import important reminders

- All homeworks are to be done INDIVIDUALLY.
- All written answers should be TYPED.
- For code submission to blackboard, make three directories /Q1, /Q2, /Q4, and then put your code for question 1, 2, 4 to the corresponding directory (Q3 has no code to deliver). Then tar them, compress them into a file ([andrew-id].tar.gz) and submit it to blackboard. As always, make sure you exclude redundant/derived files, in your tar-file.

grading

- Weight: 30% of total homeworks weight = 3% of course weight
- Q1,2: graded by Yuning He
- Q3,4: graded by Yan Zhang

re-grade requests

In case you believe you deserve more points, please
- contact the corresponding TA, only
- in writing/email,
- within 1 week from the return of the homework
- specifying your name and andrew-id, and
  1. The question you are disputing (e.g., 'HW1-Q1')
  2. The reason (e.g., 'I forgot to print the total number of rectangles')
  3. The extra points you are asking (e.g., '5 more points out of 20')
In the remote case that there is still a disagreement, please contact the instructor.
Q1 – R-Tree [40 pts]

Print answers on separate page, with ‘[course-id] [hw#] [question#] [andrew-id] [your name]’

Problem Description: The goal is to become familiar with the R-Tree algorithm. Your task is to add new functionality to the provided R-Tree package from http://www.cs.cmu.edu/~christos/courses/826.F14/HOMEWORKS/HW1/Q1/drtree.tar.gz

Setup: Please build the R-Tree Package

```
tar -xvf drtree.tar.gz; cd DRTree; make demo
```

This creates the `bin/DRmain` program. Run it on some small datasets and have some fun! It has been tested on the Unix/linux platform on the andrew machines - it most probably runs under mac-osx, and Cygwin on Windows. Running

```
make hw1
```

should load the appropriate dataset, and print ‘Algorithm not implemented’ message. Currently the R-tree package supports ‘s’ for range search, ‘i’ for insertion etc.

Implementation Details: You are required to implement the so-called Ring-Search. Figure 1 shows a 2-d scenario: given an outer rectangle and an inner rectangle, return all the rectangles that intersect (or touch, even at a single, corner point) the shaded area between these two rectangles. Thus, for Figure 1 you code should return all rectangles, except F.

![Diagram of ring-search](image)

Figure 1: Example of “ring-search”: It should return all the data rectangles that intersect, or touch, the shaded area, i.e., all rectangles, except F.

Implement the command ‘g’ for ‘rin(g)-search’ using the input dataset given in `hw1.input` (which is called by default with `make hw1`). Your code should work for any dimensionality

---

1FYI, the ‘D’ stands for ’deferred split’ R-tree - but you don’t need to worry about that.
\(d (d = 1, 2, 3, \ldots).\)

**Hint:**
- Do explore the range query functions in the code package. Modify or follow the code to implement your own functionality.
- For the final results, run `make spotless` before you run `make hw1` to clear the index file.

**Input Format:** Next, we give 2-d examples, but as we said, your code should work for *any* dimensionality, like the rest of the R-Tree code. All the input parameters are in the same line

\[
g \text{ outer-x-low} \text{ outer-x-high} \text{ outer-y-low} \text{ outer-y-high} \\
\text{ inner-x-low} \text{ inner-x-high} \text{ inner-y-low} \text{ inner-y-high}
\]

For example, one valid input could be

\[
g \ 100 \ 600 \ 100 \ 600 \ 200 \ 300 \ 300 \ 400
\]

**Output Format:** Your program should print
- the total number \(N\) of qualifying MBRs on the first line,
- and then, \(N\) lines, one for each qualifying data rectangle (record number, and their coordinates \((x\text{-low}, x\text{-high}, y\text{-low}, y\text{-high})\)) in tab-separated (tsv) format.
- We will also accept the total number \(N\) of qualifying MBRs, to be printed in the last line. (Because this will make your coding easier)
Solution:

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>98</td>
<td>492</td>
<td>981</td>
<td>488</td>
<td>971</td>
<td></td>
</tr>
<tr>
<td>3571</td>
<td>965</td>
<td>978</td>
<td>109</td>
<td>973</td>
<td></td>
</tr>
<tr>
<td>4368</td>
<td>928</td>
<td>984</td>
<td>107</td>
<td>987</td>
<td></td>
</tr>
<tr>
<td>2831</td>
<td>900</td>
<td>981</td>
<td>677</td>
<td>984</td>
<td></td>
</tr>
<tr>
<td>339</td>
<td>765</td>
<td>975</td>
<td>634</td>
<td>985</td>
<td></td>
</tr>
<tr>
<td>2029</td>
<td>275</td>
<td>996</td>
<td>326</td>
<td>974</td>
<td></td>
</tr>
<tr>
<td>1507</td>
<td>322</td>
<td>999</td>
<td>961</td>
<td>995</td>
<td></td>
</tr>
<tr>
<td>4824</td>
<td>341</td>
<td>987</td>
<td>585</td>
<td>979</td>
<td></td>
</tr>
<tr>
<td>184</td>
<td>858</td>
<td>998</td>
<td>655</td>
<td>993</td>
<td></td>
</tr>
<tr>
<td>4783</td>
<td>596</td>
<td>989</td>
<td>877</td>
<td>983</td>
<td></td>
</tr>
<tr>
<td>3057</td>
<td>616</td>
<td>971</td>
<td>515</td>
<td>992</td>
<td></td>
</tr>
<tr>
<td>2602</td>
<td>282</td>
<td>997</td>
<td>64</td>
<td>983</td>
<td></td>
</tr>
<tr>
<td>2745</td>
<td>193</td>
<td>987</td>
<td>665</td>
<td>984</td>
<td></td>
</tr>
<tr>
<td>681</td>
<td>571</td>
<td>977</td>
<td>136</td>
<td>983</td>
<td></td>
</tr>
<tr>
<td>4351</td>
<td>134</td>
<td>976</td>
<td>394</td>
<td>998</td>
<td></td>
</tr>
<tr>
<td>882</td>
<td>165</td>
<td>987</td>
<td>12</td>
<td>994</td>
<td></td>
</tr>
<tr>
<td>1326</td>
<td>361</td>
<td>975</td>
<td>206</td>
<td>985</td>
<td></td>
</tr>
<tr>
<td>1685</td>
<td>484</td>
<td>976</td>
<td>475</td>
<td>991</td>
<td></td>
</tr>
<tr>
<td>980</td>
<td>296</td>
<td>988</td>
<td>686</td>
<td>973</td>
<td></td>
</tr>
<tr>
<td>1364</td>
<td>321</td>
<td>982</td>
<td>651</td>
<td>987</td>
<td></td>
</tr>
<tr>
<td>2499</td>
<td>211</td>
<td>989</td>
<td>204</td>
<td>970</td>
<td></td>
</tr>
<tr>
<td>2020</td>
<td>81</td>
<td>979</td>
<td>221</td>
<td>979</td>
<td></td>
</tr>
<tr>
<td>122</td>
<td>59</td>
<td>988</td>
<td>820</td>
<td>996</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Grading info:

- -1 pt if one data rectangle is wrong
- -2 pts if incorrect format (e.g. missing record number)
- -3 pts if total number is wrong
- -5 pts if code doesn't handle different dimensionality
- -5 pts if duplication (no make spotless before final run)
- -10 pts if code is missing or isn't runnable
Q2 – Hilbert and Z-ordering [30 pts]

Print answers on separate page, with ‘[course-id] [hw#] [question#] [andrew-id] [your name]’

Problem Description: To goal is to become familiar with Hilbert and z-ordering algorithm. In the following tasks, we assume the ordering of both Hilbert and z-curve follows what you have seen in the lecture.

1. [10 pt] Write a program to compute the $z_{next}$ coordinates. $z_{next}$ means the next point on the z-curve given the coordinates. The command-line syntax should be: $z_{next}$ -n <order-of-curve> -d <dimension-of-curve> <x1> <x2> ... <xd>. Thus:
   - $z_{next}$ -n 2 -d 2 0 0 # should return 0 1 - going vertically!
   - $z_{next}$ -n 2 -d 3 0 0 0 # should return 0 0 1 - going vertically, again

Solution:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

Grading info:
- 0 pt for test case 0
- 2 pts for each following test case

2. [10 pt] Write a program to compute the $h_{next}$ coordinates. $h_{next}$ means the next point on the Hilbert-curve given the coordinates. To simplify the problem, we assume it is a 2-d Hilbert curve. The format should be:
   - $h_{next}$ -n 2 0 0 # should return 1 0
   - $h_{next}$ -n 2 0 1 # should return 0 2
   - $h_{next}$ -n 3 0 0 # should return 0 1

Solution:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>30</td>
<td>7</td>
</tr>
</tbody>
</table>

Grading info:
- 0 pt for test case 0
- 2 pts for each following test case
3. [5 pt] Give the results of your program on the input files [http://www.cs.cmu.edu/~christos/courses/826.F14/HOMEWORKS/HW1/Q2/input.tar.gz](http://www.cs.cmu.edu/~christos/courses/826.F14/HOMEWORKS/HW1/Q2/input.tar.gz). Make sure you echo the input, so that it is clear which answer refers to which input.

**Solution:**

**Grading info:**

- -2 pts for not echoing the input
- -3 pts for not checking corner cases

4. [5 pt] Using your programs, write code to plot a z-curve and a Hilbert curve of order 6 (64 * 64 grid) and dimension 2. For plotting, we recommend gnuplot, but anything else that runs on the linux/andrew machines, is fine.

![Figure 2: z-curve](image1)

![Figure 3: hilbert-curve](image2)
**Solution:**
Shown as Figure 2 and Figure 3.

**Grading info:**
- -2 pts for one wrong plot
- -5 pts for two wrong plots

**Hint:**
- For Hilbert curve, we recommend the code/algorithm from the following papers (click below, to get their pdf)
  - Jagadish [SIGMOD 90]
  - Roseman+ [PODS 89]
- Make your code robust and guard against all the corner cases. E.g. negative coordinates, non-integer input, coordinates out of range, etc.
Q3 – Fractal Detectives [15 pts]

Print answers on separate page, with ‘[course-id] [hw#] [question#] [andrew-id] [your name]’

Problem Description: In this problem we will see how the fractal dimension can help us guess some properties of a cloud of points.

Suppose that a physicist colleague of yours (say, ‘Mike’), has been doing experiments, measuring $M=6$ attributes, like pressure, temperature, etc., in $N$ different settings. Thus, he has a file with $N$ rows and $M$ numbers per row, and he suspects that there are correlations among the $M$ variables, obeying a yet-to-be-discovered physics law. The goal is to help ‘Mike’ as much as you can.

Example: if the measurements were about the gravitational force, which obeys Newton’s law

$$F = \frac{C \cdot m_1 \cdot m_2}{r^2}$$

then, you would have $M=4$ attributes: force $F$, 2 masses $m_1, m_2$, distance $r$; and 3 degrees of freedom: $m_1, m_2, r$. Thus, the intrinsic (fractal) dimension of such a dataset, should be close to 3.

Implementation Details: Download 5 datasets http://www.contrib.andrew.cmu.edu/~yanzhan2/5-mystery-data.tar.gz and answer the following questions for each of the 5 datasets.

1. [1 pt] Plot the correlation integral for the dataset. We recommend the FDNQ package http://www.cs.cmu.edu/~christos/SRC/fdnq_h.zip.

2. [1 pt] (a) Are there any correlations among the $M$ variables (yes/no)?
(b) Are there clusters in Mike’s cloud of points (yes/no)?
(c) What can you say about the intrinsic dimensionality of the dataset? (give a number, or say ‘undefined’, if there are clusters).

3. [1 pt] Based on the correlation integral alone, which of the choices are plausible, among [A]-[I], below? Report all that apply.

4. [0 pt] (Extra question - no points - just the admiration of the teaching staff :) ) What else can you tell Mike about his dataset, to help him discover a new physics law? Eg., can you say that, in some appropriate projection, his cloud of points looks like a point? or line? or sinusoid? or spiral? Even harder question: Can you guess the equations we used, to generate Mike’s ’mystery’ dataset?

The choices of possible shapes of each ‘mystery’ dataset, are:

(A) 1-d: CIRCLE: periphery of a circle, embedded in $M$-d space
(B) 2-d: DISK: a 2d disk, embedded in $M$-d space
(C) **SIERPINSKI**: a Sierpinski triangle, embedded in $M$-d space
(D) **3-d: CUBE**: a cube, embedded in $M$-d space
(E) **3-d: FOOTBALL**: the cloud is 3-d ellipsoid (elongated sphere, like an American football), embedded in $M$-d space.
(F) **3-d: PYRAMID**: the points form a 3-d pyramid (embedded in $M$-d space)
(G) **UNIFORM**: a cloud of points, uniformly distributed in $M$-d space
(H) **CLUSTERS**: the data points are clustered in $C$ clusters
(I) **NONE**: none of the above

**What to turn in:**
- **Code**: No code to turn in.
- **Answers**: On hard copy, submit your plots for Q3.1, and your *typed* answers for Q3.2-Q3.4, for all the 5 ‘mystery’ datasets.
Solution:

1. correlation integrals

2. (a) yes*5
   (b) only dataset2 has clusters
   (c) 1.54, undefined, 1.92, 2.90, 0.97

3. (C),(H),(B),(DEF),(A)

Grading info:
-1 pt if answer is wrong for each question each dataset
Q4 – Correlation Integral [15 pts]

Print answers on separate page, with ‘[course-id] [hw#] [question#] [andrew-id] [your name]’

**Problem Description:** The goal is to gain a stronger intuition about the fractal dimension and the correlation integral. We want to generate a synthetic dataset, whose correlation integral will match the one of Figure 4. Let $N$ be the number of points in this yet-to-be-created dataset. Notice the break-points at radius $r_1 = 10^{-6}$, $r_2 = 10^{-5}$, $r_3 = 10^{-3}$ and $r_4 = 10^{-1}$.

![Figure 4: target correlation integral](image)

1. [1 pt] Given that the count of points is $N$, what is be the value of $y_1$? We are looking for an answer like $N^3$, or $\sqrt{N}$, or $N/10^5$, etc.

2. [1 pt] What would be the value of $y_3$ (again, as a function of the count of data points $N$)?

3. [3 pt] What is the only value of $N$, that could generate the correlation integral of Figure 4? *Hint*: estimate the difference $y_3 - y_1$ as a function of $N$; derive an alternative estimation, as a plain number, using the information from the slopes and from $r_1, \ldots, r_4$.

4. [3 pt] As we learned in class, a plateau is an indication of clusters. What is your estimate for the count of clusters $C$ in the dataset of Figure 4? We expect a number like, e.g., $2^{10}$. 
5. [4 pt] Write code, to generate your own 2-dimensional dataset that would have the correlation integral of Figure 4. (There are many correct answers - any one of them is fine).

6. [3 pt] Draw and print the correlation integral of your dataset. Please also plot grid-lines, and show coordinates in log-base-10 to make grading easier. Again, we recommend the FDNQ package.\[http://www.cs.cmu.edu/~christos/SRC/fdnq_h.zip\].

Hint: It is OK if your plot has rounded, instead of sharp corners, at the break points.

What to turn in:

- **Code:** Submit your code to generate the dataset - we recommend python, but any language is acceptable, as long as it runs on andrew/linux, and it has a makefile, so that
  - `make dataset` would generate your dataset and
  - `make` would generate the drawing `integral.pdf` of its correlation integral. We recommend pdf, but jpg, png, etc are all fine.

- **Answers:** a hard copy with
  1. your answers to Q4.1-2-3-4,
  2. a copy of your code for Q4.5, and
  3. the plot for Q4.6

\[2\] in gnuplot: `set grid; set logscale xy 10; set format x "10^{%L}"`
Solution:

1. $\log N$
2. $2 \log N$
3. $10^5$
4. $10^4$
5. see example code

Grading info:

-1 pt for wrong answer 4.1, 4.2
-3 pts for wrong answer 4.3, 4.4
-1 pt if plot does not following scale, but previous answer correct