Discussion 8 : Project 4 Exercises

Congratulations to Everyone for making it through Test 1

If you didn’t do as well as you would like, come to office hours to prepare for next test and for help on projects

Exercises:

Certify Heap

Ramanujan1

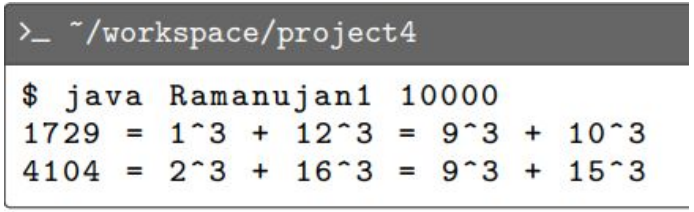
Ramanujan2

Exercise 1. (*Certify Heap*) Implement the static method `isMaxHeap()` in `CertifyHeap.java` that takes an array `a` of `Comparable` objects (excluding `a[0] = *`) and returns `true` if `a` represents a max-heap, and `false` otherwise.

```
>_ ~/workspace/project4

$ java CertifyHeap
* M A X H E A P
<ctrl-d>
false
$ java CertifyHeap
<ctrl-d>
* A A E H M P X
false
$ java CertifyHeap
<ctrl-d>
* X P M H E A A
true
```

```
import stdlib.StdIn;
import stdlib.StdOut;

public class CertifyHeap {
    // Returns true if a[] represents a max-heap, and false otherwise.
    public static boolean isMaxHeap(Comparable[] a) {
        // Set n to the number of elements in a.
        ...

        // For each node 1 <= i <= n / 2, if a[i] is less than either of its children, return
        // false, meaning a[] does not represent a max-heap. If no such i exists, return true.
        ...
    }

    // Returns true of v is less than w, and false otherwise.
    private static boolean less(Comparable v, Comparable w) {
        return (v.compareTo(w) < 0);
    }

    // Unit tests the library. [DO NOT EDIT]
    public static void main(String[] args) {
        String[] a = StdIn.readAllStrings();
        StdOut.println(isMaxHeap(a));
    }
}
```

Ramanujan1



```
>_ ~/workspace/project4
$ java Ramanujan1 10000
1729 = 1^3 + 12^3 = 9^3 + 10^3
4104 = 2^3 + 16^3 = 9^3 + 15^3
```

. (Ramanujan's Taxi) Srinivasa Ramanujan was an Indian mathematician who became famous for his intuition for numbers. When the English mathematician G. H. Hardy came to visit him one day, Hardy remarked that the number of his taxi was 1729, a rather dull number. To which Ramanujan replied, "No, Hardy! It is a very interesting number. It is the smallest number expressible as the sum of two cubes in two different ways." Verify this claim by writing a program Ramanujan1.java that accepts n (int) as command-line argument and writes to standard output all integers less than or equal to n that can be expressed as the sum of two cubes in two different ways. In other words, find distinct positive integers a, b, c, and d such that $a^3 + b^3 = c^3 + d^3 \leq n.$

```
import stdlib.StdOut;

public class Ramanujan1 {
    // Entry point.
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        ...
    }
}
```

- Use four nested `for` loops, with these bounds on the loop variables: $0 < a \leq \sqrt[3]{n}$, $a < b \leq \sqrt[3]{n - a^3}$, $a < c \leq \sqrt[3]{n}$, and $c < d \leq \sqrt[3]{n - c^3}$

Do not explicitly compute cube roots, and instead use `x * x * x < y` in place of `x < Math.cbrt(y)`.

Exercise 3. (*Ramanujan's Taxi Redux*) Write a program `Ramanujan2.java` that uses a minimum-oriented priority queue to solve the problem from Exercise 2.

```
>_ ~/workspace/project4

$ java Ramanujan2 10000
1729 = 1^3 + 12^3 = 9^3 + 10^3
4104 = 9^3 + 15^3 = 2^3 + 16^3
```

Directions:

- Initialize a min-PQ `pq` with pairs $(1,2), (2,3), (3,4), \ldots, (i, i+1)$, where $i < \sqrt[3]{n}$
- While i is not empty:
  - Remove the smallest pair (call it *current*) from `pq`.
  - Print the previous pair (k, l) and current pair (i, j) if $k^3 + l^3 = i^3 + j^3 \leq n$.
  - If $i < \sqrt[3]{n}$, insert the pair $(i, j+1)$ into `pq`.

Again, do not explicitly compute cube roots, and instead use `x * x * x < y` in place of `x < Math.cbrt(y)`.

```java
import dsa.MinPQ;
import stdlib.StdOut;

public class Ramanujan2 {
    // Entry point.
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        ...
    }

    // A data type that encapsulates a pair of numbers (i, j) and the sum of their cubes.
    private static class Pair implements Comparable<Pair> {
        private int i;          // first number in the pair
        private int j;          // second number in the pair
        private int sumOfCubes; // i^3 + j^3

        // Constructs a pair (i, j).
        public Pair(int i, int j) {
            this.i = i;
            this.j = j;
            sumOfCubes = i * i * i + j * j * j;
        }

        // Returns a comparison of pairs this and other based on their sum-of-cubes values.
        public int compareTo(Pair other) {
            return sumOfCubes - other.sumOfCubes;
        }
    }
}
```