Homework 2 Prob 6.

Double Click here to edit this cell

- Name: 김동규
- Student ID: 201800615
- Submission date: 20230411

You must run this homework code on Google Colab

- DON'T run on Google Colab Pro
- DON'T use GPU or TPU

Remark. If any kind of loops including for-loop, while-loops, list comprehension, and other loops are found, you get no points (0점).

Use numpy wherever it is possible.

Total: 30 pts

You must run the following two cells to make sure you are running on Google Colab

```
!cat /proc/cpuinfo
```

```
processor       : 0
vendor_id       : AuthenticAMD
cpu family      : 23
model           : 49
model name      : AMD EPYC 7B12
stepping        : 0
microcode       : 0xffffffff
cpu MHz         : 2249.998
cache size      : 512 KB
physical id     : 0
siblings        : 2
core id         : 0
cpu cores       : 1
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx mmxext fxsr
bugs            : sysret_ss_attrs null_seg spectre_v1 spectre_v2 spec_store_bypass retbleed
bogomips        : 4499.99
TLB size        : 3072 4K pages
clflush size    : 64
cache_alignment : 64
address sizes   : 48 bits physical, 48 bits virtual
power management:

processor       : 1
vendor_id       : AuthenticAMD
cpu family      : 23
model           : 49
model name      : AMD EPYC 7B12
stepping        : 0
microcode       : 0xffffffff
cpu MHz         : 2249.998
cache size      : 512 KB
physical id     : 0
siblings        : 2
core id         : 0
cpu cores       : 1
apicid          : 1
initial apicid  : 1
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx mmxext fxsr
```

```
bugs            : sysret_ss_attrs null_seg spectre_v1 spectre_v2 spec_store_bypass retbleed
bogomips        : 4499.99
TLB size        : 3072 4K pages
clflush size    : 64
cache_alignment : 64
address sizes   : 48 bits physical, 48 bits virtual
power management:
```

```
!cat /proc/meminfo
```

```
MemTotal:       13297200 kB
MemFree:         8833160 kB
MemAvailable:   12432528 kB
Buffers:          189476 kB
Cached:          3582420 kB
SwapCached:            0 kB
Active:           381240 kB
Inactive:        3856360 kB
Active(anon):       1040 kB
Inactive(anon):   460060 kB
Active(file):     380200 kB
Inactive(file):  3396300 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:             0 kB
SwapFree:              0 kB
Dirty:             18648 kB
Writeback:             0 kB
AnonPages:        465600 kB
Mapped:           250676 kB
Shmem:              1348 kB
KReclaimable:     128640 kB
Slab:             157984 kB
SReclaimable:     128640 kB
SUnreclaim:        29344 kB
KernelStack:        4208 kB
PageTables:         6084 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:     6648600 kB
Committed_AS:    2641440 kB
VmallocTotal:   34359738367 kB
VmallocUsed:        9060 kB
VmallocChunk:          0 kB
Percpu:             1328 kB
HardwareCorrupted:     0 kB
AnonHugePages:         0 kB
ShmemHugePages:        0 kB
ShmemPmdMapped:        0 kB
FileHugePages:         0 kB
FilePmdMapped:         0 kB
CmaTotal:              0 kB
CmaFree:               0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
Hugetlb:               0 kB
DirectMap4k:       84800 kB
DirectMap2M:     4106240 kB
DirectMap1G:    11534336 kB
```

Problem 6 (20 pts)

- **find_k_nearest_index_big** returns the index of the k-nearest for 50 million data
- We want to time the execution
- *Do not use sklearn, scipy or any module computing k-nearest points directly*
- Use numpy functions only

```
import numpy as np

def find_k_nearest_index_big(data, center, k=1):
    result=kdtree.build(center,data,k)

class kdtree:
    def __init__(self,data):
      self.root=self.build(data)

    def build(self,data,depth=0):
      n=data.shape[0]
```

```
        if n==0:
          return None
        dim = depth%np.data.shape[1]
        mid=n//2
        data=data[data[:,dim].argsort()]
        node={
            'point': data[mid],
            'left': self.build(self,data[:mid],depth+1),
            'right': self.build(self,data[mid+1:],depth+1)
            }

      def query(self,point,k=1):
        bestpoints=[None]*data.shape[1]
        bestdists=[np.inf]*data.shape[1]


        def recursive(node):
          if node is None:
            return None
          dist = np.linalg.norm(point - node['point'])
          if dist<bestdists:
            bestdists[0]=dist
            bestpoints[0]=node['point']
            sortedindex=np.argsort(bestdists)
            bestdists=[bestdists[i] for i in sortedindex]
            bestpoints=[bestpoints[i] for i in sortedindex]
          dim=node['point'].shape[0]
          diff=point[dim%data.shape[1]]-node['point'][dim%data.shape[1]]
          if diff<=bestdists[0]:
            recursive(node['left'])
          if diff>=bestdists[0]:
            recursive(node['right'])
          recursive(self.root)
          return bestpoints(self.root)
```

```
# DO NOT EDIT THIS CELL

import time

np.random.seed(100)
data = np.random.randn(30000000,20)   # 30 million data
k = 5
center = np.random.randn(20)
start = time.time()
print(find_k_nearest_index_big(data, center, k))
end = time.time()

lapse = end - start
total = 10
weight = 1.5
grace = 20
my_point = int(total / (weight ** (lapse // grace)))
print(f'Total time taken : {lapse} seconds')
print(f'My point is {my_point}')
```

Your time must be around:

```
[ ___________________________________________ ]
Total time taken : 5.2564404010772705 seconds
My point is 10
```

```
# DO NOT EDIT THIS CELL

import time

np.random.seed(100)
data = np.random.randn(50000000,20)   # 50 million data
k = 5
center = np.random.randn(20)
start = time.time()
print(find_k_nearest_index_big(data, center, k))
end = time.time()

lapse = end - start
total = 10
weight = 1.5
grace = 20
my_point = int(total / (weight ** (lapse // grace)))
```

```
print(f'Total time taken : {lapse} seconds')
print(f'My point is {my_point}')
```

Your time must be around:

```
[ ____________________________________________ ]
Total time taken : 9.744923830032349 seconds
My point is 10
```

```
# DO NOT EDIT THIS CELL

import time

np.random.seed(100)
data = np.random.randn(70000000,20)   # 70 million data
k = 5
center = np.random.randn(20)
start = time.time()
print(find_k_nearest_index_big(data, center, k))
end = time.time()

lapse = end - start
total = 10
weight = 1.5
grace = 40
my_point = int(total / (weight ** (lapse // grace)))
print(f'Total time taken : {lapse} seconds')
print(f'My point is {my_point}')
```

Your time must be around:

```
[ ____________________________________________ ]
Total time taken : 20.60638999938965 seconds
My point is 10
```

Ethics:

If you cheat, you will get negatgive of the total points. If the homework total is 22 and you cheat, you get -22.

What to submit

- Run **all cells** after restarting the kernel
- Goto "File -> Print Preview"
- Print the page as pdf
- Submit the pdf file in google classroom
- Pdf file name must be in a form of: homework_2_prob6_홍길동_202300001.pdf
- No late homeworks will be accepted
- Your homework will be graded on the basis of correctness, performance, and programming skills

11초 오후 11:52에 완료됨