

Problem 1

用数学归纳法证明 *Dijkstra* 算法的正确性。

证明：

Dijkstra 算法：

DIJKSTRA.(G, w, s)

```

1  INITIALIZE – SINGLE – SOURCE( $G, s$ )
2   $S = \emptyset$ 
3   $Q = G.V$ 
4  while  $Q \neq \emptyset$ 
5       $u = EXTRACT – MIN(Q)$ 
6       $S = S \cup \{u\}$ 
7      for each vertex  $v \in G.Adj[u]$ 
8          RELAX( $u, v, w$ )

```

数学归纳法证明 *Dijkstra* 算法的正确性：

(1) 命题：

Dijkstra 算法运行在带权重的有向图 $G = (V, E)$ 时，如果所有权重为非负值，则在进行到第 k 步时，对于 $\forall u \in S$ ，有 $u.d = \delta(s, u)$ 。

(2) 归纳基础：

$k = 0$ 时， $S = \emptyset$ ，假设直接成立。

$k = 1$ 时， $S = \{s\}$ ， $s.d = \delta(s, s) = 0$ 。成立

(3) 归纳假设：

如果进行到第 k 步时，对于 $\forall u \in S$ ，有 $u.d = \delta(s, u)$ ，则进行 $k + 1$ 步时，对于所选取的 $v \in Q$ ，有 $v.d = \delta(s, v)$ 。

(4) 反证法证明：

假设进行到 $k + 1$ 步时， $v.d > \delta(s, v)$ ，即 $v.d \geq \delta(s, v)$ 。首先，因为 $s.d = \delta(s, s) = 0$ ，所以 $v \neq s$ ，因此将 v 加入 S 时， $S \neq \emptyset$ 。此时，一定存在从 s 到 v 的路径，那么也一定存在一个最短路径，记为 p^* 。令 u 为算法所找到的 $s \rightarrow v$ 路径中 v 的前驱。在 p^* 中，令最后一个 $\in S$ 的点为 x ，令第一个 $\in Q$ 的点为 y 。 $y \neq v$ ，因为如果 $y = v$ ，则 v 直接与 x 相连，则根据算法 *EXTRACT – MIN* 的操作， $v.d \leq y.d$ ，则 $v.d \leq \delta(s, v) = y.d + \delta(y, v)$ ，显然矛盾。同时，我们也可以根据上述分析得到 $y.d \geq v.d$ 。所以， $\delta(s, v) = y.d + \delta(y, v) \geq y.d \geq v.d$ 。这与我们的假设 $v.d > \delta(s, v)$ 矛盾，所以假设不成立，即 $v.d = \delta(s, v)$ 。

Problem 2

分治算法之最近点问题,结合示意图分别对何老师和郑欣同学描述的合并算法进行阐述,并尝试编程验证。

答:

基础论证: 因为如果 $\text{dist}(p, p') < \delta$, 则 p 和 p' 一定出现在 $2\delta \times \delta$ 的空间内, 所以一次只需要考虑一个 $2\delta \times \delta$ 。

a 最多需要找 6 个

为简化分析, 如果一个点落在一个 $\frac{\delta}{2} \times \frac{\delta}{2}$ 格子内, 则将其位置归约为格子的右下角顶点。

如果 $\text{dist}(p, p') < \delta$, 则两点一定分列在 *split* 分割线两边。

如果 $\text{dist}(p, p') < \delta$, 则另一点一定落在以该点为圆心, 以 δ 为半径的圆内。

当我们从上到下依次遍历每个点时, 我们只需要从该点向下寻找可能的点对使得其距离小于 δ 。因为如果 $y' > y$, 则在上一次遍历即会找出该点对。

综合以上三个限制, 假设 p 落在靠近 *split* 分割线的一个格子, 我们作出图 1。其中红色打勾的格子为另一点可能出现的地方。因此, 最多需要找 6 个格子。

我们可以发现, 不管以一个格子中的哪一个点为圆心, 其最终确定的需要找的格子都不会超过 6。当 p 落在远离 *split* 分割线的一个格子中时, 则根据上述分析范式, 只需要找 3 个。

b 最多需要找 3 个

我们考虑一个 $2\delta \times \delta$ 空间。对于每一个落在该空间内的点, 我们以该点为圆心, 以 $\frac{\delta}{2}$ 为半径作圆。如果 $\text{dist}(p, p') < \delta$, 则以两点为圆心的圆必定相交。因此, 我们考虑如何在一个 $2\delta \times \delta$ 空间内尽可能地塞入更多圆心, 亦即在一个 $3\delta \times 2\delta$ 空间内尽可能地多塞入圆, 如图 2 所示。

通过作图, 我们发现最多可以塞入 6 个圆, 又因为点出现在一个 $\frac{\delta}{2}$ 格子内或右下边界, 所以图 2 中左上角的点实际在 $2\delta \times \delta$ 空间外。那么接下来再来考虑寻找一个 $\text{dist}(p, p') < \delta$ 的点对。对于一个点 p , 我们同样以其所在格的右下顶点为圆心, 以 δ 为半径作圆, 如图 3 所示。

作图有以下限制:

如果 $\text{dist}(p, p') < \delta$, 则两点一定分列在 *split* 分割线两边。

如果 $\text{dist}(p, p') < \delta$, 则另一点一定落在以该点为圆心, 以 δ 为半径的圆内。

当我们从上到下依次遍历每个点时, 我们只需要从该点向下寻找可能的点对使得其距离小于 δ 。因为如果 $y' > y$, 则在上一次遍历即会找出该点对。

另外, 在 *split* 分割线另一边的点两两距离大于 δ , 即以 $\frac{\delta}{2}$ 为半径的圆不相交。

综合以上限制, 我绘制出了极限情况, 如图 3 所示。

从图 3 可以看出, 对于点 p , 在另一边至多找到 3 个点能够组成距离小于 δ 的点对, 即, 最多寻找 3 个点。

c 程序模拟

为了简化问题, 我们只模拟在空间 K 中寻找最小点对, 并且已知维数 $d = 2$, 已知最小距离 δ 。模拟思路是: 随机生成两个序列 s_1, s_2 , 分别代表 *split* 分割线两边距离 δ 以内的点。满



图 1: 最多找 6 个

足 $\forall p_1, p_2 \in s_i (i = 1, 2), \text{dist}(p_1, p_2) \geq \delta$; 将两个序列合并, 并按 y 排序; 从高到低依次遍历每个点, 在另一边试图找到一个点, 使他们之间的距离小于 δ 。统计对于每个点找到的最多的点的数量。根据 a, b 的分析, 每个点最多找到 3 个距离小于 δ 的点。

我使用 `c++` 来实现。实现代码放到了

https://github.com/Fir-lat/DivideConquer_ClosestPairs.git

上。模拟程序运行结果如图 4。可以更改 `P2` 对象中的常量来控制每次生成的点的特征。如果感兴趣, 可以移步上述链接。

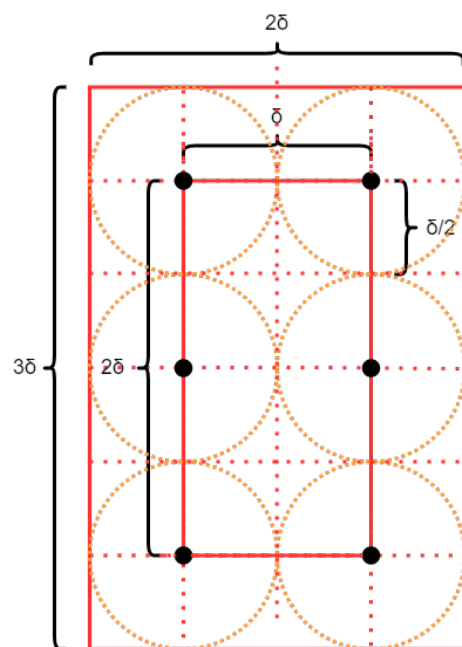


图 2: 最多找 3 个

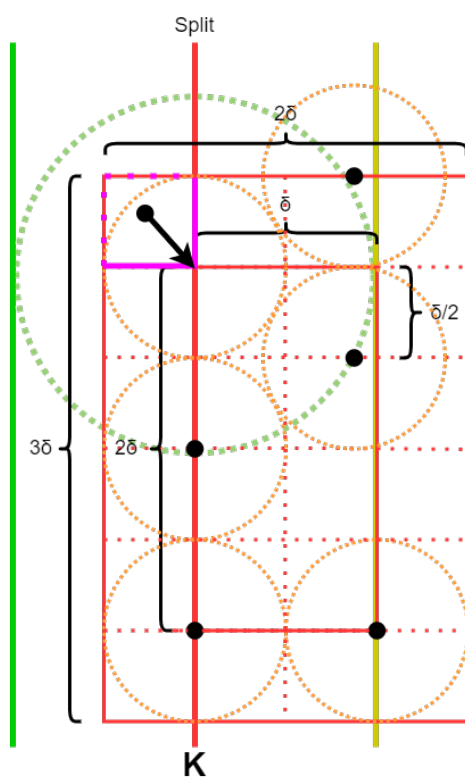
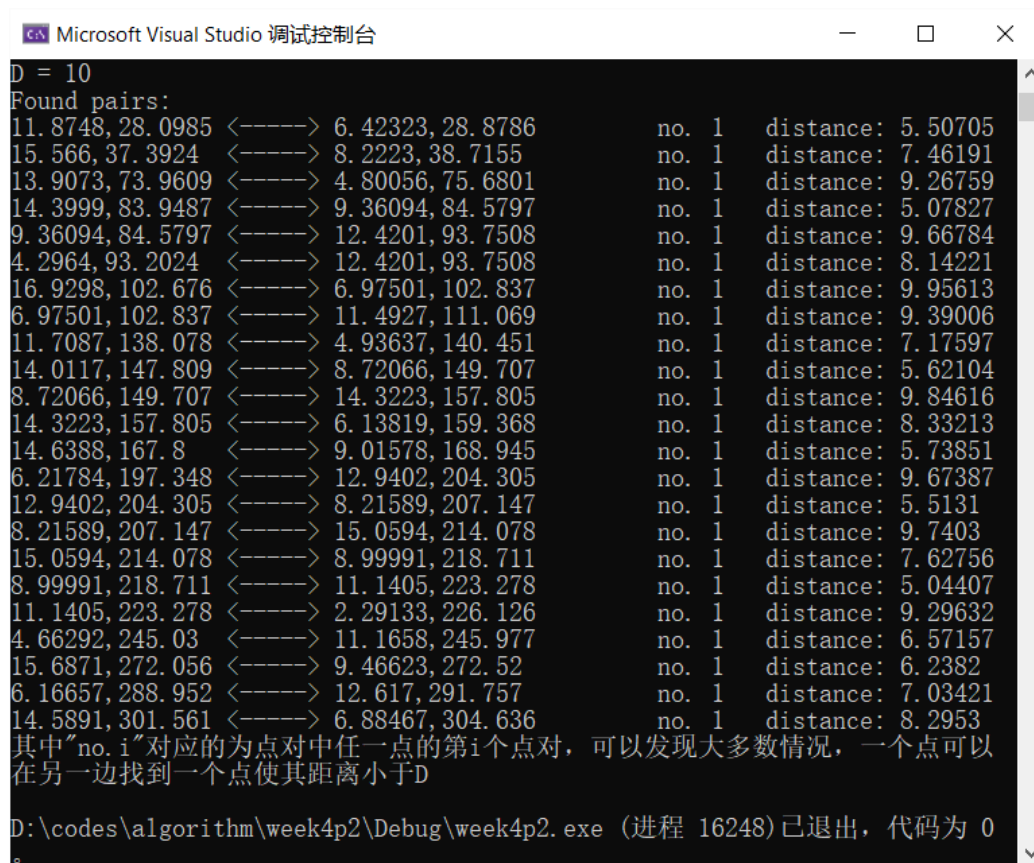


图 3: 最多找 3 个



```

Microsoft Visual Studio 调试控制台
D = 10
Found pairs:
11. 8748, 28. 0985 <-----> 6. 42323, 28. 8786      no. 1    distance: 5. 50705
15. 566, 37. 3924 <-----> 8. 2223, 38. 7155      no. 1    distance: 7. 46191
13. 9073, 73. 9609 <-----> 4. 80056, 75. 6801      no. 1    distance: 9. 26759
14. 3999, 83. 9487 <-----> 9. 36094, 84. 5797      no. 1    distance: 5. 07827
9. 36094, 84. 5797 <-----> 12. 4201, 93. 7508      no. 1    distance: 9. 66784
4. 2964, 93. 2024 <-----> 12. 4201, 93. 7508      no. 1    distance: 8. 14221
16. 9298, 102. 676 <-----> 6. 97501, 102. 837      no. 1    distance: 9. 95613
6. 97501, 102. 837 <-----> 11. 4927, 111. 069      no. 1    distance: 9. 39006
11. 7087, 138. 078 <-----> 4. 93637, 140. 451      no. 1    distance: 7. 17597
14. 0117, 147. 809 <-----> 8. 72066, 149. 707      no. 1    distance: 5. 62104
8. 72066, 149. 707 <-----> 14. 3223, 157. 805      no. 1    distance: 9. 84616
14. 3223, 157. 805 <-----> 6. 13819, 159. 368      no. 1    distance: 8. 33213
14. 6388, 167. 8 <-----> 9. 01578, 168. 945      no. 1    distance: 5. 73851
6. 21784, 197. 348 <-----> 12. 9402, 204. 305      no. 1    distance: 9. 67387
12. 9402, 204. 305 <-----> 8. 21589, 207. 147      no. 1    distance: 5. 5131
8. 21589, 207. 147 <-----> 15. 0594, 214. 078      no. 1    distance: 9. 7403
15. 0594, 214. 078 <-----> 8. 99991, 218. 711      no. 1    distance: 7. 62756
8. 99991, 218. 711 <-----> 11. 1405, 223. 278      no. 1    distance: 5. 04407
11. 1405, 223. 278 <-----> 2. 29133, 226. 126      no. 1    distance: 9. 29632
4. 66292, 245. 03 <-----> 11. 1658, 245. 977      no. 1    distance: 6. 57157
15. 6871, 272. 056 <-----> 9. 46623, 272. 52      no. 1    distance: 6. 2382
6. 16657, 288. 952 <-----> 12. 617, 291. 757      no. 1    distance: 7. 03421
14. 5891, 301. 561 <-----> 6. 88467, 304. 636      no. 1    distance: 8. 2953
其中"no. i"对应的为点对中任一点的第i个点对，可以发现大多数情况，一个点可以在另一边找到一个点使其距离小于D
D:\codes\algorithm\week4p2\Debug\week4p2.exe (进程 16248) 已退出，代码为 0

```

图 4: 程序模拟