Read Me KD-Tree

Creator : George Maroulis, gmaroulis@ceid.upatras.gr

Requirements :

- Python 3.6 or greater
- Modules :
    - numpy
    - time
    - random

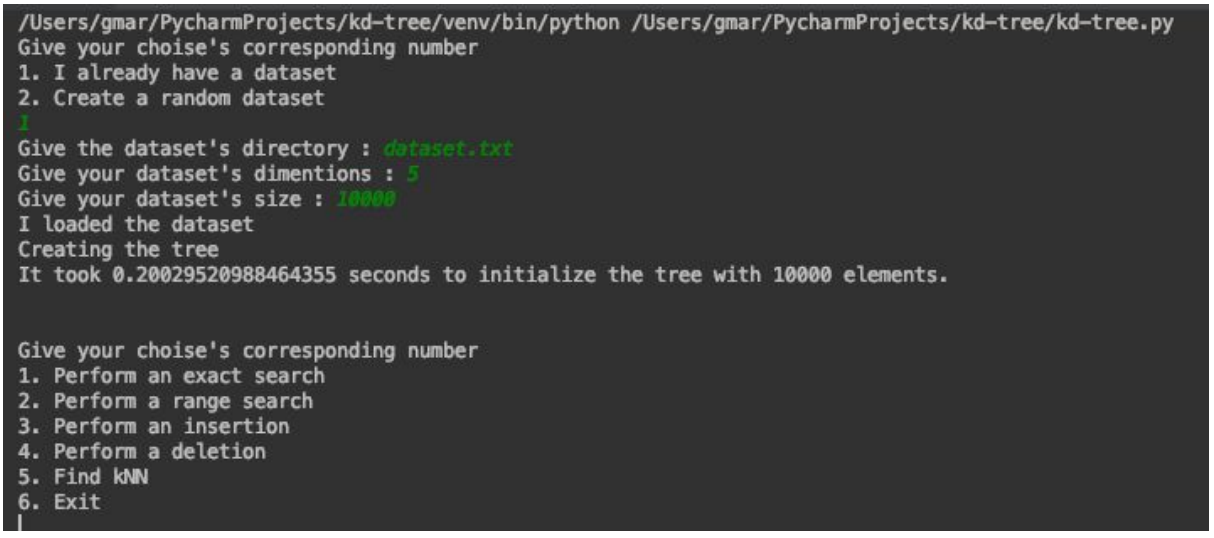
The tree supports all the basic functions that you would expect from a kd-tree :

- Create
- Insert
- Delete
- Exact match
- Range search
- kNN

How to load your data :

When executed the code will prompt you to choose if you want to use your own data or you want the script to generate data for you.

Those are the two paths you can take :



```
/Users/gmar/PycharmProjects/kd-tree/venv/bin/python /Users/gmar/PycharmProjects/kd-tree/kd-tree.py
Give your choise's corresponding number
1. I already have a dataset
2. Create a random dataset
2
Give me the number of elements you want me to create : 100000
Give the dimentions you prefer : 6
Give me the max value of the elements : 1
Give me the min value of the elements : -1
I created a dataset named dataset.txt in ./
Creating the tree
It took 2.3074920177459717 seconds to initialize the tree with 100000 elements.


Give your choise's corresponding number
1. Perform an exact search
2. Perform a range search
3. Perform an insertion
4. Perform a deletion
5. Find kNN
6. Exit
```

The dataset should be of the type :

- Every line is an element
- The values of different dimensions of an element are seperated with a white space

```
0.914674651378762 0.19171138308898628 -0.4115527044892364 0.8360332176500125 -0.3035159733656838 -0.7038192826337368
0.04386862542572523 0.9374267027808978 0.6755489098962755 -0.44530229472760663 0.9282052861542305 0.4488804774662254
-0.3777656360316526 0.9932203409878391 0.34238699110258697 0.6851630273169504 -0.8551050277598828 -0.6552901523548225
-0.9066301163975521 -0.06256000700130881 -0.40604070662325786 -0.5119157450751204 -0.8351934596503858 0.7594905354034012
0.8898620020647396 0.1458662251558862 -0.0295805114468084 0.7853768874068334 -0.7031327440393076 0.039429756273304584
-0.7123214408108216 -0.5833342338397056 -0.7361747699299441 -0.8971159369179367 -0.9202945628776873 0.48978193450458374
-0.2533776095261866 -0.11198117083533599 0.857925556443413 0.6590757616420095 0.6574876804223342 -0.699975472714105
```

7 elements with 6 dimensions each

How to perform an exact search :

In the previous menu you press 1 and enter, then you are asked to give the value of each dimension of the element you are searching. The function will return 1 if the element is in the dataset, 0 if it isn't.

```
Give your choise's corresponding number
1. Perform an exact search
2. Perform a range search
3. Perform an insertion
4. Perform a deletion
5. Find kNN
6. Exit
1
Give me the value for dimention 1 : 1
Give me the value for dimention 2 : 1
Give me the value for dimention 3 : 1
Give me the value for dimention 4 : 1
Give me the value for dimention 5 : 1
Give me the value for dimention 6 : 1
Not in the dataset
It took 6.985664367675781e-05 seconds to perform an exact search.
```

How to perform a range search :

In the previous menu you press 2 and enter, then you are asked to give the value of each dimension of the left bound , and then to give the value of each dimension of the right bound. The function will return a list with the elements in the above range, you can ether inspect the elements or go back to the menu.

```
Give me the value for dimention 1 for min : 0
Give me the value for dimention 2 for min : 0
Give me the value for dimention 3 for min : 0
Give me the value for dimention 4 for min : 0
Give me the value for dimention 5 for min : 0
Give me the value for dimention 6 for min : 0
Give me the value for dimention 1 for max : 1
Give me the value for dimention 2 for max : 1
Give me the value for dimention 3 for max : 1
Give me the value for dimention 4 for max : 1
Give me the value for dimention 5 for max : 1
Give me the value for dimention 6 for max : 1
Found 1504 elements in that range
It took 0.014880895614624023 seconds to perform an range search.
1. Show me the results and move on
2. Move on
2
```

How to perform an insertion :

In the previous menu you press 3 and enter, then you are asked to give the value of each dimension of the element you want to insert.

```
Give your choise's corresponding number
1. Perform an exact search
2. Perform a range search
3. Perform an insertion
4. Perform a deletion
5. Find kNN
6. Exit
3
Give me the value for dimention 1 to insert : 1
Give me the value for dimention 2 to insert : 1
Give me the value for dimention 3 to insert : 1
Give me the value for dimention 4 to insert : 1
Give me the value for dimention 5 to insert : 1
Give me the value for dimention 6 to insert : 1
Inserted successfully
It took 6.198883056640625e-05 seconds to perform an insertion.
```

How to perform a deletion :

In the previous menu you press 4 and enter, then you are asked to give the value of each dimension of the element you want to delete.

```
Give your choise's corresponding number
1. Perform an exact search
2. Perform a range search
3. Perform an insertion
4. Perform a deletion
5. Find kNN
6. Exit
4
Give me the value for dimention 1 to delete : 1
Give me the value for dimention 2 to delete : 1
Give me the value for dimention 3 to delete : 1
Give me the value for dimention 4 to delete : 1
Give me the value for dimention 5 to delete : 1
Give me the value for dimention 6 to delete : 1
Element deleted
It took 0.001577138900756836 seconds to delete an element.
```

How to find kNN :

In the previous menu you press 5 and enter, then you are asked to give the value of each dimension of the element you want to find its neighbors, and the you are asked for how many neighbors you want it to find. The function return a list with the k neighbors.

```
Give your choise's corresponding number
1. Perform an exact search
2. Perform a range search
3. Perform an insertion
4. Perform a deletion
5. Find kNN
6. Exit
5
Give me the value for dimention 1 to find kNN : 1
Give me the value for dimention 2 to find kNN : 1
Give me the value for dimention 3 to find kNN : 1
Give me the value for dimention 4 to find kNN : 1
Give me the value for dimention 5 to find kNN : 1
Give me the value for dimention 6 to find kNN : 1
How many neighbors : 9
I found [(0.8113512316841356, 0.9106549217629358, 0.8094665313469136,
  0.8269279752278069, 0.6585466609131603, 0.5344507830982701),
  (0.6971402144731058, 0.9763413871022746, 0.48280276705618275,
  0.5339893150015229, 0.5888631405194986, 0.9290036377758324),
  (0.8439714765698791, 0.5404647536974914, 0.8089605705191807,
  0.7058459243230548, 0.5751512708553093, 0.5157636492469184),
  (0.5514240858689838, 0.8134802918074793, 0.7478713946892777,
  0.6033493281657472, 0.8278754384889191, 0.680409949538288),
  (0.6848408227612548, 0.7630499971005433, 0.9623335163800217,
  0.6547458377261501, 0.556235777015442, 0.24640033377969717),
  (0.9254900572395899, 0.8739838759049661, 0.8519546070530883,
  0.9378433217605733, 0.3609343944342689, 0.4771724911598907),
  (0.5254143219081739, 0.6536872205018063, 0.9601919944293684,
  0.2286058901565211, 0.5840833226620863, 0.24099524875403233),
  (0.7730757516510609, 0.9761133145671679, 0.26742968146162327,
  0.572179034971035, 0.5379652004832229, 0.38441247700936576),
  (0.7747596578834299, 0.08411898644170623, 0.5279816185375759,
  0.8682332897530323, 0.7718888728516096, 0.028106291370630032)] 9NN
  in 0.00037097930908203125 second
```