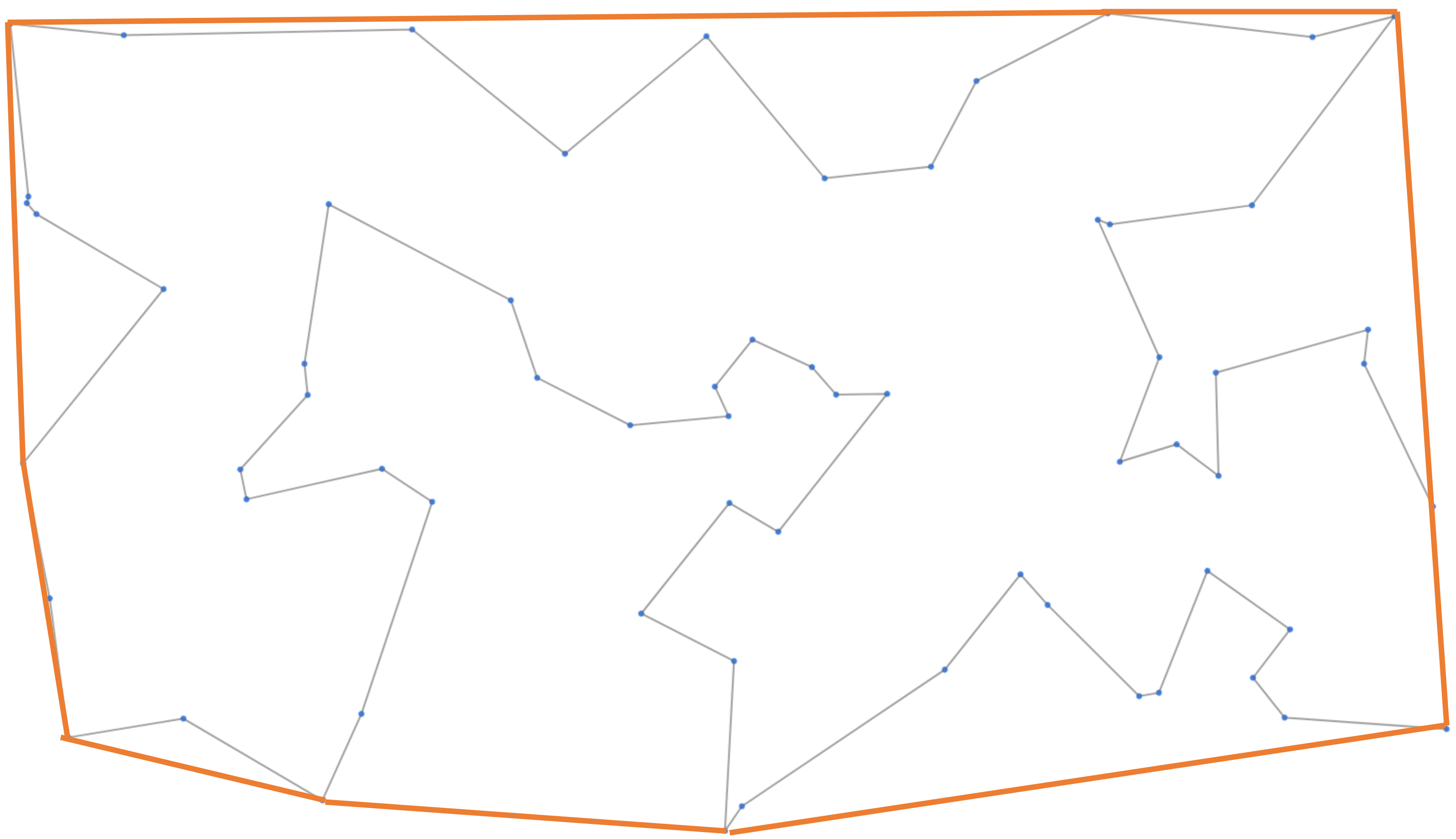
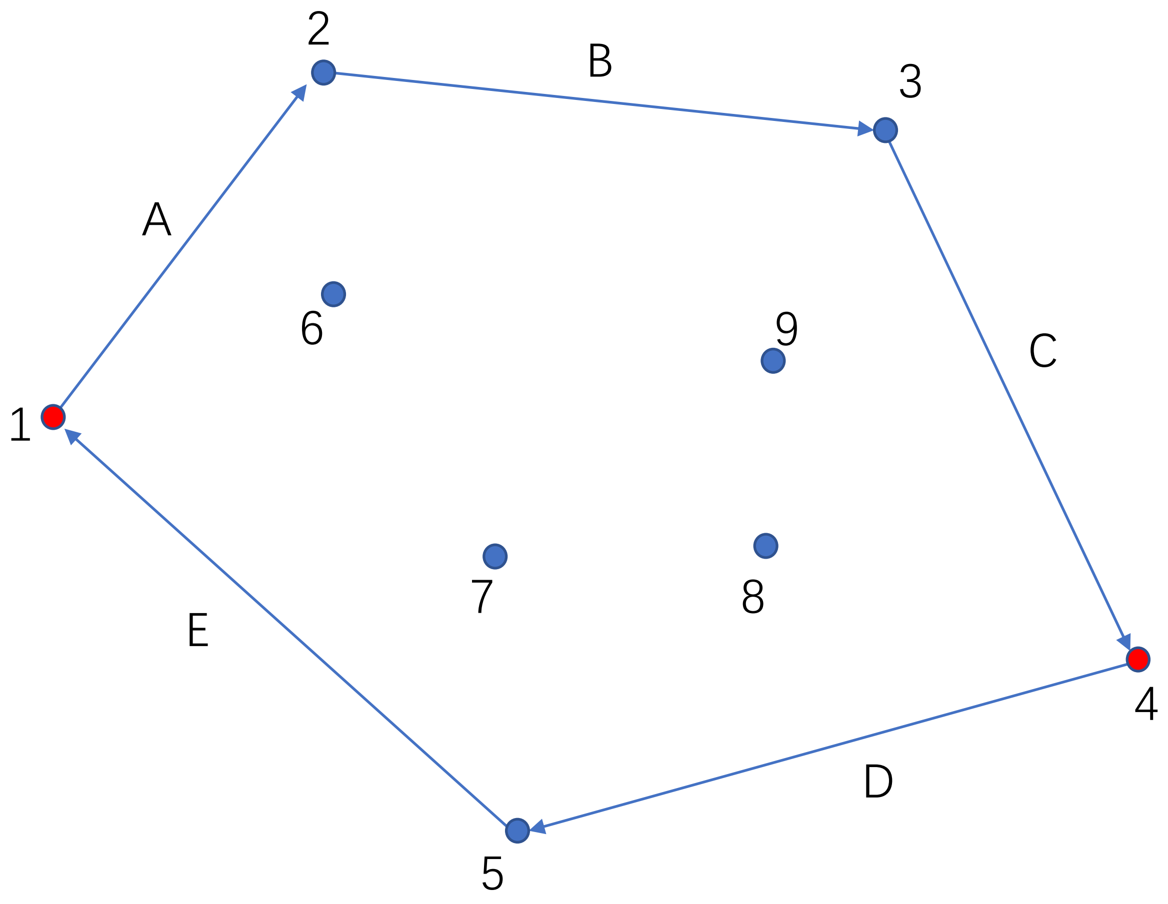
Find the convex hull



Example: n = 64

The points in convex hull will have the same relative order in the best result.
(If not, there will be intersect in the best result, which cause contradictions.)

Find the convex hull



To find the convex hull, we first find the leftmost point and rightmost point in all points. Then find the upper part and lower part of the convex hull.

We search the points by clockwise, so that in for each point, we should find the line segment with the biggest slope and smaller then the previous line segment.

i.e.

Upper part: A -> B -> C

Lower part: D -> E

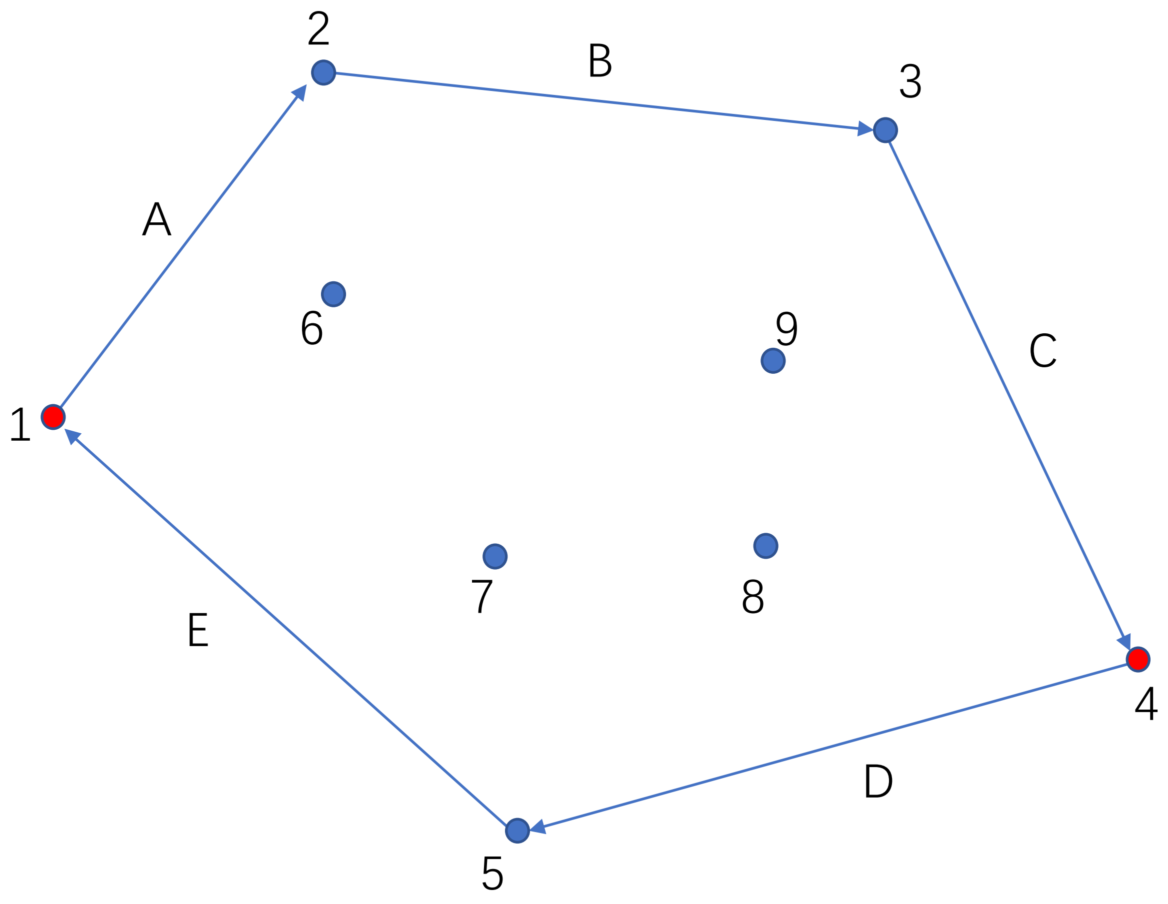
we have slope(A) > slope(B) > slope(C)

And slope(D) > slope(E)

Then we conbine 2 parts and can get the convex hull.

Time Comlexity: $O(n^2)$

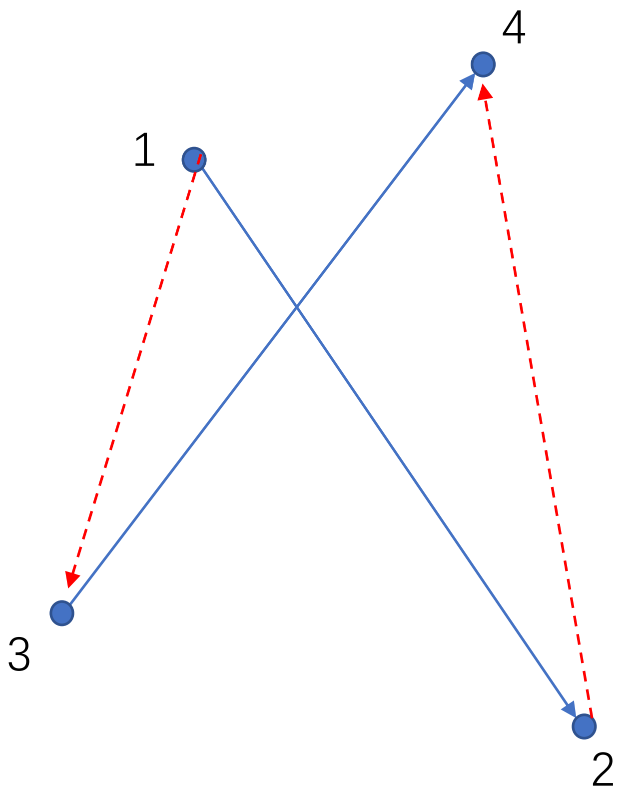
Greedy insert



Then we use greedy to insert inner points (6,7,8,9)
The existing route is [1,2,3,4,5,1]
And we try to insert each inner points (6,7,8,9) to each position in the existing route ([1,2,3,4,5,1]). And find the best choice which have the minimum distance increase.
i.e. if insert 6 to [1,2] is the optimum
We will have [1,6,2,3,4,5,1] as existing route and inner points (7,8,9).
Then we can do the next insertion.

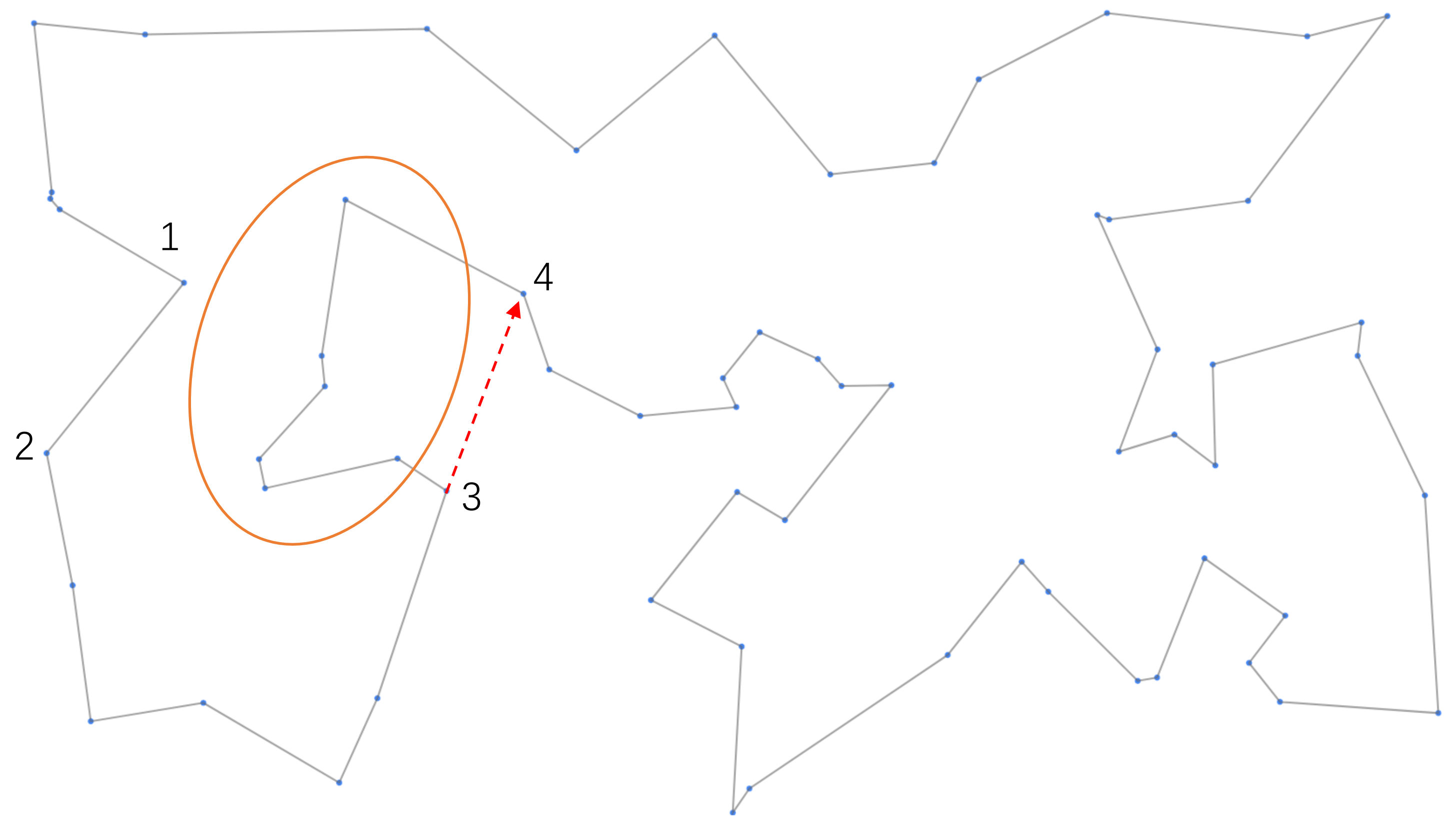
Time Comlexity:
$O(n^3)$ for brute force or
$O(n^2 \log(n))$ for binary search

Intersection swap



First, find the line intersection and judge it is in the line segment or not.
If so, we can change 1-> 2 and 3 -> 4 to
1 -> 3 and 2 -> 4.
This change decrease the total distance because of the triangle inequality.

Re-insert sequent points group

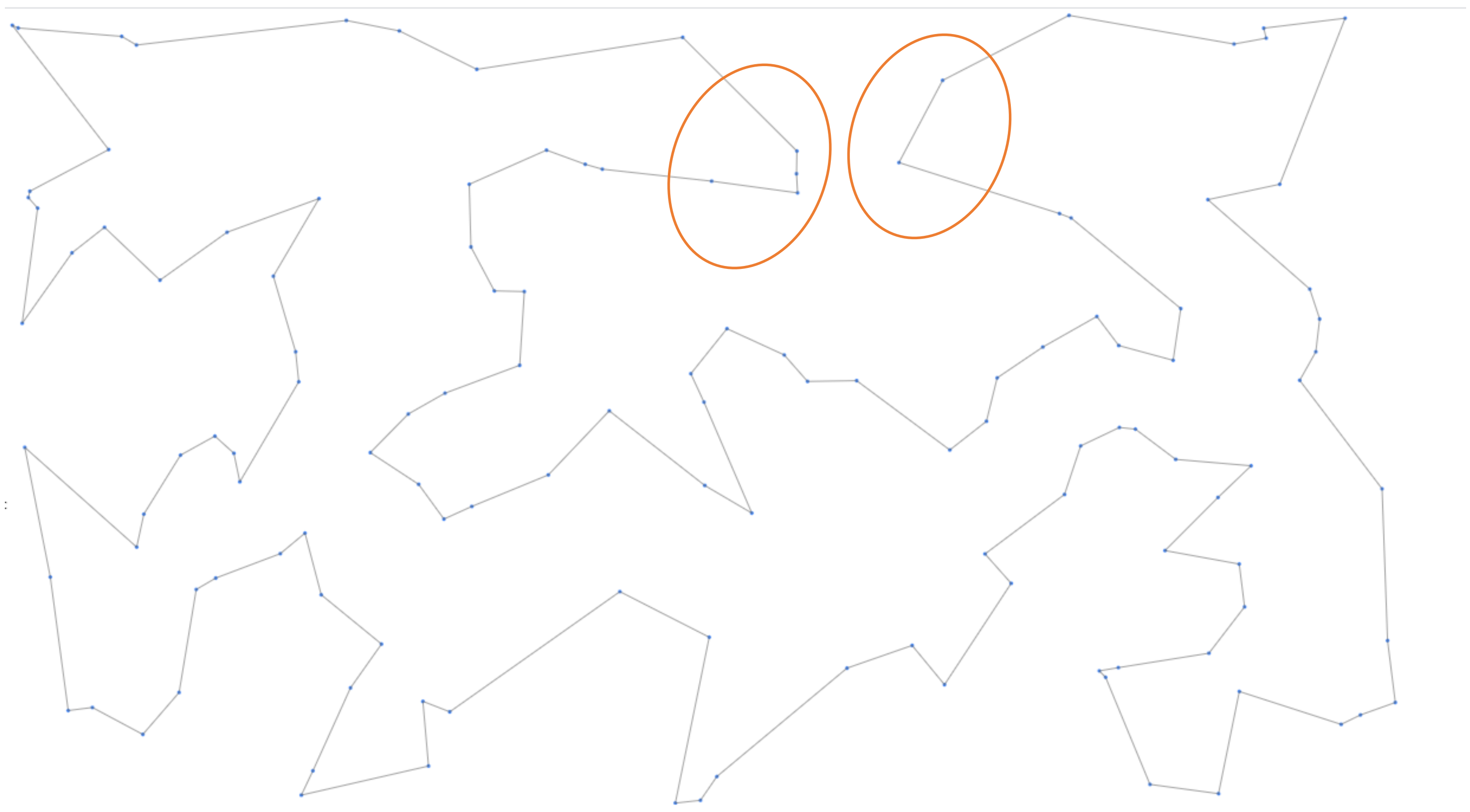


Example: n = 64

We use greedy for each point when in the first insertion, so it just ensure the optimum for one point but not for some gathering points.
i.e. like ponts in the red area, they are now connected by 3 and 4. But they may be better to connect 1 and 2.
According to such hypothesis, we can separate sequent points (no more than m) from the result and re-insert them to the result again.
i.e. separate the points in the red area and connect 3 -> 4. then use greedy to re-insert them.

Time Comlexity: $O(n^2m^2)$
(For each start point, there are m end points, and need to insert m points)

Re-insert two sequent points groups



Example: n = 128

Similar to the last page, consider if two sequent points groups need to re-insert at the same time.
We can separate them from the result and do the similar re-insert.

Time Comlexity: $O(n^2m^3)$
(Let size of each group be m)
we can choose exactly m elements for each group, it will be $O(n^2m)$