算法基础 Lab 2

求最近点对算法

王世炟

PB20151796

October 24, 2022

Part 1: 实验要求

本次实验要求我们完成**求最近点对算法**的模型实现，并对 data.txt 中的数据进行测试。具体需要完成以下部分：

- 完成基础的暴力搜索算法
- 完成分治算法
- 对比两种算法的时间

Part 2: 暴力搜索算法

参考教材 33.4 节的内容，可以写出以下流程：

Algorithm 1 Algorithm of ViolentFind

ViolentFind(Points)

1: Calculate distances of every pair of Points
2: Find the smallest distance

可以容易的将上述流程转化成代码：

```
1  double ViolentFind(Point t[], int length)
2  {
3      double dis;
4      double temp = Distance(t[0], t[1]);
5      pos1 = 0;
6      pos2 = 1;
7      for (int i = 0; i < length; i++)
8      {
9          for (int j = i + 1; j < length; j++)
10         {
```

```
            dis = Distance(t[i], t[j]);
            if (temp > dis)
            {
                temp = dis;
                pos1 = i;
                pos2 = j;
            }
        }
    }
    return temp;
}
```

Part 3: 分治算法

分治算法的思想就是现将原问题分解为子问题，求解子问题的解之后再进行合并，得到原问题的解。在本题中，具体步骤如下：

函数声明：

```
double NearestPair(Point tempX[], Point tempY[], int length, Point &a,
Point &b)
```

3.1 分解递归

分解：找出一条垂直线 l, 它把点集 P 对分为满足下列条件的两个集合 P_L 和 P_R：使得 $|P_L| = \lceil \frac{|P|}{2} \rceil$, $|P_R| = \lfloor \frac{|P|}{2} \rfloor$, P_L 中所有点在 l 的左侧或在直线上，P_R 中所有点在 l 的右侧或在直线上。数组 X 被划分为两个数组 X_L 和 X_R ，分别包含 P_L 和 P_R 中的点，并按 x 坐标单调递增的顺序进行排序；同理对 Y 如此操作。之后解决两个子问题。

源代码：

```
    Point ptl[length];
    Point ptr[length];
    Point ptrY[length];
    Point ptlY[length];
    double mid = tempX[(length - 1) / 2].x;
    for (i = 0; i <= (length - 1) / 2; i++)
    {
        ptl[i] = tempX[i];
    }
    i = (length - 1) / 2 + 1;
    for (j = 0 ; i < length; i++)
    {
```

```
13          ptr[j++] = tempX[i];
14      }
15      for (i = 0; i < length; i++)
16      {
17          if (tempY[i].x <= mid)
18          {
19              ptlY[i] = tempY[i];
20          }
21          else
22          {
23              ptrY[i] = tempY[i];
24          }
25      }
26
27      d1 = NearestPair(ptl, ptlY, (length - 1) / 2 + 1, a1, b1);
28      d2 = NearestPair(ptr, ptrY, length - (length - 1) / 2 - 1, a2, b2);
29      if (d1 < d2)
30      {
31          dis = d1;
32          a = a1;
33          b = b1;
34      }
35      else
36      {
37          dis = d2;
38          a = a2;
39          b = b2;
40      }
```

3.2 合并

最近点要么是某次递归调用找出的距离为 δ 的点对，要么是 P_L 中的一个点 P_R 中的一个点组成的点对。并且由教材讲述，只需要寻找 p 在 Y' 中紧随其后的 7 个点，并计算距离，与对短距离比较，找出新的最短距离即是原问题的答案。

```
1  Point ptm[length];
2  int k = 0;
3  for (i = 0; i < length; i++)
4  {
5      if (abs(tempY[i].x - mid) <= dis)
6      {
7          ptm[k++] = tempY[i];
```

```
    }
}

for (i = 0; i < k; i++)
{
    if (ptm[i].x - mid > 0)
    //p 在右半部分，找随后7个点中在左半部分的点
    {
        for (j = i + 1; j <= i + 7 && j < k; j++)
        {
            if (ptm[j].x - mid <= 0)
            {
                if (Distance(ptm[i], ptm[j]) < dis)
                {
                    dis = Distance(ptm[i], ptm[j]);
                    a = ptm[i];
                    b = ptm[j];
                }
            }
        }
    }
    else
    {//p 在左半部分，找随后7个点中在右半部分的点
        for (j = i + 1; j <= i + 7 && j < k; j++)
        {
            if (ptm[j].x - mid > 0)
            {
                if (Distance(ptm[i], ptm[j]) < dis)
                {
                    dis = Distance(ptm[i], ptm[j]);
                    a = ptm[i];
                    b = ptm[j];
                }
            }
        }
    }
}
return dis;
```

Part 4: 实验结果分析

表 1: results

算法	时间 (ms)
暴力搜索	223.962
分治算法	2.516

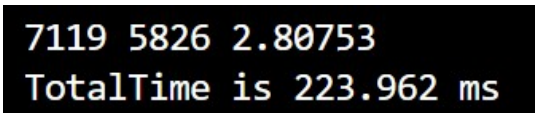

图 1: violent

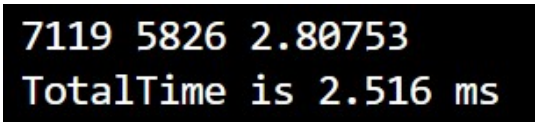

图 2: merge

通过以上对比，我们可以明显观察到分治算法带来的效率提高。提升了接近 100 倍。

Part 5 实验总结

本次实验完成了快速排序算法的实现及其优化。在实验过程中获得了以下收获:

- 感受到了分治算法相对于暴搜的效率提高
- 体会到了学习算法与实践的重要性