**Algorithm 1:** FindShortestPathBetweenTwoPoints(A, B)

**input** ：A(x, y)，B(x, y)，障碍物集合

**output** 构成最短路径的路径点列表 pointList

**:**

注：此算法仅考虑二维平面；障碍物均为矩形；路径点列表 *pointList* 记录了最短路径上的必经点，但 *pointList* 不包括真实起点 $A$，在函数外手动添加;

初始化路径点列表 pointList、pointListUP、pointListDOWN;

**if** 线段 $AB$ 之间没有障碍物 **then**

  $A \rightarrow B$ 为最短路径，pointList.add(B);

  return pointList;

**else**

  计算：线段 AB 依次穿过障碍物 $O_1, O_2, O_3 ... O_n$;

  **for** $i \leftarrow 1$ **to** $n$ **do**

    计算：点 A 对障碍物 $O_i$ 的上边缘可见点 $P_{Ai}^{up}$;

    **if** 直线 $AP_{Ai}^{up}$ 与 $O_{i+1}, O_{i+2} ... O_n$ 均不相交 **then**

      可知：$P_{Ai}^{up}$ 在 A 到 B 的最短路径上，pointListUP.add($P_{Ai}^{up}$);

      计算：点 B 对障碍物 $O_i$ 的上边缘可见点 $P_{Bi}^{up}$;

      **if** $P_{Bi}^{up}$ 与 $P_{Ai}^{up}$ 是同一个点 **then**

        **if** 线段 $P_{Bi}^{up}B$ 不穿过障碍物 **then**

          $A \rightarrow P_{Bi}^{up} \rightarrow B$ 为最短路径，pointListUP.add(B);

        **else**

          以 $P_{Ai}^{up}$ 为起点，找 $P_{Ai}^{up}$ 到 B 的最短路径，

          pointListUP.add(FindShortestPathBetweenTwoPoints($P_{Ai}^{up}$, B));

      **else**

        **if** 直线 $P_{Bi}^{up}P_{Ai}^{up}$ 穿过障碍物 $O_{i+1}, O_{i+2} ... O_n$ 其中一个或多个 **then**

          可知：$P_{Bi}^{up}$ 不在最短路径上;

          以 $P_{Ai}^{up}$ 为起点，找 $P_{Ai}^{up}$ 到 B 的最短路径，

          pointListUP.add(FindShortestPathBetweenTwoPoints($P_{Ai}^{up}$, B));

        **else**

          可知：$P_{Bi}^{up}$ 在最短路径上，pointListUP.add($P_{Bi}^{up}$);

          以 $P_{Bi}^{up}$ 为起点，找 $P_{Bi}^{up}$ 到 B 的最短路径，

          pointListUP.add(FindShortestPathBetweenTwoPoints($P_{Bi}^{up}$, B));

    **else**

      可知：$P_{Ai}^{up}$ 不在 A 到 B 的最短路径上;

  **for** $i \leftarrow 1$ **to** $n$ **do**

    计算：点 A 对障碍物 $O_i$ 的下边缘可见点 $P_{Ai}^{down}$;

    **if** 直线 $AP_{Ai}^{down}$ 与 $O_{i+1}, O_{i+2} ... O_n$ 均不相交 **then**

      可知：$P_{Ai}^{down}$ 在 A 到 B 的最短路径上，pointListDOWN.add($P_{Ai}^{down}$);

      计算：点 B 对障碍物 $O_i$ 的下边缘可见点 $P_{Bi}^{down}$;

      **if** $P_{Bi}^{down}$ 与 $P_{Ai}^{down}$ 是同一个点 **then**

        **if** 线段 $P_{Bi}^{down}B$ 不穿过障碍物 **then**

          $A \rightarrow P_{Bi}^{down} \rightarrow B$ 为最短路径，pointListDOWN.add(B);

        **else**

          以 $P_{Ai}^{down}$ 为起点，找 $P_{Ai}^{down}$ 到 B 的最短路径，

          pointListDOWN.add(FindShortestPathBetweenTwoPoints($P_{Ai}^{down}$, B));

      **else**

        **if** 直线 $P_{Bi}^{down}P_{Ai}^{down}$ 穿过障碍物 $O_{i+1}, O_{i+2} ... O_n$ 其中一个或多个 **then**

          可知：$P_{Bi}^{down}$ 不在最短路径上;

          以 $P_{Ai}^{down}$ 为起点，找 $P_{Ai}^{down}$ 到 B 的最短路径，

          pointListDOWN.add(FindShortestPathBetweenTwoPoints($P_{Ai}^{down}$, B));

        **else**

          可知：$P_{Bi}^{down}$ 在最短路径上，pointList.add($P_{Bi}^{down}$);

          以 $P_{Bi}^{down}$ 为起点，找 $P_{Bi}^{down}$ 到 B 的最短路径，

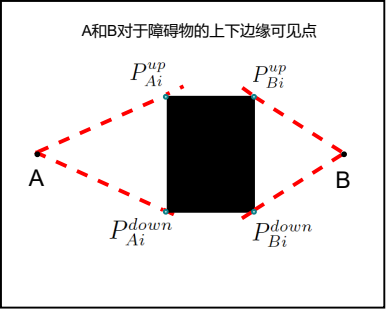
          pointListDOWN.add(FindShortestPathBetweenTwoPoints($P_{Bi}^{down}$, B));

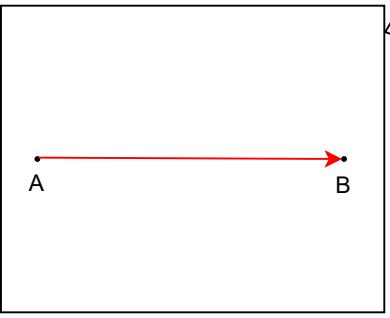
    **else**

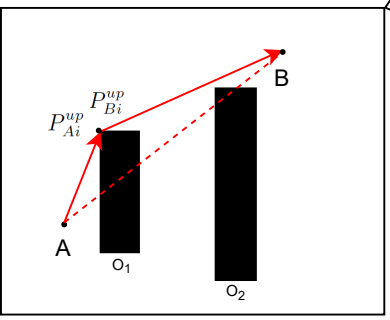
      可知：$P_{Ai}^{down}$ 不在 A 到 B 的最短路径上;

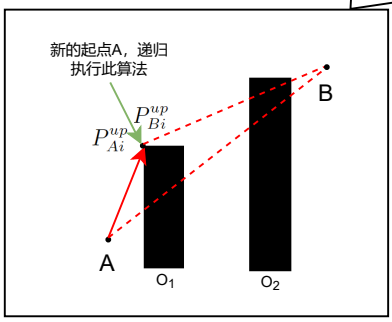
  计算：比较 $A \rightarrow B$ 的两条路径 pointListUP 和 pointListDOWN 的路径长度哪条更短;

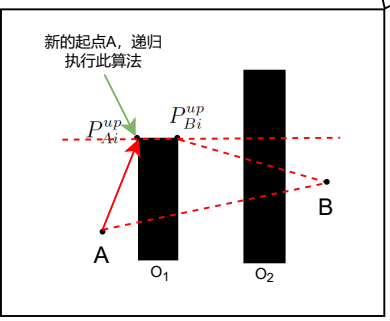
  return pointListUP 和 pointListDOWN 二者中路径长度更短的那条路径列表;

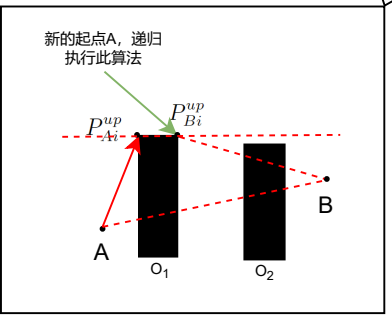


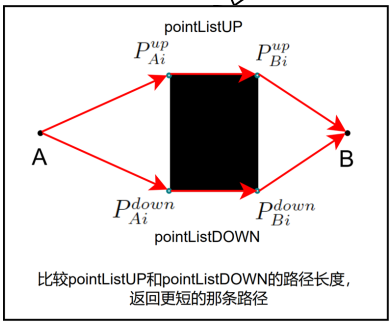




**Algorithm 1:** FindShortestPathBetweenTwoPoints(A, B)

**input** ：A(x, y)，B(x, y)，障碍物集合
**output** 构成最短路径的路径点列表 pointList
：
注：此算法仅考虑二维平面；障碍物均为矩形；路径点列表 *pointList* 记录了最短路径上的必经点，但 *pointList* 不包括真实起点 *A*，在函数外手动添加；
初始化路径点列表 pointList、pointListUP、pointListDOWN;
**if** 线段 $AB$ 之间没有障碍物 **then**
  $A \rightarrow B$ 为最短路径，pointList.add(B);
  return pointList;
**else**
  计算：线段 AB 依次穿过障碍物 $O_1, O_2, O_3 ... O_n$;
  **for** $i \leftarrow 1$ **to** $n$ **do**
    计算：点 A 对障碍物 $O_i$ 的上边缘可见点 $P_{Ai}^{up}$;
    **if** 直线 $AP_{Ai}^{up}$ 与 $O_{i+1}, O_{i+2} ... O_n$ 均不相交 **then**
      可知：$P_{Ai}^{up}$ 在 A 到 B 的最短路径上，pointListUP.add($P_{Ai}^{up}$);
      计算：点 B 对障碍物 $O_i$ 的上边缘可见点 $P_{Bi}^{up}$;
      **if** $P_{Bi}^{up}$ 与 $P_{Ai}^{up}$ 是同一个点 **then**
        **if** 线段 $P_{Bi}^{up}B$ 不穿过障碍物 **then**
          $A \rightarrow P_{Bi}^{up} \rightarrow B$ 为最短路径，pointListUP.add(B);
        **else**
          以 $P_{Ai}^{up}$ 为起点，找 $P_{Ai}^{up}$ 到 B 的最短路径，
          pointListUP.add(FindShortestPathBetweenTwoPoints($P_{Ai}^{up}$, B));
      **else**
        **if** 直线 $P_{Bi}^{up}P_{Ai}^{up}$ 穿过障碍物 $O_{i+1}, O_{i+2} ... O_n$ 其中一个或多个 **then**
          可知：$P_{Bi}^{up}$ 不在最短路径上;
          以 $P_{Ai}^{up}$ 为起点，找 $P_{Ai}^{up}$ 到 B 的最短路径，
          pointListUP.add(FindShortestPathBetweenTwoPoints($P_{Ai}^{up}$, B));
        **else**
          可知：$P_{Bi}^{up}$ 在最短路径上，pointListUP.add($P_{Bi}^{up}$);
          以 $P_{Bi}^{up}$ 为起点，找 $P_{Bi}^{up}$ 到 B 的最短路径，
          pointListUP.add(FindShortestPathBetweenTwoPoints($P_{Bi}^{up}$, B));
    **else**
      可知：$P_{Ai}^{up}$ 不在 A 到 B 的最短路径上;
  **for** $i \leftarrow 1$ **to** $n$ **do**
    计算：点 A 对障碍物 $O_i$ 的下边缘可见点 $P_{Ai}^{down}$;
    **if** 直线 $AP_{Ai}^{down}$ 与 $O_{i+1}, O_{i+2} ... O_n$ 均不相交 **then**
      可知：$P_{Ai}^{down}$ 在 A 到 B 的最短路径上，pointListDOWN.add($P_{Ai}^{down}$);
      计算：点 B 对障碍物 $O_i$ 的下边缘可见点 $P_{Bi}^{down}$;
      **if** $P_{Bi}^{down}$ 与 $P_{Ai}^{down}$ 是同一个点 **then**
        **if** 线段 $P_{Bi}^{down}B$ 不穿过障碍物 **then**
          $A \rightarrow P_{Bi}^{down} \rightarrow B$ 为最短路径，pointListDOWN.add(B);
        **else**
          以 $P_{Ai}^{down}$ 为起点，找 $P_{Ai}^{down}$ 到 B 的最短路径，
          pointListDOWN.add(FindShortestPathBetweenTwoPoints($P_{Ai}^{down}$, B));
      **else**
        **if** 直线 $P_{Bi}^{down}P_{Ai}^{down}$ 穿过障碍物 $O_{i+1}, O_{i+2} ... O_n$ 其中一个或多个 **then**
          可知：$P_{Bi}^{down}$ 不在最短路径上;
          以 $P_{Ai}^{down}$ 为起点，找 $P_{Ai}^{down}$ 到 B 的最短路径，
          pointListDOWN.add(FindShortestPathBetweenTwoPoints($P_{Ai}^{down}$, B));
        **else**
          可知：$P_{Bi}^{down}$ 在最短路径上，pointList.add($P_{Bi}^{down}$);
          以 $P_{Bi}^{down}$ 为起点，找 $P_{Bi}^{down}$ 到 B 的最短路径，
          pointListDOWN.add(FindShortestPathBetweenTwoPoints($P_{Bi}^{down}$, B));
    **else**
      可知：$P_{Ai}^{down}$ 不在 A 到 B 的最短路径上;
  计算：比较 $A \rightarrow B$ 的两条路径 pointListUP 和 pointListDOWN 的路径长度哪条更短;
  return pointListUP 和 pointListDOWN 二者中路径长度更短的那条路径列表;