for range : 60.00
Query Object Count: 97000
Query Time : 3613.0000 ms

Range query for range : 80.00
Query Object Count: 99900
Query Time : 3970.0000 ms

Range query for range : 100.00
Query Object Count: 100000
Query Time : 3814.0000 ms

glsysen-yo ~/Desktop/Projects/SpatialData/Project 2/Codes/KDTree 7 main 3
./KDTree
KDTree Query!!!
Select Query Type
1. Range Query
2. KNN Query
3.
Select Dataset
1. clustered_dataset.txt
2. gaussian_dataset.txt
3. uniform_dataset.txt
4.
[Range Query]
Test Query Point (x,y) : (250.000000, 250.000000)
Range query for range : 10.00
Query Object Count: 0
Query Time : 45.0000 ms

Range query for range : 20.00
Query Object Count: 500
Query Time : 63.0000 ms

Range query for range : 40.00
Query Object Count: 2000
Query Time : 180.0000 ms

Range query for range : 60.00
Query Object Count: 4000
Query Time : 279.0000 ms

Range query for range : 80.00
Query Object Count: 6000
Query Time : 453.0000 ms

Range query for range : 100.00
Query Object Count: 7000
Query Time : 574.0000 ms
```

KD Tree 를 사용하여 각각 Clustered, Gaussian, Uniform 한 dataset 에 대해 Range query 를 질의한 결과입니다.

### B) KNN Query

KD Tree 를 사용하여 Clustered, Gaussian, Uniform 한 dataset 에 대해 KNN query 를 질의한 결과입니다.

KNN query 의 경우, K 값을 {1,10,20,30,...100}뿐만 아니라 range 값 또한 radius = 10, 50, 100 으로 변화를 주면서 모두 비교를 해보았습니다.

#### ㄱ. R = 10, (Clustered, Gaussian, Uniform 순서)

```
glsysen-yo ~/Desktop/Projects/SpatialData/Project 2/Codes/KDTree 7 main 3
./KDTree
KDTree Query!!!
Select Query Type
1. Range Query
2. KNN Query
3.
Select Dataset
1. clustered_dataset.txt
2. gaussian_dataset.txt
3. uniform_dataset.txt
4.
[KNN Query]
Test Query Point (x,y) : (250.000000, 250.000000)
KNN query for range 10.0, K : 1
Query Time : 4.0000 ms

KNN query for range: 10.0, K : 10
Query Time : 2.0000 ms

KNN query for range: 10.0, K : 20
Query Time : 6.0000 ms

KNN query for range: 10.0, K : 30
Query Time : 2.0000 ms

KNN query for range: 10.0, K : 40
Query Time : 2.0000 ms

KNN query for range: 10.0, K : 50
Query Time : 8.0000 ms

KNN query for range: 10.0, K : 60
Query Time : 8.0000 ms

KNN query for range: 10.0, K : 70
Query Time : 1.0000 ms

KNN query for range: 10.0, K : 80
Query Time : 0.0000 ms

KNN query for range: 10.0, K : 90
Query Time : 2.0000 ms

KNN query for range: 10.0, K : 100
Query Time : 2.0000 ms

glsysen-yo ~/Desktop/Projects/SpatialData/Project 2/Codes/KDTree 7 main 3
./KDTree
KDTree Query!!!
Select Query Type
1. Range Query
2. KNN Query
3.
Select Dataset
1. clustered_dataset.txt
2. gaussian_dataset.txt
3. uniform_dataset.txt
4.
[KNN Query]
Test Query Point (x,y) : (250.000000, 250.000000)
KNN query for range 10.0, K : 1
Query Time : 362.0000 ms

KNN query for range: 10.0, K : 10
Query Time : 222.0000 ms

KNN query for range: 10.0, K : 20
Query Time : 262.0000 ms

KNN query for range: 10.0, K : 30
Query Time : 350.0000 ms

KNN query for range: 10.0, K : 40
Query Time : 350.0000 ms

KNN query for range: 10.0, K : 50
Query Time : 320.0000 ms

KNN query for range: 10.0, K : 60
Query Time : 320.0000 ms

KNN query for range: 10.0, K : 70
Query Time : 361.0000 ms

KNN query for range: 10.0, K : 80
Query Time : 361.0000 ms

KNN query for range: 10.0, K : 90
Query Time : 225.0000 ms

KNN query for range: 10.0, K : 100
Query Time : 224.0000 ms

glsysen-yo ~/Desktop/Projects/SpatialData/Project 2/Codes/KDTree 7 main 3
./KDTree
KDTree Query!!!
Select Query Type
1. Range Query
2. KNN Query
3.
Select Dataset
1. clustered_dataset.txt
2. gaussian_dataset.txt
3. uniform_dataset.txt
4.
[KNN Query]
Test Query Point (x,y) : (250.000000, 250.000000)
KNN query for range 10.0, K : 1
Query Time : 14.0000 ms

KNN query for range: 10.0, K : 10
Query Time : 7.0000 ms

KNN query for range: 10.0, K : 20
Query Time : 10.0000 ms

KNN query for range: 10.0, K : 30
Query Time : 9.0000 ms

KNN query for range: 10.0, K : 40
Query Time : 9.0000 ms

KNN query for range: 10.0, K : 50
Query Time : 9.0000 ms

KNN query for range: 10.0, K : 60
Query Time : 7.0000 ms

KNN query for range: 10.0, K : 70
Query Time : 8.0000 ms

KNN query for range: 10.0, K : 80
Query Time : 11.0000 ms

KNN query for range: 10.0, K : 90
Query Time : 7.0000 ms

KNN query for range: 10.0, K : 100
Query Time : 9.0000 ms
```

#### ㄴ. R = 50, (Clustered, Gaussian, Uniform 순서)

```
glsysen-yo ~/Desktop/Projects/SpatialData/Project 2/Codes/KDTree 7 main 3
./KDTree
KDTree Query!!!
Select Query Type
1. Range Query
2. KNN Query
3.
Select Dataset
1. clustered_dataset.txt
2. gaussian_dataset.txt
3. uniform_dataset.txt
4.
[KNN Query]
Test Query Point (x,y) : (250.000000, 250.000000)
KNN query for range 50.0, K : 1
Query Time : 206.0000 ms

KNN query for range: 50.0, K : 10
Query Time : 230.0000 ms

KNN query for range: 50.0, K : 20
Query Time : 220.0000 ms

KNN query for range: 50.0, K : 30
Query Time : 222.0000 ms

KNN query for range: 50.0, K : 40
Query Time : 220.0000 ms

KNN query for range: 50.0, K : 50
Query Time : 220.0000 ms

KNN query for range: 50.0, K : 60
Query Time : 220.0000 ms

KNN query for range: 50.0, K : 70
Query Time : 363.0000 ms

KNN query for range: 50.0, K : 80
Query Time : 360.0000 ms

KNN query for range: 50.0, K : 90
Query Time : 362.0000 ms

KNN query for range: 50.0, K : 100
Query Time : 220.0000 ms

glsysen-yo ~/Desktop/Projects/SpatialData/Project 2/Codes/KDTree 7 main 3
./KDTree
KDTree Query!!!
Select Query Type
1. Range Query
2. KNN Query
3.
Select Dataset
1. clustered_dataset.txt
2. gaussian_dataset.txt
3. uniform_dataset.txt
4.
[KNN Query]
Test Query Point (x,y) : (250.000000, 250.000000)
KNN query for range 50.0, K : 1
Query Time : 1020.0000 ms

KNN query for range: 50.0, K : 10
Query Time : 2010.0000 ms

KNN query for range: 50.0, K : 20
Query Time : 2000.0000 ms

KNN query for range: 50.0, K : 30
Query Time : 2000.0000 ms

KNN query for range: 50.0, K : 40
Query Time : 2000.0000 ms

KNN query for range: 50.0, K : 50
Query Time : 2000.0000 ms

KNN query for range: 50.0, K : 60
Query Time : 2000.0000 ms

KNN query for range: 50.0, K : 70
Query Time : 2000.0000 ms

KNN query for range: 50.0, K : 80
Query Time : 2000.0000 ms

KNN query for range: 50.0, K : 90
Query Time : 2000.0000 ms

KNN query for range: 50.0, K : 100
Query Time : 2000.0000 ms

glsysen-yo ~/Desktop/Projects/SpatialData/Project 2/Codes/KDTree 7 main 3
./KDTree
KDTree Query!!!
Select Query Type
1. Range Query
2. KNN Query
3.
Select Dataset
1. clustered_dataset.txt
2. gaussian_dataset.txt
3. uniform_dataset.txt
4.
[KNN Query]
Test Query Point (x,y) : (250.000000, 250.000000)
KNN query for range 50.0, K : 1
Query Time : 14.0000 ms

KNN query for range: 50.0, K : 10
Query Time : 107.0000 ms

KNN query for range: 50.0, K : 20
Query Time : 103.0000 ms

KNN query for range: 50.0, K : 30
Query Time : 100.0000 ms

KNN query for range: 50.0, K : 40
Query Time : 111.0000 ms

KNN query for range: 50.0, K : 50
Query Time : 111.0000 ms

KNN query for range: 50.0, K : 60
Query Time : 100.0000 ms

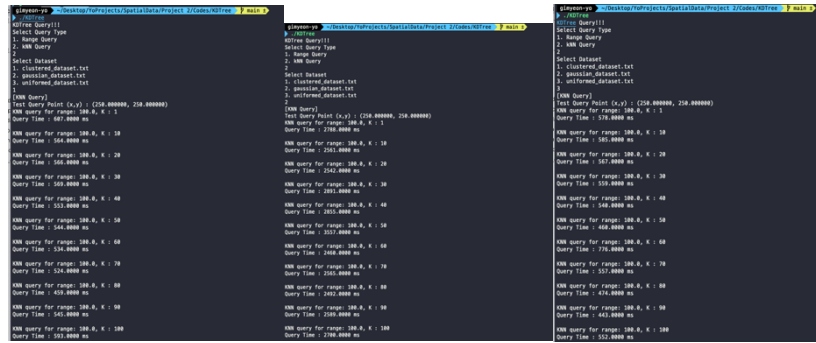
KNN query for range: 50.0, K : 70
Query Time : 111.0000 ms

KNN query for range: 50.0, K : 80
Query Time : 100.0000 ms

KNN query for range: 50.0, K : 90
Query Time : 100.0000 ms

KNN query for range: 50.0, K : 100
Query Time : 100.0000 ms
```

#### ㄷ. R = 100, (Clustered, Gaussian, Uniform 순서)



**결과 분석 :** 우선, 전체적으로 Bruteforce 에 비해서는 훨씬 효율성이 높은 결과를 보인다.

당연히 radius 가 높아질수록 더 오래 걸리는 결과를 확인할 수 있었고,

gaussian 같이 데이터가 불균형한 데이터에 대해서 가장 오래 걸리는 효율성을 확인했다.

그 반대로 uniform data 와 같이 균일한 데이터 분포에서 가장 빠른 시간을 보였는데, 이는 KD Tree 의 특성상 데이터가 균일할 수록 더 pruning 과정에서 많은 데이터들을 prune 할 수 있기 때문이다.

**<감사합니다.>**