Generating a convex hull with the Divide-And-Conquer (DAC) algorithm.

Shinkook Cha (Student ID: 22000741)

What is the Divide-And-Conquer (DAC) algorithm?

The Divide-and-Conquer algorithm in a nutshell could be described as a way to solve a bigger problem by dividing it into smaller problems of the same type.

For example, let's say that we want to sort a sequence of numbers in ascending order. We could solve this problem by dividing the numbers into a subset of 2 numbers, sorting the two numbers in each subset, and sorting two sorted sets recursively until the sets merge into a single set.

In detail, the Divide-And-Conquer Algorithm has the following paradigm [1]:

Step 1: Divide the problem into sub-problems, essentially the same as the original, but smaller in size.
Step 2: Conquer the sub-problems by solving them recursively.
Step 3: Combine the results of the sub-problems to create a solution to the original problem.

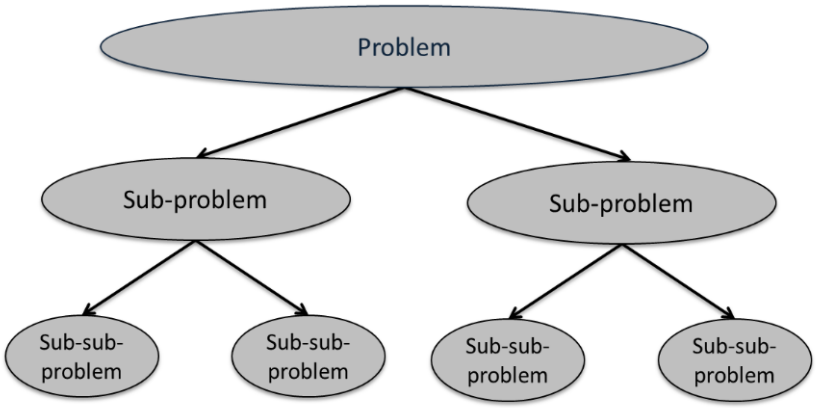

Fig 1. Illustration of the Divide-And-Conquer Algorithm

by Prof. Junghyun Kim, School of Applied Artificial Intelligence, Handong Global University

Algorithm Divide-and-Conquer (DAC) algorithm

```
1: Define a problem
2:     if the problem is small enough:
3:         Directly solve the problem
4:     else if:
5:         Divide the problem into sub-problems (e.g., P_1, P_2, P_3, ⋯, P_k)
6:         Conquer the sub-problems
7:         Combine the results of the sub-problems
8: End
```

Fig 2. Pseudocode of the Divide-and-Conquer Algorithm

by Prof. Junghyun Kim, School of Applied Artificial Intelligence, Handong Global University

The Convex Hull Problem:

In the Convex Hull Problem, given a set of points, a convex hull of the set of points must be outputted.

What is a Convex Hull?

The convex hull is a polygon that **encloses all the points** in a set **without having any in-bending corners** [2]

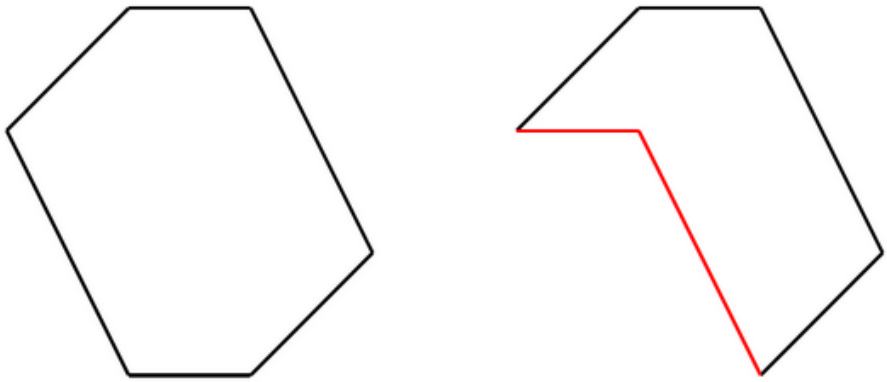

Fig 3. A convex polygon on the left side, non-convex on the right side.

The polygon on the left-hand side of the image is a convex hull since there are no in-bending corners. On the other hand, the polygon on the right-hand side is not a convex hull since the sides of the polygon colored in red are in-bending.

Convex Hull Problem Solution using the DAC Algorithm [3]:

Step 1. Sort the input points array based on x components.
Step 2. Divide the input points into two parts, as equally as possible, by a divider, an x coordinate that forms a vertical line.
Step 3. Repeat until the subproblem becomes only one point
Step 4. Merge the subproblems recursively.

In step 4, the **Two-Finger Method** – an algorithm developed by Professor Srinivas Devadas – is utilized to find the upper and the lower tangent lines prior to the merging of the convex hulls.

The Two-Finger Method:

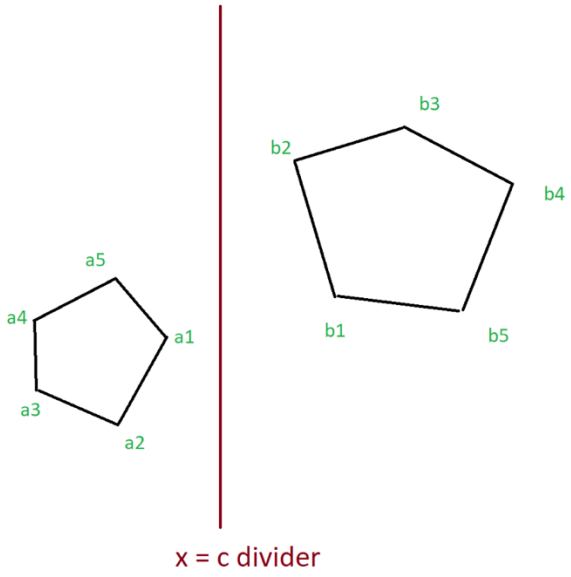

Fig 4. Two Polygons

For the given set of polygons, the Two-Finger Method follows the sequence of steps below to find the upper tangent line indices:

Step 1. Find the point that is closest to the divider on both sides.
Step 2. Connect them and find the intersection point.

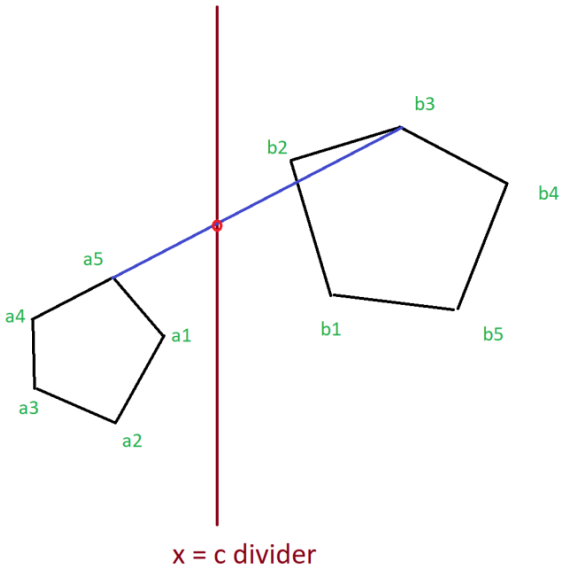

Fig 5. Step 2. Connect them and find the intersection point.

Step 3. Try the next point **clockwise** on the **right side** but **leave the left side intact**. If the resulting intersection has a bigger y intersection, we work on the left. If not, we move back to the last point, a1.

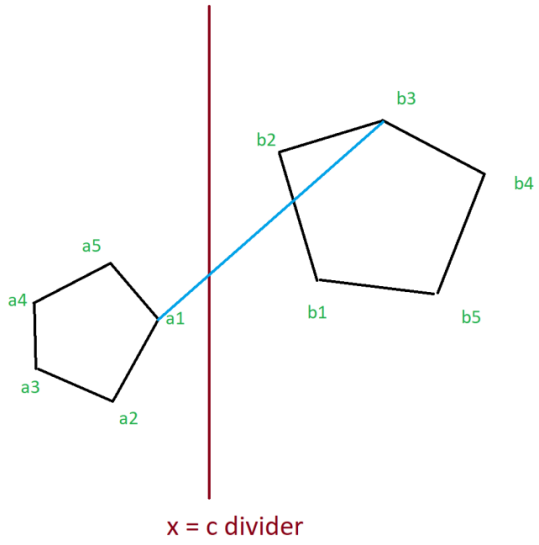

Fig 6. Step 3. Try the next point clockwise on the right side.

Step 4. Try the next point **counterclockwise** on the left, which is a5. And see if that can increase our recorded y intersection value. If not, we move back, otherwise, we keep the change.

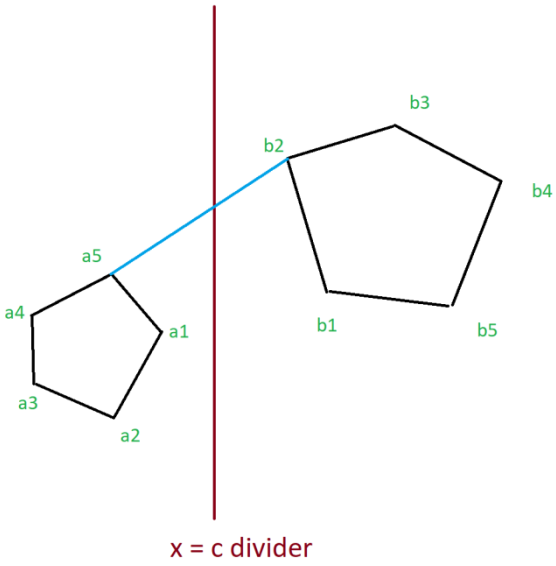

Fig 7. Step 4. Try the next point counterclockwise on the left.

Step 5. keep the process and work on two sides alternatively until both sides are not able to optimize the intersection value.
The following steps of the two-finger algorithm output the **upper tangent indices** of the merged convex hull, which is a5-b2.

To find the **lower tangent line indices** of the merged convex hull, move the index **counterclockwise in the right polygon** and **clockwise for the left polygon**. For instance, initially, the vertices connected are a1-b2. However, in the next step, move from b2 -> b1 instead of b2. -> b3 to calculate the intersection value. As for the left polygon, move from a1 -> a2 instead of a1 -> a5 to calculate the intersection value.

Step 4. Merge Two Convex Hulls:

1. Add the left convex hull's upper tangent points to the new convex hull array.
2. Add the right convex hull's upper tangent points to the new convex hull array.
3. Add the points between the right convex hull's upper tangent point and the right convex hull's lower tangent point (inclusive of the lower tangent point) in a clockwise direction to the new convex hull array.
4. Add the left convex hull's lower tangent points to the new convex hull array.
5. Add the points between the left convex hull's lower tangent point and the left convex hull's upper tangent point (exclusive of the upper tangent point) in a clockwise direction to the new convex hull array.

Result:

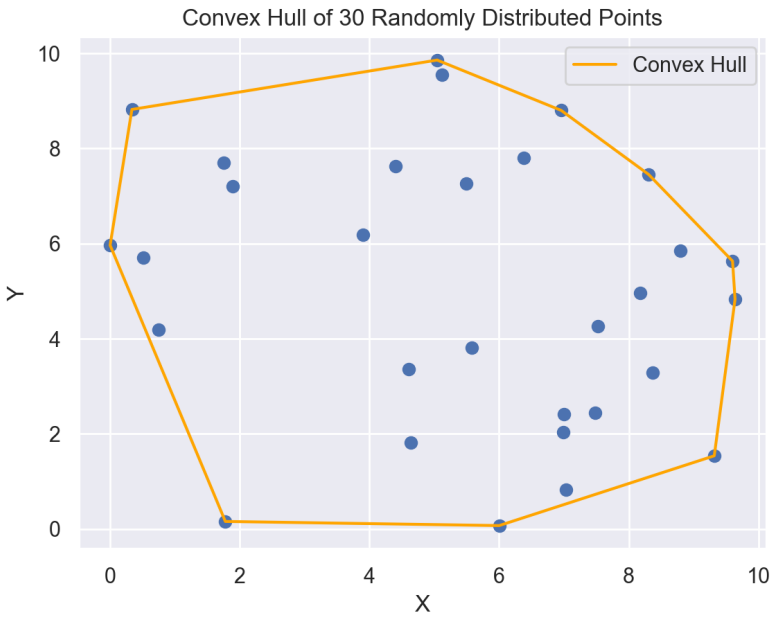

Fig 8. Convex Hull of 30 Points

References

[1] Module 5-1-2 Divde and conquer algorithm (n.d.) - Convex hull. by Prof. Junghyun Kim, School of

Applied Artificial Intelligence, Handong Global University

[2] Sommer, P. (2020, June 19). A gentle introduction to the convex hull problem - Pascal Sommer - Medium. *Medium*. https://medium.com/@pascal.sommer.ch/a-gentle-introduction-to-the-convex-hull-problem-62dfcabee90c

[3] jingyuLiu, V. a. P. B. (2018, January 26). *[Algorithm] Two Finger Convex Hull Merging Algorithm*. Jingyu Liu. https://jayjingyuliu.wordpress.com/2018/01/24/algorithm-two-finger-convex-hull-merging-algorithm/