NOI2017 《分身术》试题讨论

命题人　余行江 彭雨翔

清华大学 交叉信息研究院

2017 年 7 月 21 日

题目大意

给出平面上的 n 个点，以及 m 个如下格式的询问：

- 从这 n 个点中选定 k 个点删去。剩下 $n-k$ 个点所构成的凸包面积是多少？

需要给出在线算法。

数据范围及条件

- $n, m \leq 10^5$
- k 为常数级 $(k \leq 100)$
- n 个点构成的凸壳上总存在两个没有被删除的点

得分情况

- 70 分：2 人
- 50 60 分：3 人
- 25 40 分：24 人
- 5 20 分：245 人
- 0 分：163 人

关于动态凸包问题的复杂度

用可持久化平衡树可以实现完全动态凸包 [1]。

- 时间复杂度 $O(n \log n) - O(\log^2 n)$。

[1] 陈立杰 [2], 2013 IOI 集训队论文, 2013

关于动态凸包问题的复杂度

Type	Year	Insertion	Deletion	Query
Online	Preparata **1979**	$O(\log n)$	-	$O(\log n)$
	Overmars and Leeuwen **1981**	$O(\log^2 n)$	$O(\log^2 n)$	$O(\log n)$
	Hershberger and Suri **1992**	-	$O_A(\log n)$	$O(\log n)$
Offline	Hershberger and Suri **1996**	$O_A(\log n)$	$O_A(\log n)$	$O(\log n)$
Online	Brodal and Jacob **2002**	$O_A(\log n)$	$O_A(\log n)$	$O(\log n)$

- O_A = Amortized (均摊)

关于离线

(Acknowledgement: 吉如一，于纪平)

利用分治也可以离线地实现动态凸包。

- 插入-删除：等价于一个点在某个时间段上出现。
- 按时间分治，每个时间段按分治结构拆成 $O(\log K)$ 个恰好完全覆盖一个分治节点的区间。
- 按 dfs 序遍历结构，进入每个节点时加入所有相关区间上的点，在叶子上处理询问。

时间复杂度 $O_A(K \log K \log n) - O(\log n)$。

- $K = \sum k$

场外

(Acknowledgement：ImmortalCO)

对于每个询问，考虑上凸壳。

- 按 x 轴顺序排序删去点。
- 对于相邻的两个删去点，求出其之间部分的上凸壳。
  - 等价于询问以原 n 个点为准，某区间 $[l, r]$ 内的凸壳。
  - 预处理线段树结构后可以 $O(1)$ 找到 $[l, r]$ 第一次被分割的节点 [3]。
  - 用可持久化平衡树预处理 *mid* 前后缀的凸壳后，$O(\log n)$ 内可合并。
- 再对 $O(k)$ 个凸壳合并即可。

时间复杂度 $O(n \log^2 n) - O(k \log n)$。

[3]可参考 http://immortalco.blog.uoj.ac/blog/2102

一些部分分算法

Case $k = 1$

只有删去最外层凸壳上的某个点才会使凸包改变。对所有这样的点预处理答案。删去凸壳上的一个点后，找到其在凸壳上两边的点 a, b，然后暴力重构 $[X_a, X_b]$ 这一段的新凸壳。注意这么做的复杂度是 $O_A(n)$，即均摊线性。

可以稍加讨论地推广到 k 等于 2 （可能更高 *）的情况。

参考算法 -Case $k = 1$

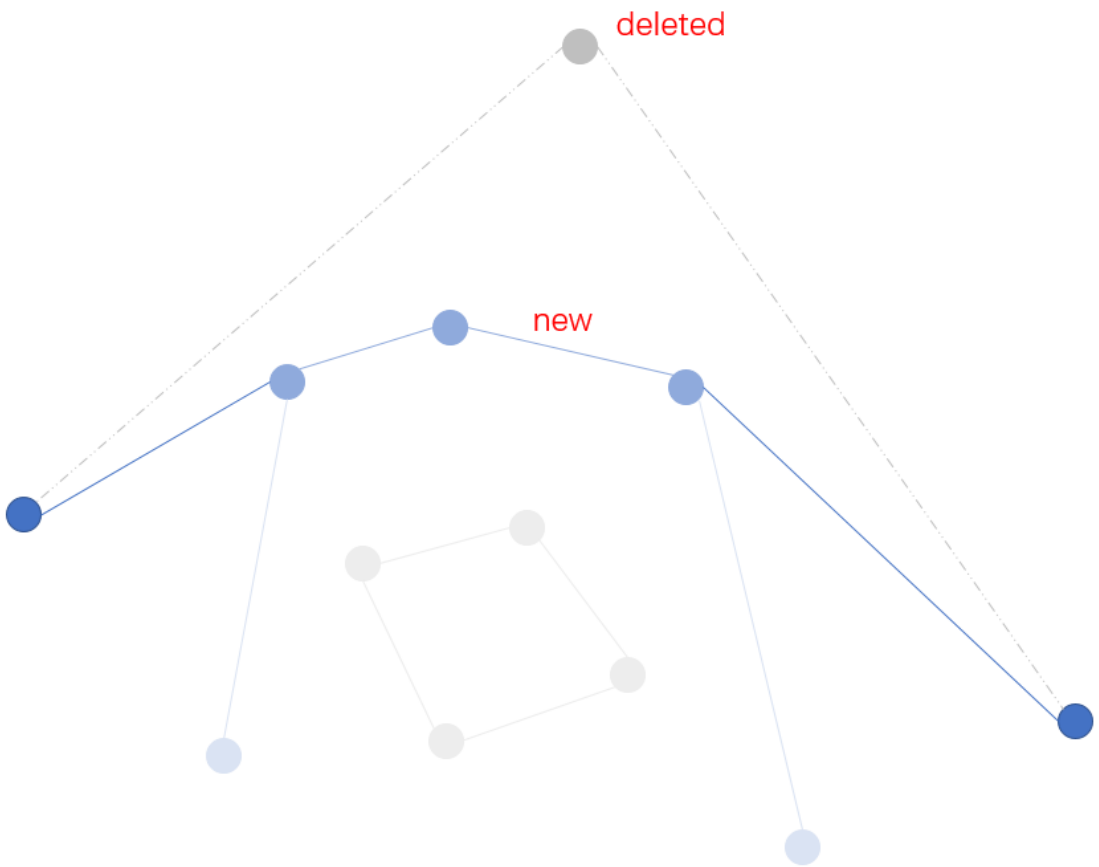
当只删去 1 个点时

- 删去点不在凸壳上：凸包不变
- 删去点在凸壳上：
  - 原凸壳中除删去点以外的所有点依然在删去该点之后的凸壳上
  - 对于删去该点的部分，凸壳会向第二层凸包收缩

第二层凸包

删去第一层凸包上的所有点后，剩下的点所构成的凸包

参考算法 -Case $k = 1$ (cont.)

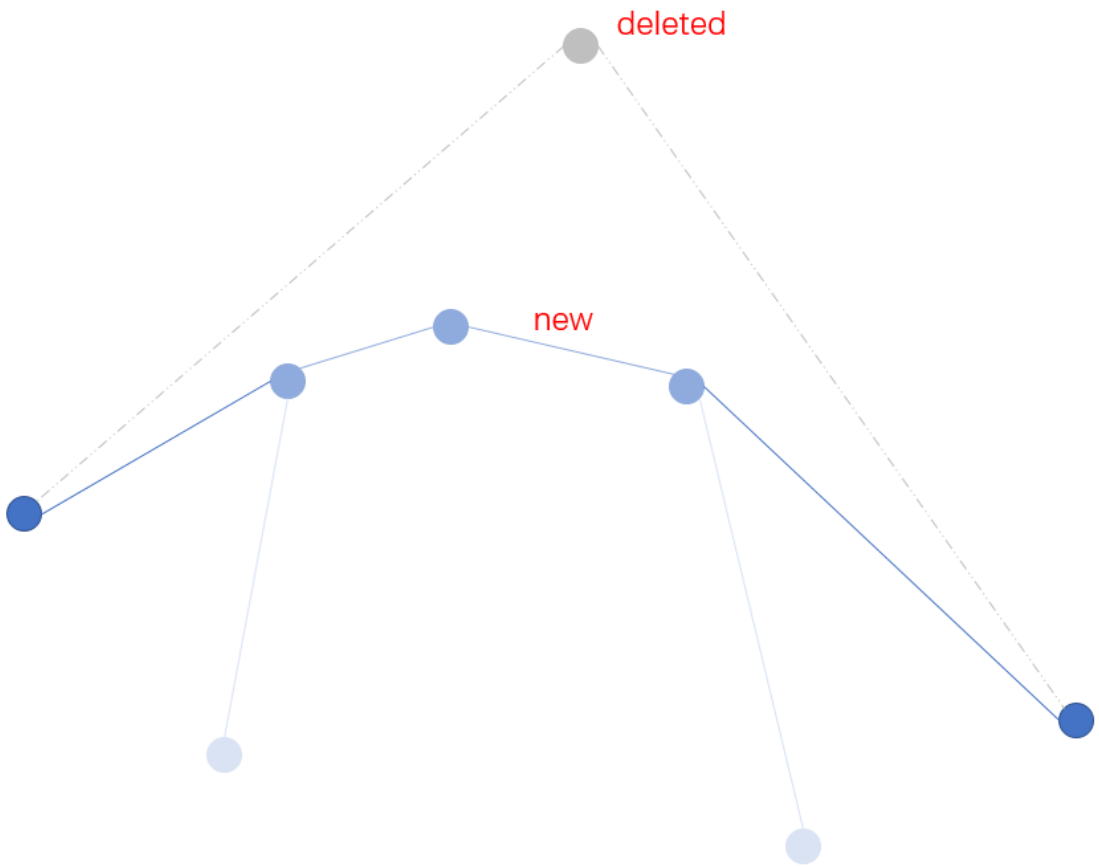


参考算法 -Case $k = 1$ (cont.)

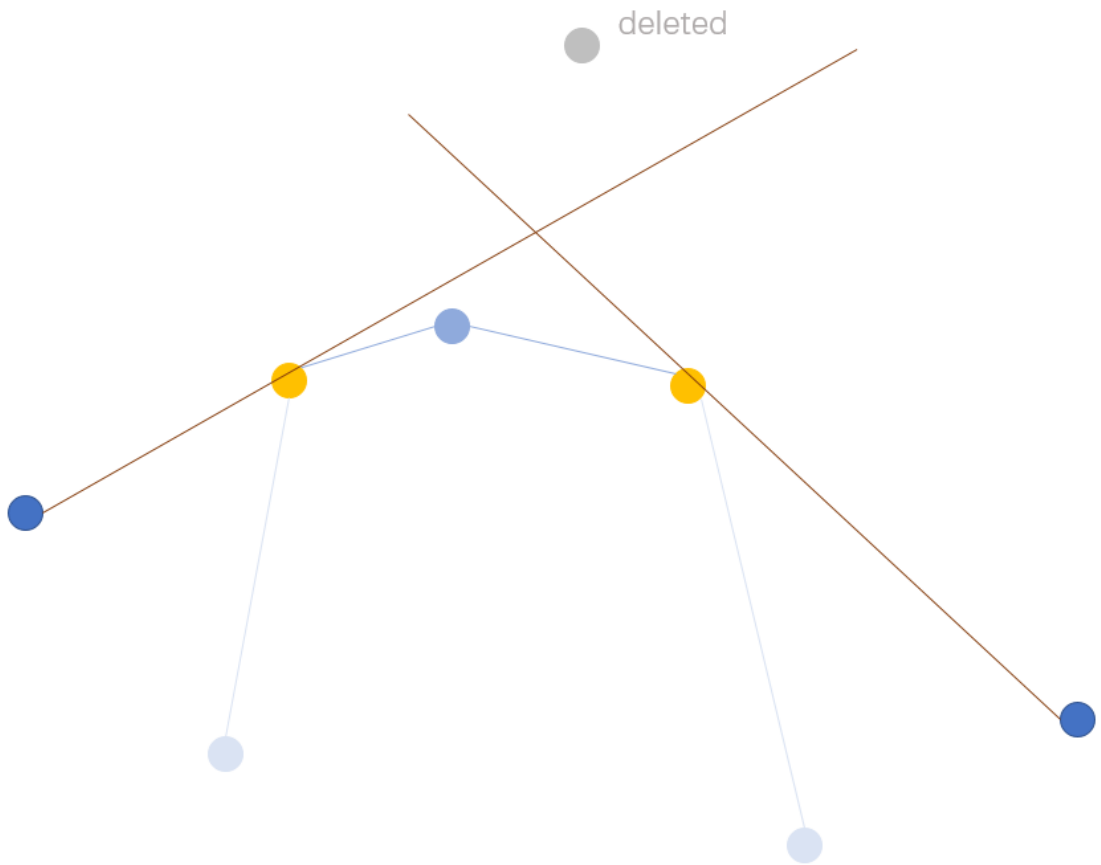
凸壳收缩的方式是以两端固定的两个点，向第二层凸包做切线。

- 所得到的新凸壳部分，即为第二层凸壳上，两切线所在切点按对应时针方向 所对应的部分。

参考算法 -Case $k = 1$ (cont.)



参考算法 -Case $k = 1$ (cont.)



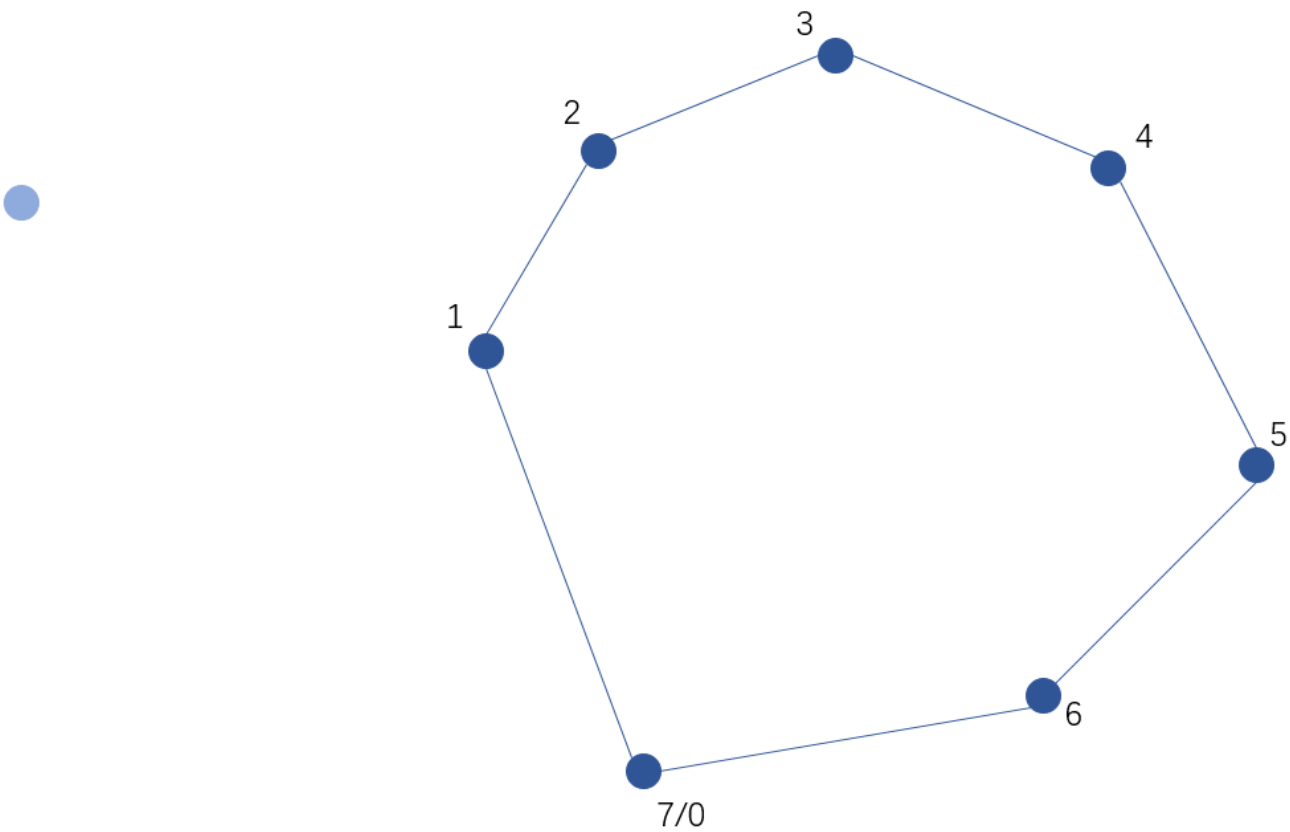
参考算法 -Case $k = 1$ (cont.)

可以用二分法来寻找切线。

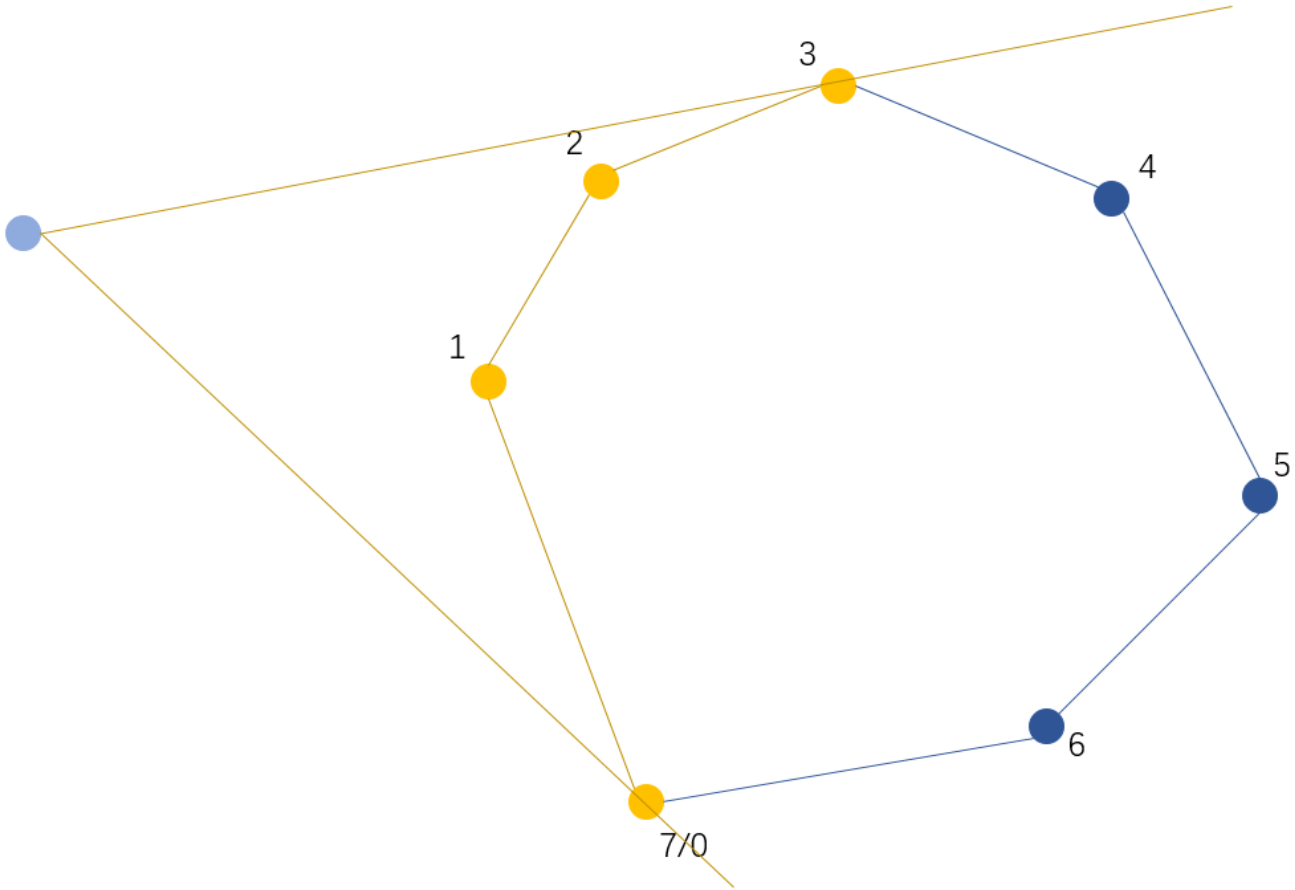
参考方法

以按顺时针方向存储凸包，找顺时针方向切线为例。

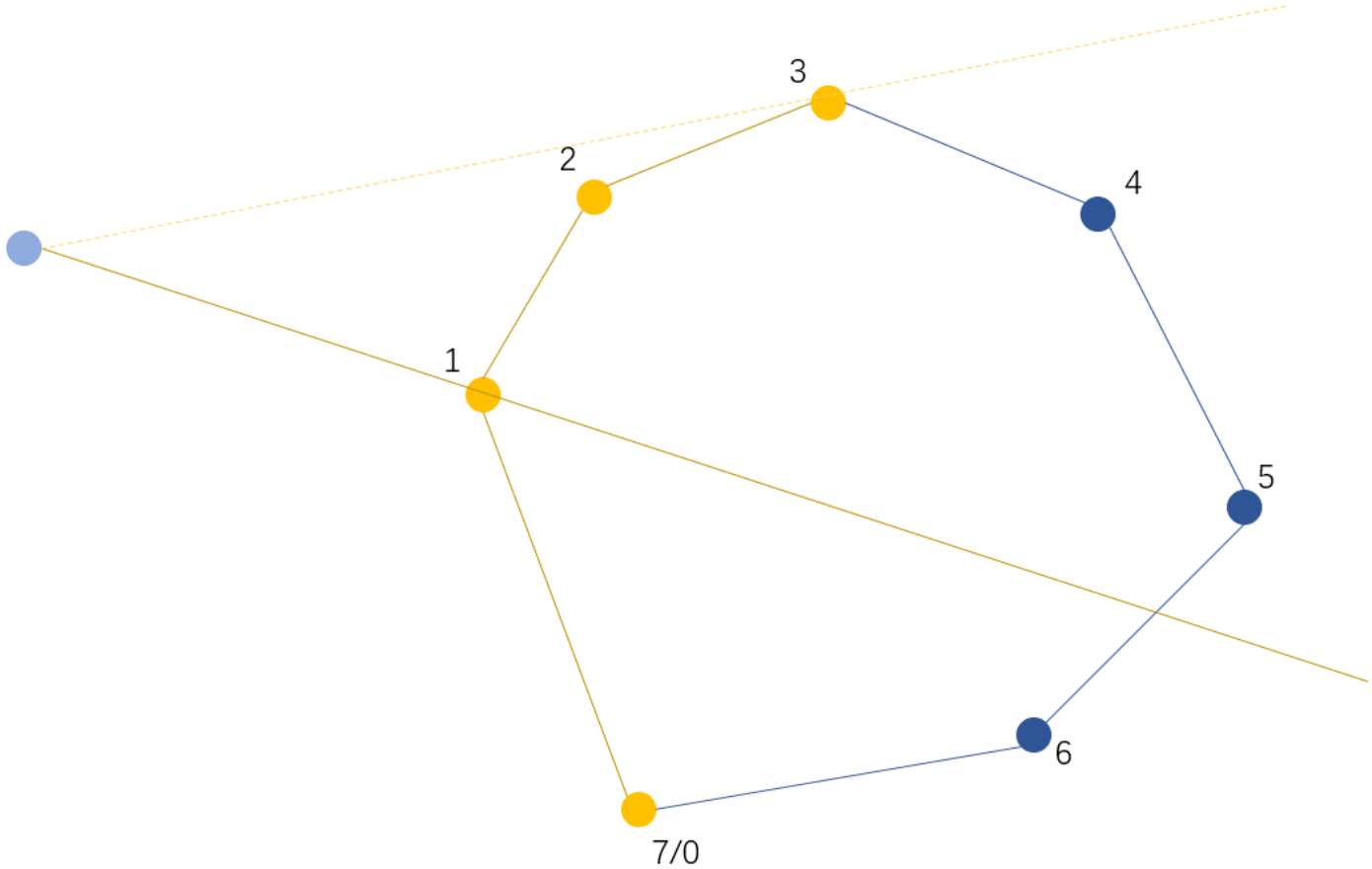
参考算法 -Case $k = 1$ (cont.)



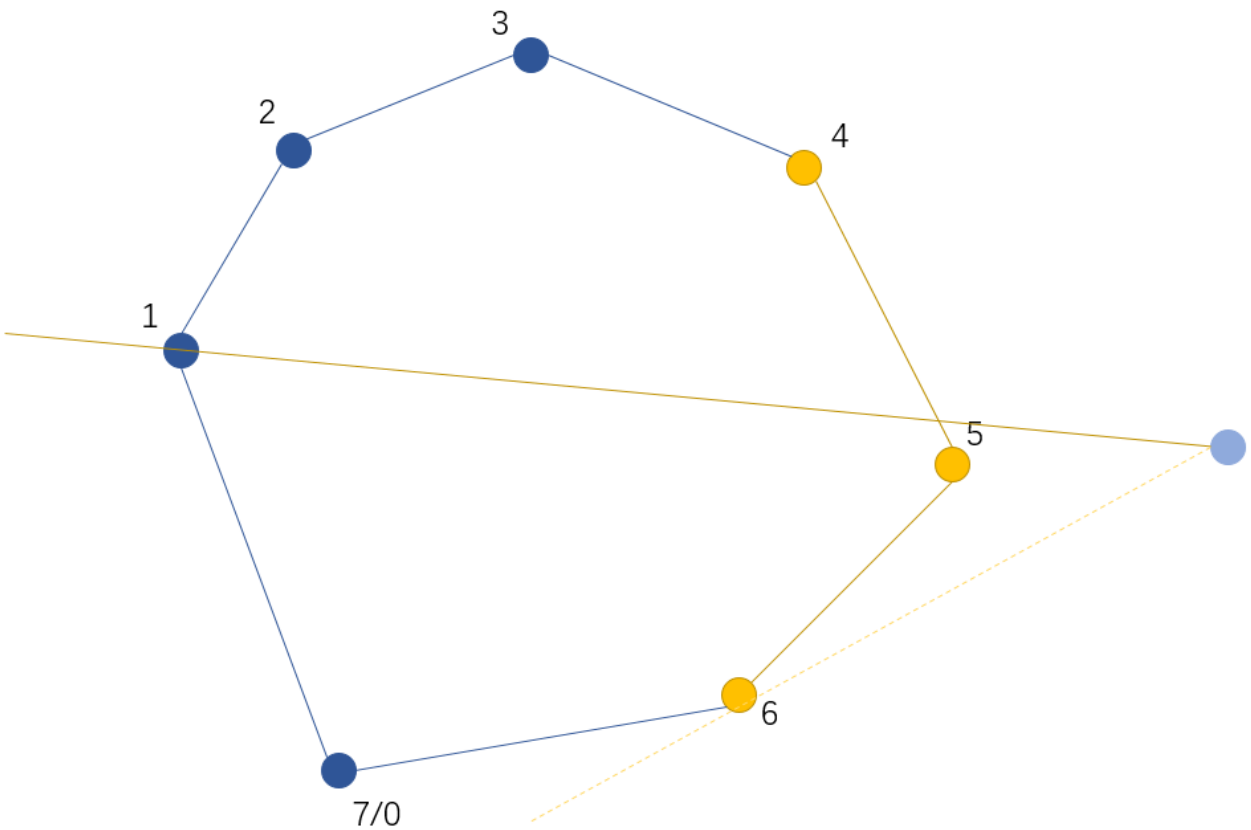
参考算法 -Case $k = 1$ (cont.)



参考算法 -Case $k = 1$ (cont.)



参考算法 -Case $k = 1$ (cont.)



伪代码：查找点到凸包的切线（相同时针序）

```
find_tangle_clockwise(a, CH):
1   point_contact = CH_1
2   if CH_1 is visible in terms of aCH_0 then
3   |   binary search in (1, n]
4   |       if CH_mid is not visible in terms of aCH_0 then
5   |   |       search (1, mid]
    |       end
6   |       else if CH_mid is visible in terms of aCH_{mid-1} then
7   |   |       point_contact = CH_mid
8   |   |       search (mid, n]
    |       end
9   |       else
10  |   |       search (1, mid]
    |       end
    end
```

伪代码：查找点到凸包的切线（相同时针序）(cont.)

```
1  if CH_1 is not visible in terms of vec(aCH_0) then
2      binary search in (1, n]
3          if CH_mid is not visible in terms of vec(aCH_0) then
4              search (mid, n]
           end
5          else if CH_mid is visible in terms of vec(aCH_{mid-1}) then
6              point_contact = CH_mid
7              search (mid, n]
           end
8          else
9              search (1, mid]
           end
   end
```

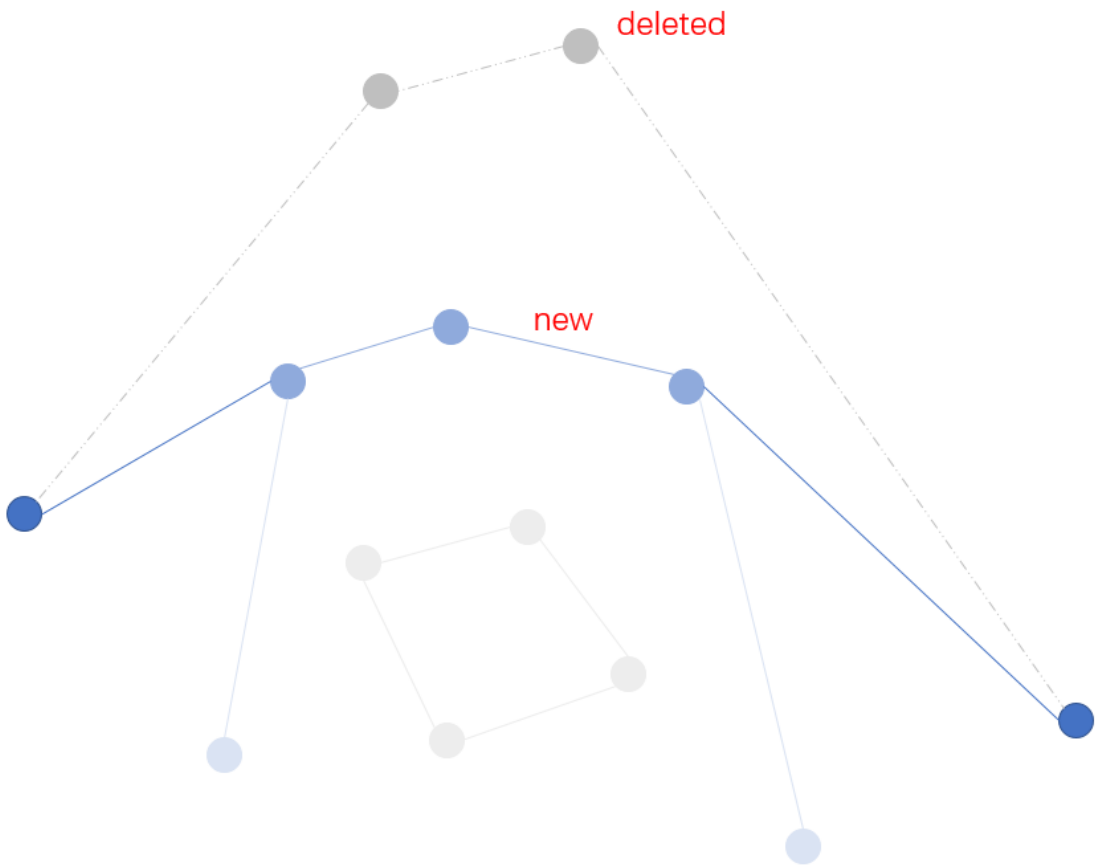
伪代码：查找点到凸包的切线（相同时针序）(cont.)

```
1  if CH_1 is not visible in terms of vec(aCH_0) then
2      binary search in (1, n]
3          if CH_mid is not visible in terms of vec(aCH_0) then
4              search (mid, n]
           end
5          else if CH_mid is visible in terms of vec(aCH_{mid-1}) then
6              point_contact = CH_mid
7              search (mid, n]
           end
8          else
9              search (1, mid]
           end
   end
```

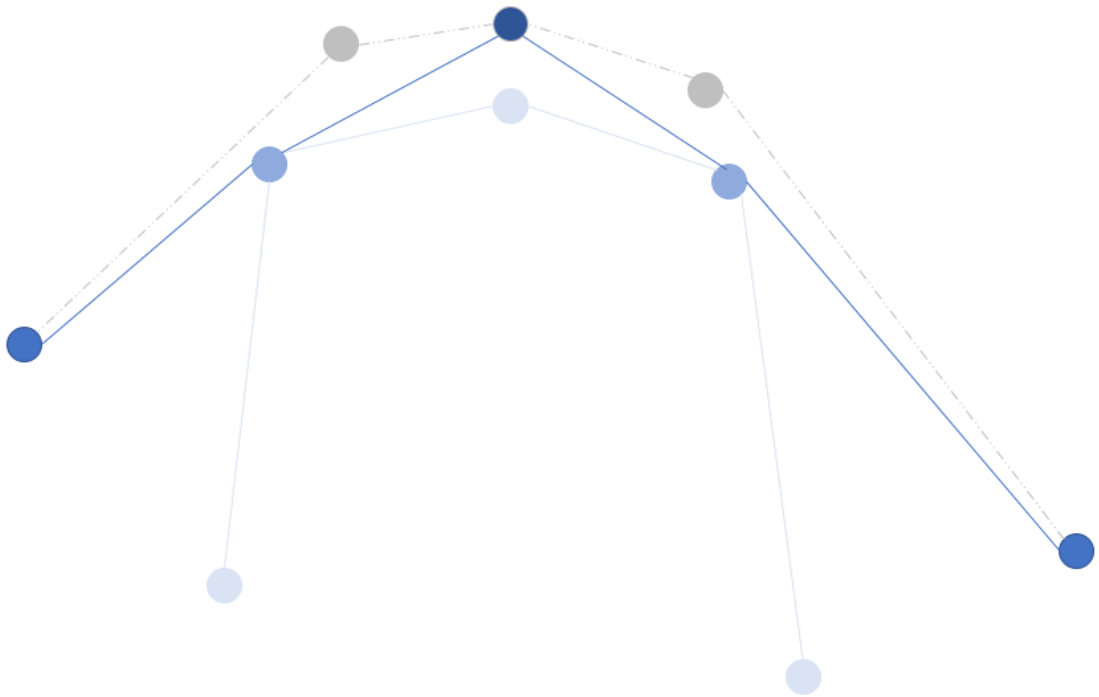
若删掉的两个点都在第一层凸壳上，类似 $k = 1$ 的情况。

- 若两个点不相邻，分别处理
- 若两个点相邻，将其视为一个整体

参考算法 -Case $k = 2$ (cont.)



参考算法 -Case $k = 2$ (cont.)

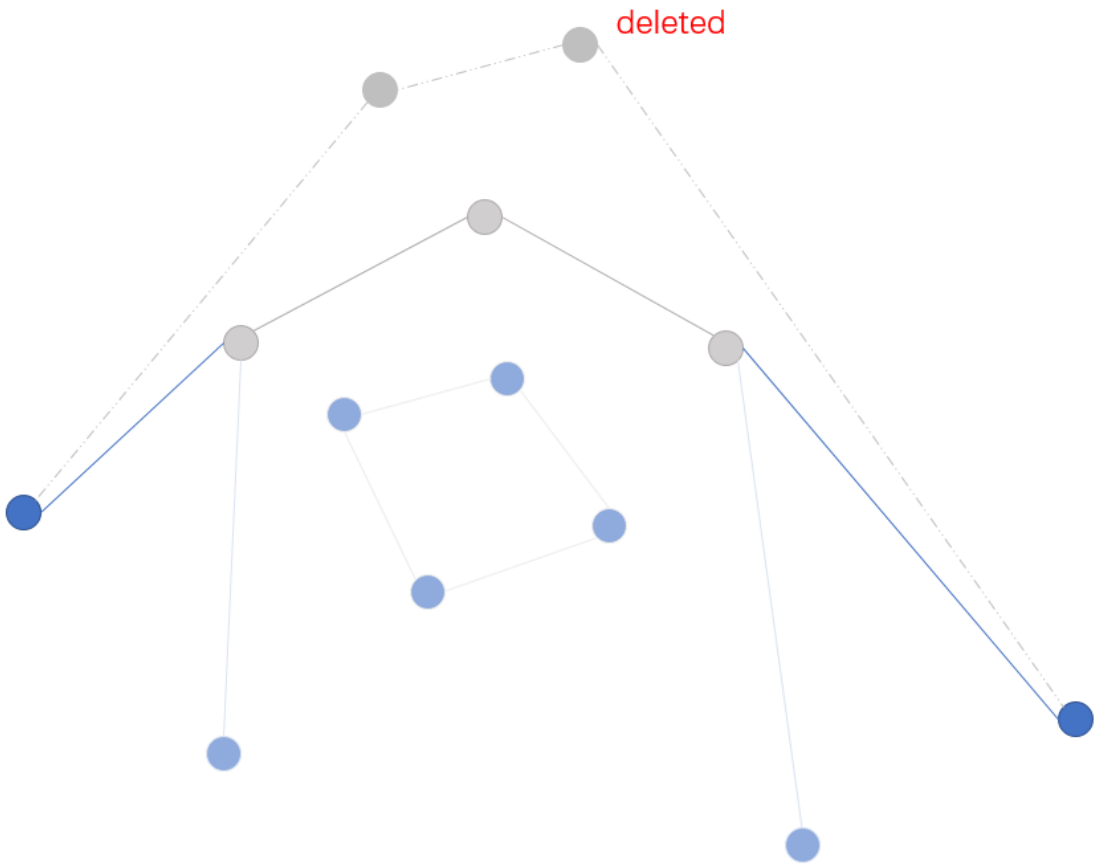


参考算法 -Case $k = 2$ (cont.)

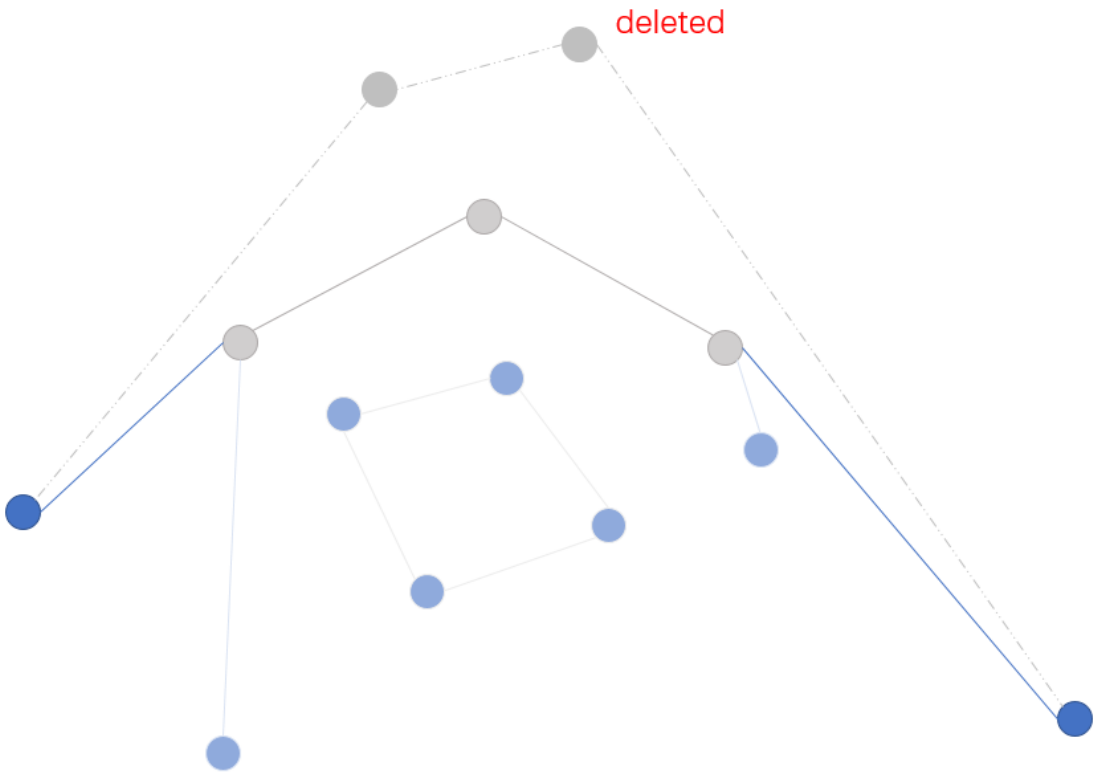
一个点在第一层凸壳上，另一个点在第二层凸壳上

- 先删掉在第一层凸壳上的点
- 若第二层凸壳上的点在删掉第一个点后新得到的凸壳上
  - 情况 1：向第三层凸包收缩
  - 情况 2：被第二层凸壳上，与删去点相邻的两个点卡住

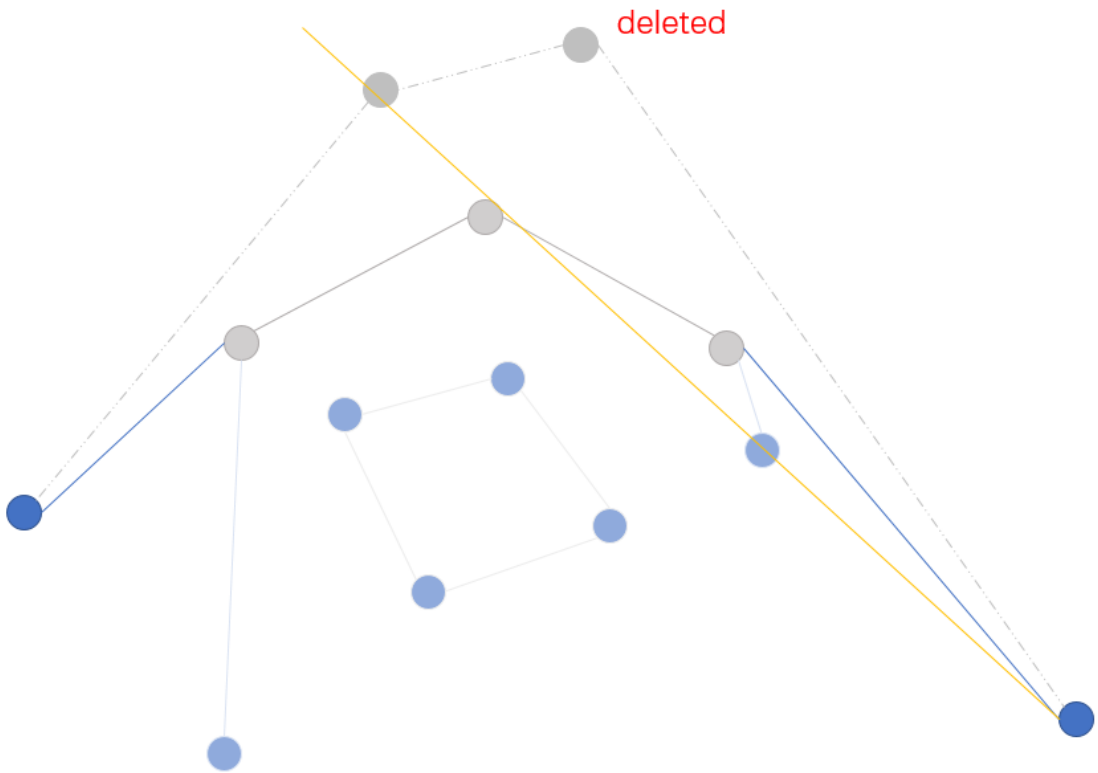
参考算法 -Case $k = 2$ (cont.)



参考算法 -Case $k = 2$ (cont.)



参考算法 -Case $k = 2$ (cont.)

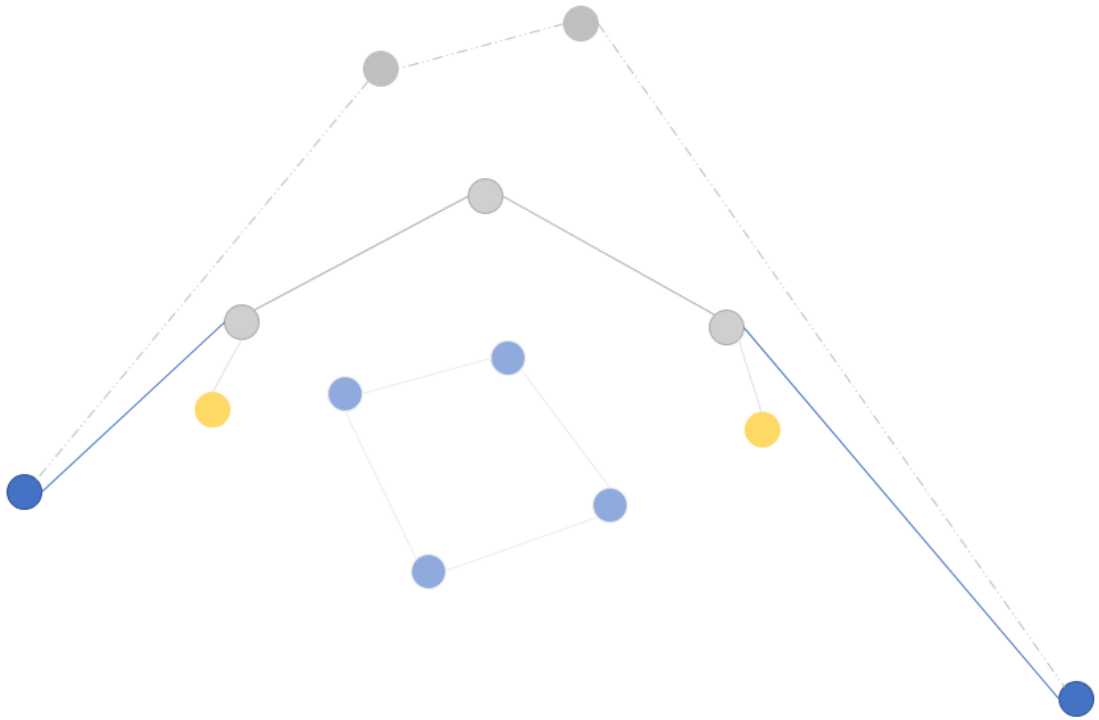


参考算法 -Case $k = 3$

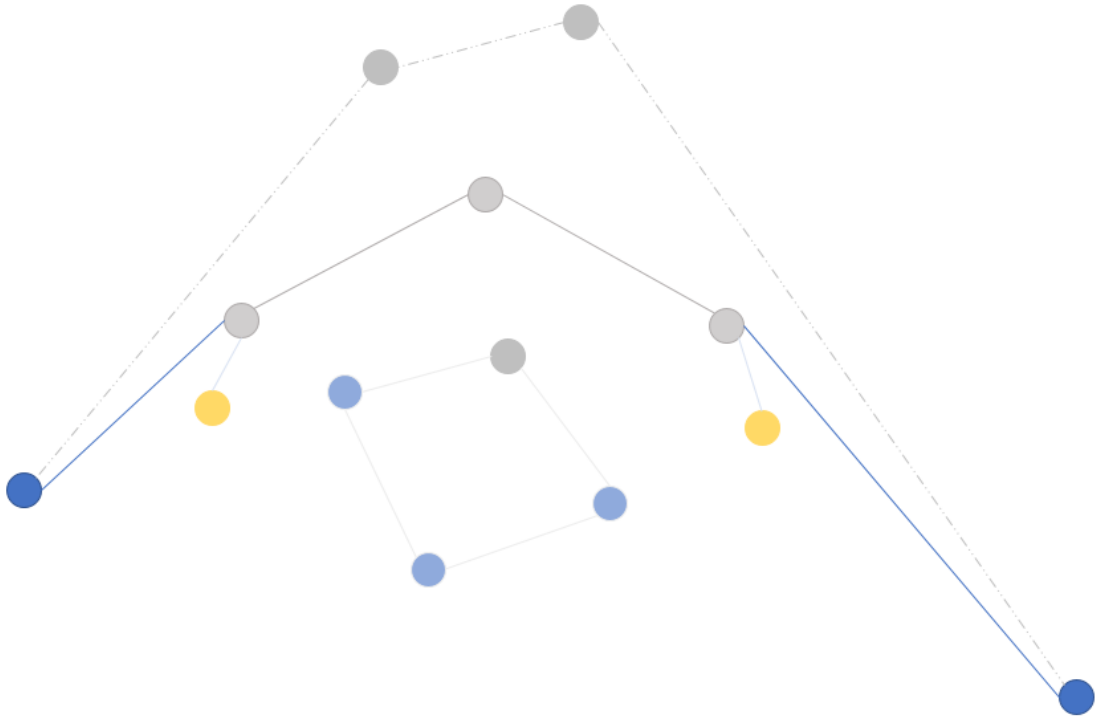
类似 $k = 2$，当某个在第三层凸壳上的点被删去之后

- 情况 1：向第三层凸包收缩
- 情况 2：被第三层凸壳上，与删去点相邻的两个点卡住
- 情况 3：被第二层凸壳上，与对应删去点相邻的两个点卡住

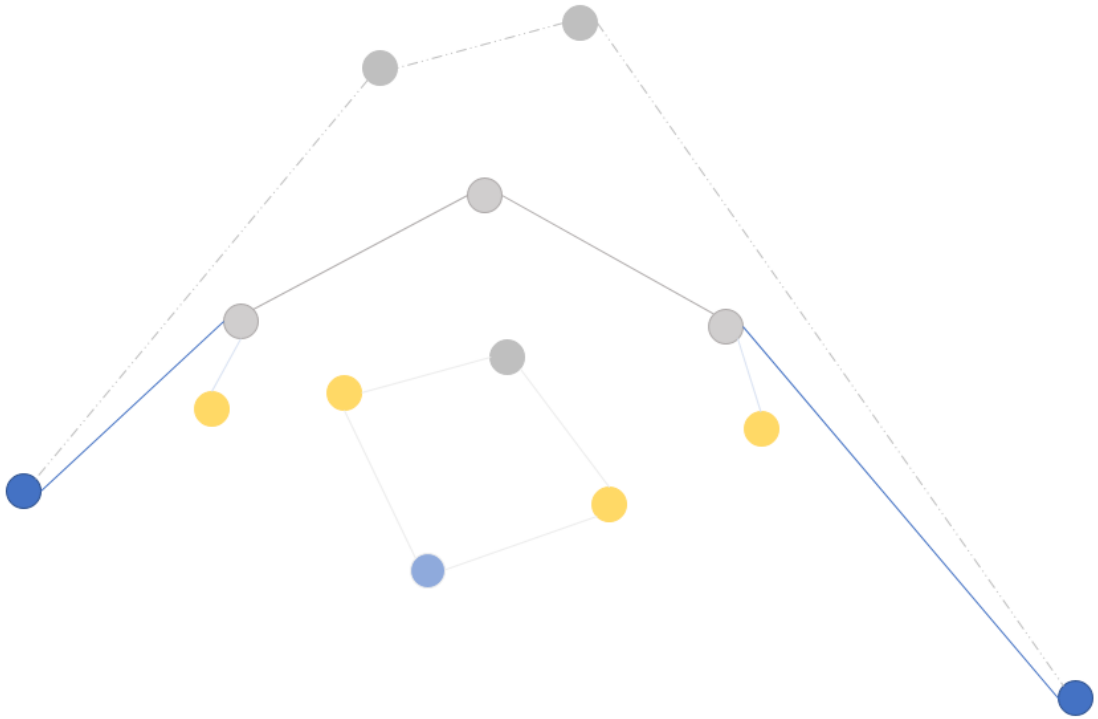
参考算法 -Case $k = 3$ (cont.)



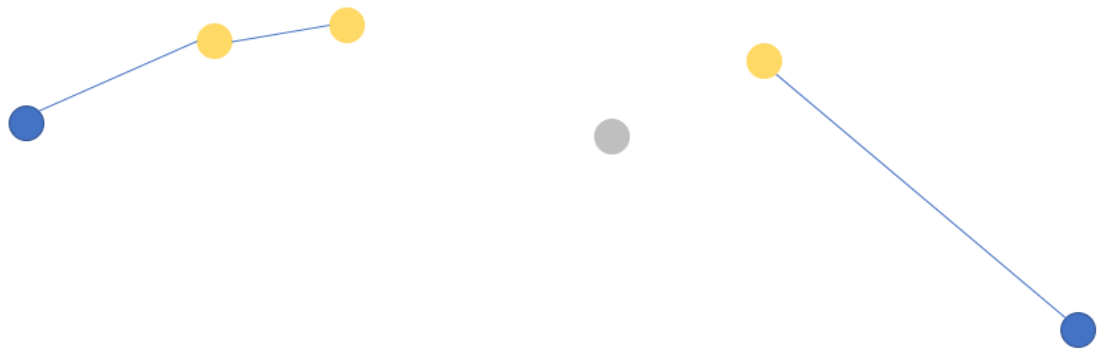
参考算法 -Case $k = 3$ (cont.)



参考算法 -Case $k = 3$ (cont.)



参考算法 -Case $k = 3$ (cont.)



参考算法 -Case $k = O(1)$

将删去点按所在凸壳层排序，并按此顺序依次删除。
对于每一个正在收缩的部分，除了维护当前得到新凸壳，还维护：

- 以第一层凸壳的两个固定点开始的，所有备选点构成的凸包。

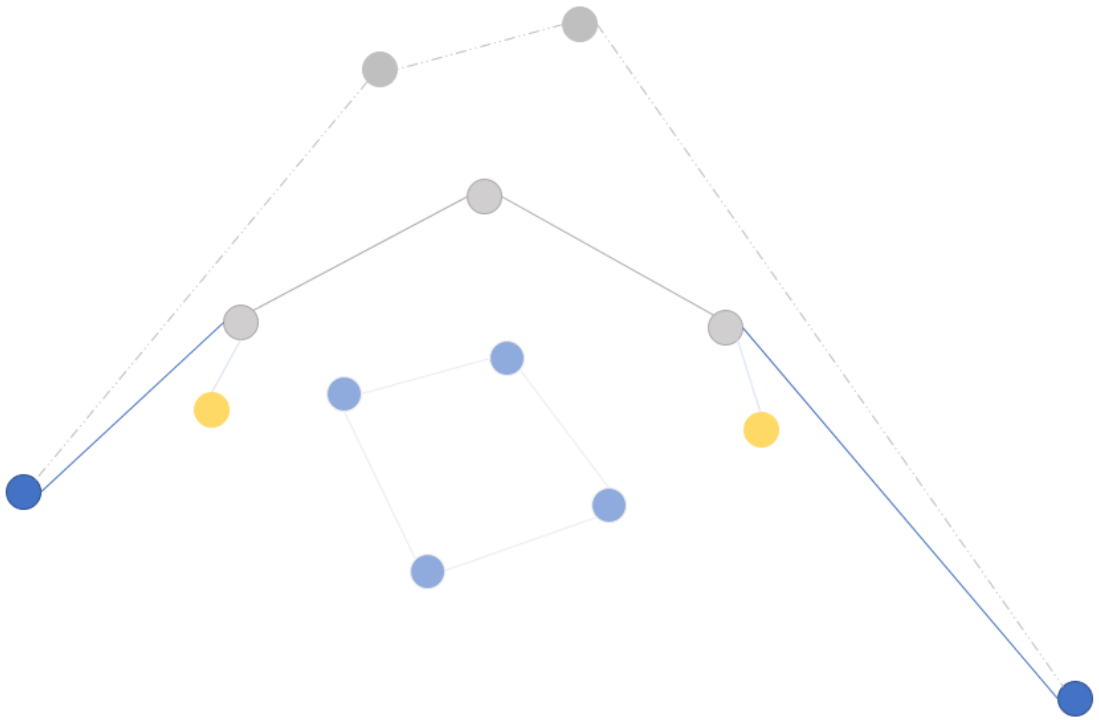
备选点

每一层凸壳上对应删去连续段的两端（即没有被删去的两个点）。

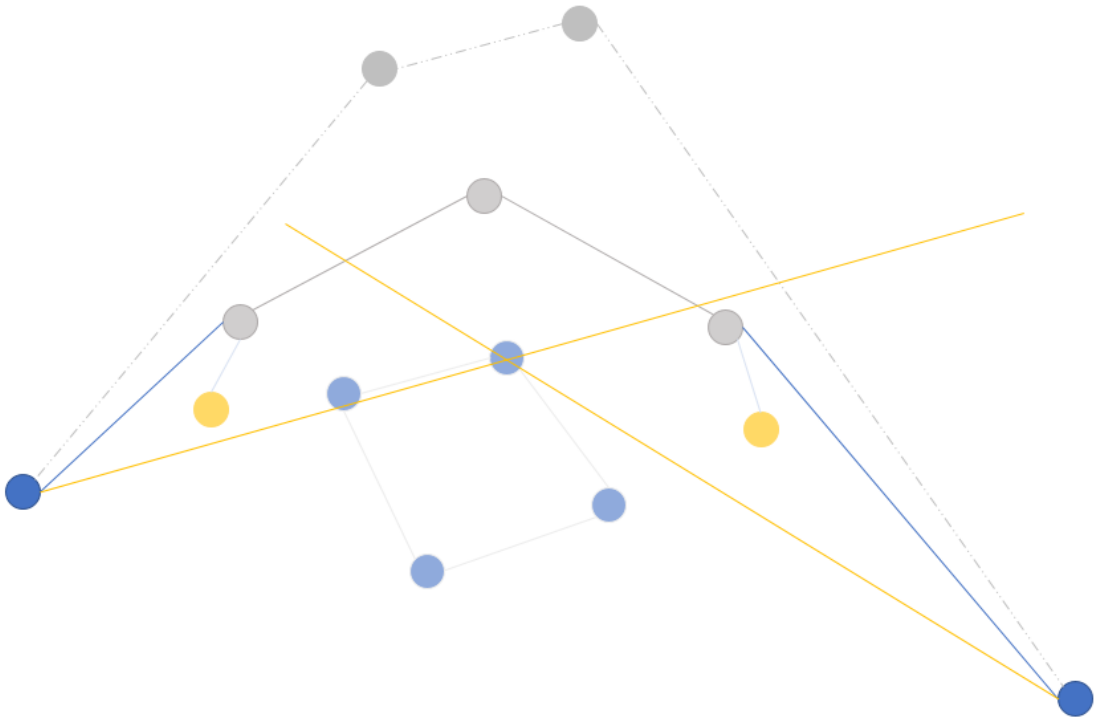
Lemma.

每一层只有这两个备选点可能在之后被重新采用。一个备选点被采用的条件是：
处在备选点构成凸包的两端且不被当前二分凸壳层所得到的点（见后例）遮挡。

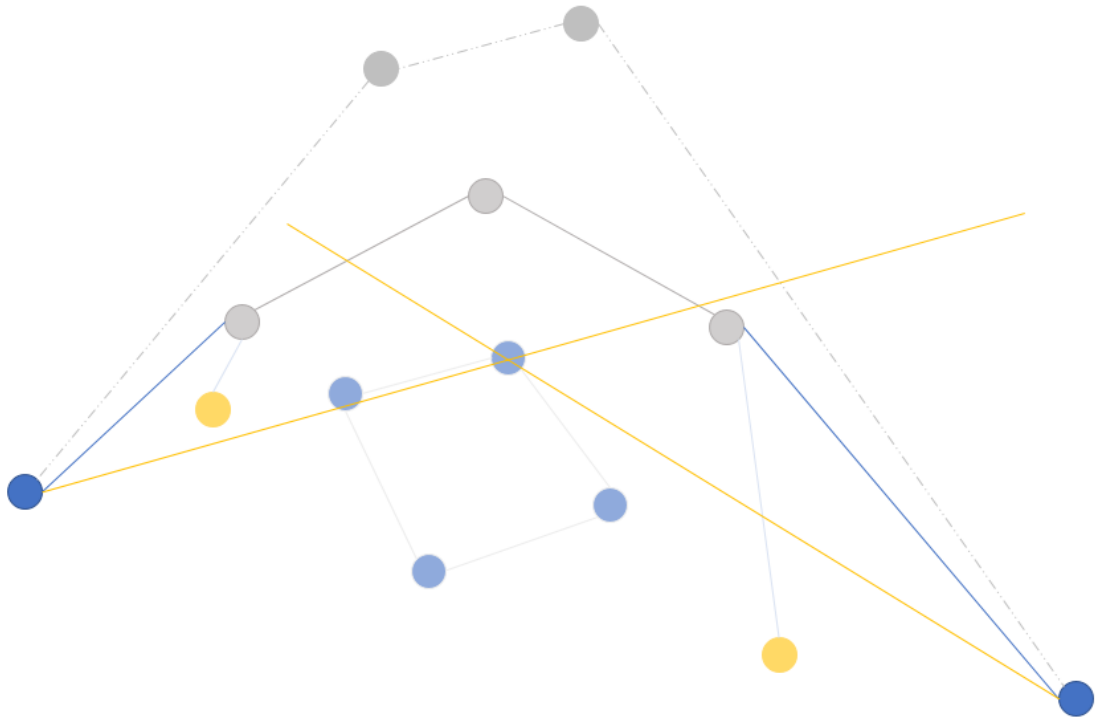
参考算法 -Case $k = O(1)$ (cont.)



参考算法 -Case $k = O(1)$ (cont.)



参考算法 -Case $k = O(1)$ (cont.)



参考算法 -Case $k = O(1)$

- 一个正在收缩的部分可能会分裂成若干个新的收缩部分，但最多不会超过 k 个。
- 将删去点按所在层排序分组后，可以在 $O_A(1)$ 时间内推导下一层的收缩部分。
- 最多只有 k 层。

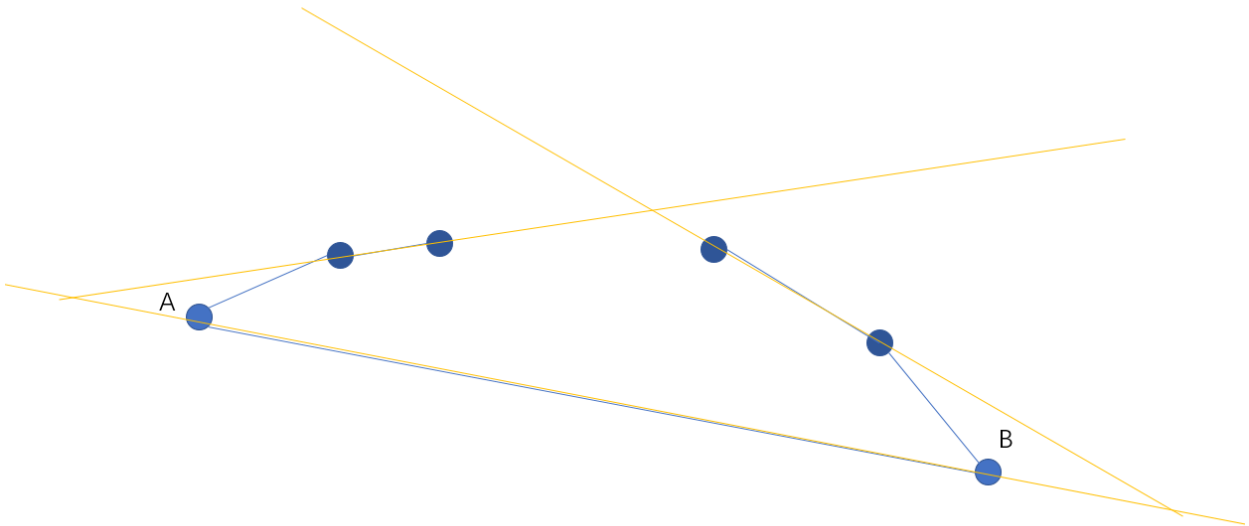
时间复杂度 $O(n \max k) - O(k \log n)$。

参考算法 -编程时的一些简化

- 本题限制：第一层凸壳上一定留有两个点没有被删去。
- 不需要考虑一个收缩部分的两端相同的情况。

参考算法 -编程时的一些简化

- 维护备选点凸包，每次新加入两个点时，可以证明两个点不可能处在下图中的特定区域（标识 A, B）。
- 可以将判断过程简化到 $O(1)$。



- 鸣谢吉如一对于本题的验题相关工作。
- 鸣谢张瑞喆对于本题题面修订和部分分设计的相关工作。