课程报告 (作业 01)

谢鹏飞

（东南大学吴健雄学院，210089）

2021 年 3 月 5 日

1 代码原理

代码的核心部分就是如下的一个三重循环：

```
for (int i = 1; i < n; i++) {
    for (int j = i; j < n; j++) {
        for (int k = j; k < n; k++) {
            if (is_right_triangle(i, j, k))
                sql.INSERT(i, j, k);
        }
    }
}
```

其中每一级循环都从上一个循环中的指标开始，一直循环到终点，每一级循环都是线性的，因此时间复杂度应该为 $O(n^3)$.

至于空间复杂度，存储结构中利用了一个 `vector`，每个 `vector` 都由一个大小为 3 的数组构成，因此空间复杂度应为 $O(n*3)$.

2 统计运行时间

为了统计循环所用时间，分别在每次循环开始前和结束后记录了时间指标 `start` 和 `end`. 对时间进行处理后得到如下输出:

n	time
100	0.005
200	0.031
300	0.107
400	0.241
500	0.491
600	0.866
700	1.361
800	2.049
900	2.871
1000	3.893

利用 Python 对数据做了一些简单处理，拟合出一条平滑曲线，图像近似为一个三次函数，与前文所述的时间复杂度为 $O(n^3)$ 相符.

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

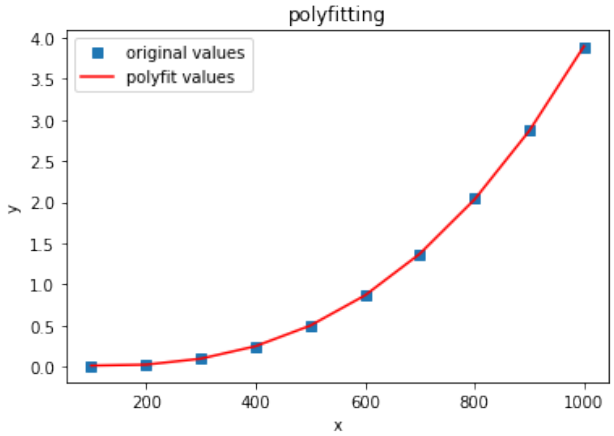
[2]:
```
key = [100,200,300,400,500,600,700,800,900,1000]
num = [0.005,0.031,0.107,0.241,0.491,0.866,1.361,2.049,2.871,3.893]
x = np.array(key)
y = np.array(num)
# 用 3 次多项式拟合
f1 = np.polyfit(x, y, 3)
p1 = np.poly1d(f1)
yvals = p1(x)
print("拟合后的方程为:\n",p1)
```

拟合后的方程为:

```
            3             2
3.175e-09 x + 1.125e-06 x - 0.0004441 x + 0.042
```

[3]:
```
plot1 = plt.plot(x, y, 's',label='original values')
plot2 = plt.plot(x, yvals, 'r',label='polyfit values')
plt.xlabel('x')
plt.ylabel('y')
```

```
plt.legend(loc=2)
plt.title('polyfitting')
plt.show()
```



3 Source Code

```
#include<iostream>
#include<ctime>
#include<vector>
#include<iomanip>
using namespace std;

//test if the three numbers are able to form a right triangle
//assert i < j < k
bool is_right_triangle(int i, int j, int k)
{
  if (i * i + j * j == k * k)
    return true;
  else
    return false;
}

//store the data
class database
{
```

```
20    vector<int*> db;
21  public:
22    //insert a new triangle into the db
23    void INSERT(int i, int j, int k)
24    {
25      int* temp = new int[3];
26      temp[0] = i; temp[1] = j; temp[2] = k;
27      db.push_back(temp);
28    }
29    void print()
30    {
31      int len = db.size();
32      for (int i = 0; i < len; i++) {
33        cout << db[i][0] << " "
34             << db[i][1] << " "
35             << db[i][2] << " "
36             << endl;
37      }
38      cout << "Total counts: " << len << endl;
39    }
40    int size()
41    {
42      return db.size();
43    }
44  };
45
46  void main()
47  {
48    for (int n = 100; n <= 1000; n += 100)
49    {
50      database sql;
51      clock_t start, end; //set start and end indicator to
52      start = clock();  //caculate the time consumed
53      for (int i = 1; i < n; i++) {
54        for (int j = i; j < n; j++) {
55          for (int k = j; k < n; k++) {
56            if (is_right_triangle(i, j, k))
57              sql.INSERT(i, j, k);
58          }
59        }
60      }
```

```
61        end = clock();
62        //sql.print();
63        /*if you want to print all the solutions
64        please uncomment the line above*/
65
66        cout << "n: " << setw(4) << n
67             << "  time: " << setprecision(5)
68             << (double)(end - start) / CLOCKS_PER_SEC
69             << endl;
70      }
71    }
72    /* Output:
73    n :  100  time : 0.005
74    n :  200  time : 0.031
75    n :  300  time : 0.107
76    n :  400  time : 0.241
77    n :  500  time : 0.491
78    n :  600  time : 0.866
79    n :  700  time : 1.361
80    n :  800  time : 2.049
81    n :  900  time : 2.871
82    n : 1000  time : 3.893
83    */
```