The Convex Hulls

Implementation of Two-dimensional Convex Hull Algorithms and Comparing Their Performances

Zeynep Nur Öztürk

Outline

1.Introduction

2. Background for Convex Hulls

3. Progress

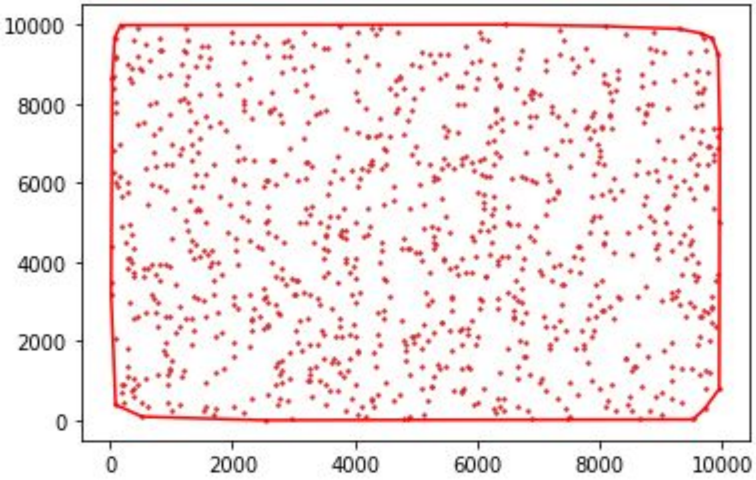
3. Result

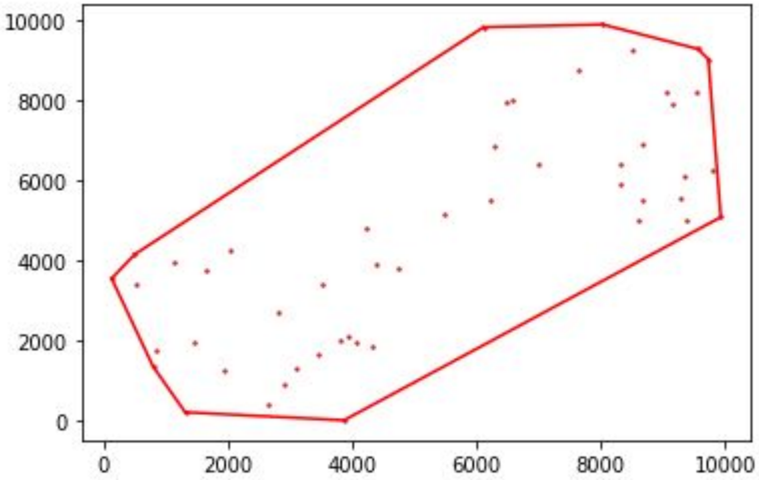
4. Demo

1. Introduction

What is Convex Hull?

Convex hull is the smallest convex set which contains all the points in it.





Why Convex Hull?

It has wide application like in

- Math-> analyze asymptotic beh. of poynomials
- Statistic -> visualize the spread of 2D sample points
- Economic -> When actual data non-complex, made convex
- Geometric modeling -> In Bezier curve, quickly detecting intersections of curves

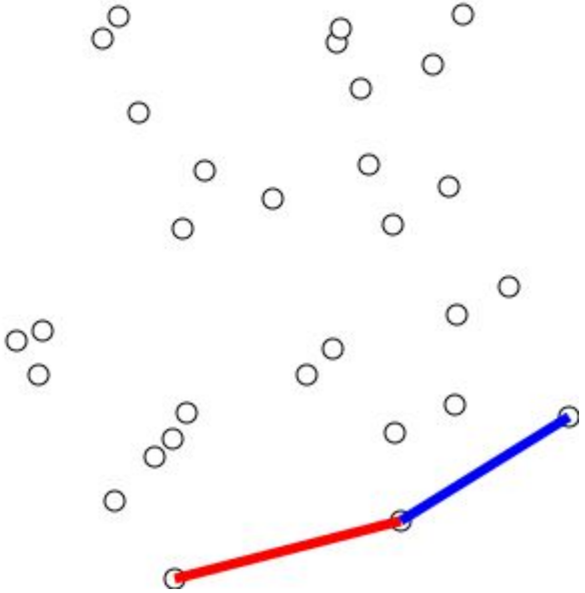
2. Background

Algorithms

- In general, most of the algorithms can be solved in time O(nlogn) for 2D or 3D
    - Graham Scan -> O(nlogn)
    - Gift wrapping (Jarvis March) -> O(nlogn)
    - Quickhull -> ave. O(nlogn) -> wor. O(n^2)
    - Divide & Conquer -> O(nlogn)

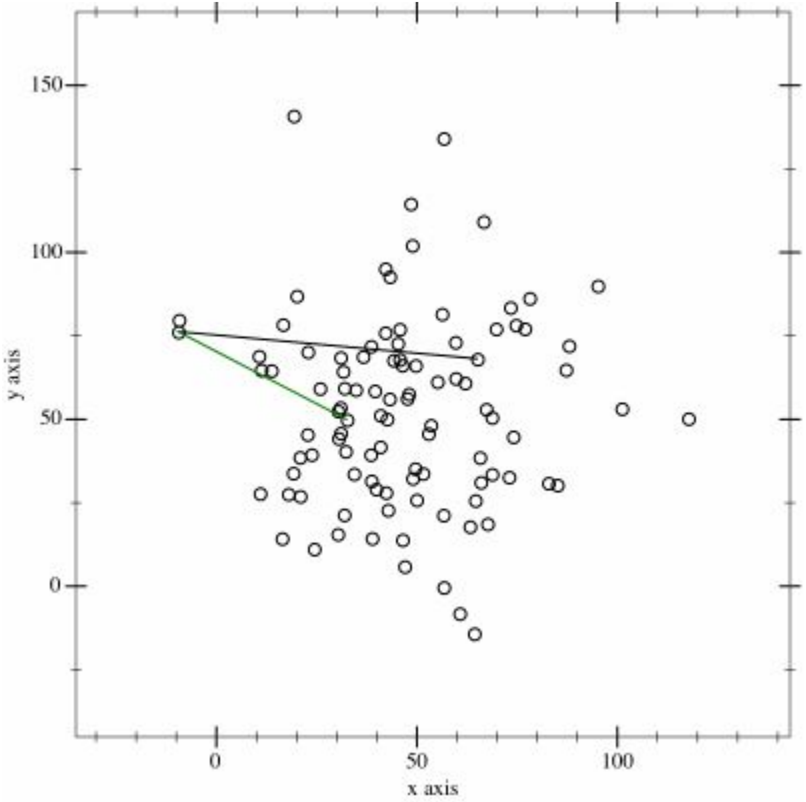
Graham Scan

- Ronald Graham 1972
- O(nlogn)
- First order vertices based on polar angle
- By left test, move around the points
- If left test is true add to stack or delete from until it is true



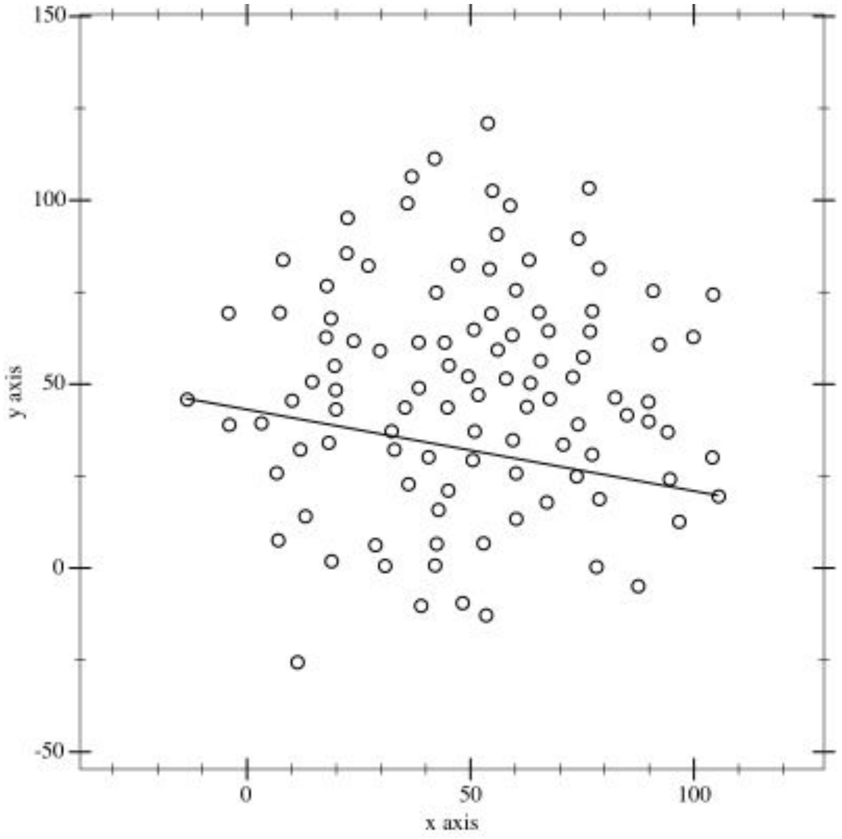
Jarvis March

- R. A. Jarvis-1973
- O(nh)
- It takes smallest point based on x or y.
- Sort all points based on polar angle
- Beside Graham, only calculates the points on the convex hull.



Quickhull

- Bradford Barber-1996
- Worst case O(n^2) Average case O(nlogn)
- Divides points into 2
- Find farthest point by calculating triangle area as recursive until no points left on recursive function



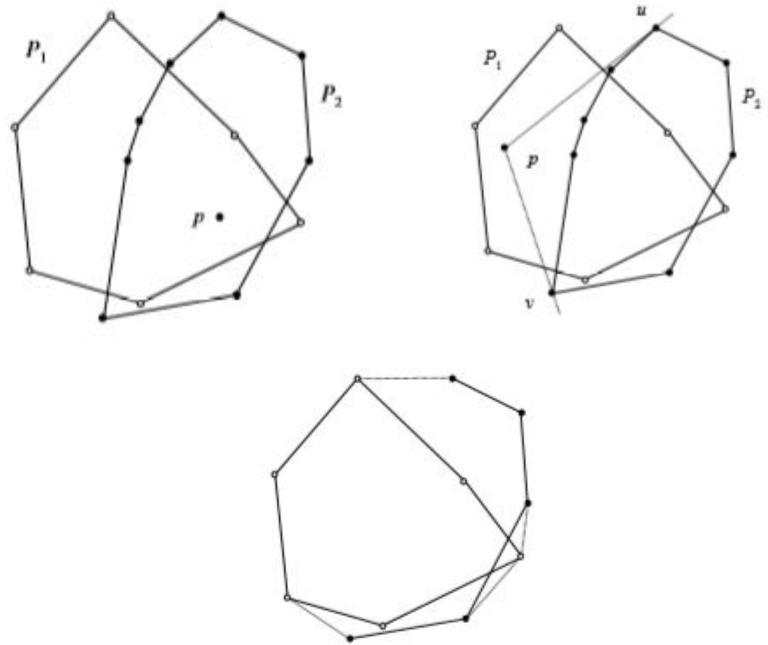
Divide and Conquer

- Quickhull recursively
- Then Merge them

Merge Hull (Merging Process of D&C)

1. P1 & P2 are convex hulls, find p internal P1
2. If p internal P2 -> step 3
   Else -> step 4
3. Sort based on polar for p
4. Find u and v and eliminate middle chain in P2
5. Graham Scan

- Merge: p1 -> m p2-> n So, O(m+n)=O(N)
- In Total O(NlogN)



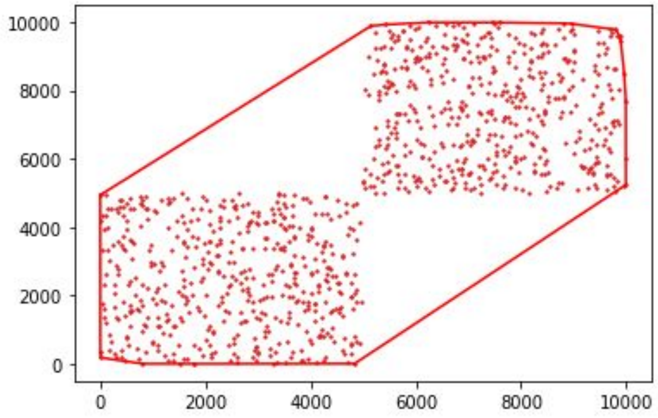
3. Progress

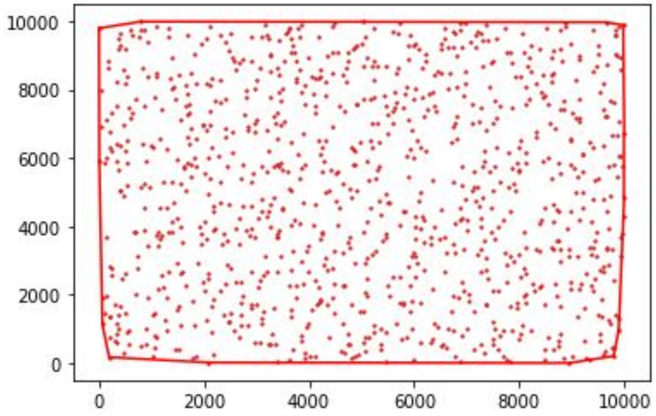
Language & Libraries

- Python 3 for coding
- The libraries:
  - random for point generation
  - math for tan calculations
  - numpy as array
  - time for timing calculations
  - matplotlib to show visual results

Features

- Clustered and non-clustered points
- from 1000 to 10.000.000 random point size
- Time calculation





4. Results

Graham Scan	Uniform(s)	Clustered(s)	Expected
1,000	0.00348	0.00343	O(nlogn)=3,000
10,000	0.04643	0.04083	O(nlogn)=40,000
100,000	1.26024	0.48540	O(nlogn)=500,000
1,000,000	7.10247	7.25312	O(nlogn)=6,000,000
2,000,000	14.91750	14.30834	O(nlogn)=12,602,060
3,000,000	23.81629	23.92488	O(nlogn)=19,431,363
4,000,000	31.19035	30.92443	O(nlogn)=26,408,240
5,000,000	40.74763	38.05435	O(nlogn)=33,494,850
10,000,000	87.29779	80.77861	O(nlogn)=70,000,000

Jarvis' March	Uniform(s)	Clustered(s)	Expected
1,000	0.01776 h = 23	0.01535 h = 18	u->O(nh)=23,000 c->O(nh)=18,000
10,000	0.23531 h = 26	0.25829 h = 26	u->O(nh)=260,000 c->O(nh)=260,000
100,000	5.52423 h = 37	4.01435 h = 35	u->O(nh)=3,700,000 c->O(nh)=3,500,000
1,000,000	66.89897 h = 33	69.90606 h = 41	u->O(nh)=33,000,000 c->O(nh)=41,000,000
2,000,000	174.11069 h = 47	115.76362 h = 44	u->O(nh)=94,000,000 c->O(nh)=88,000,000
3,000,000	244.66016 h = 45	295.58773 h = 48	u->O(nh)=135,000,000 c->O(nh)=144,000,000
4,000,000	340.58623 h = 40	414.12098 h = 48	u->O(nh)=160,000,000 c->O(nh)=192,000,000
5,000,000	353.88278 h = 47	469.60794 h = 44	u->O(nh)=235,000,000 c->O(nh)=220,000,000
10,000,000	356.32224 h=50	1145.9719 h=66	u->O(nh)=500,000,000 c->O(nh)=660,000,000

Quickhull	Uniform(s)	Clustered(s)	Expected
1,000	0.00512	0.00384	O(nlogn)=3,000 O(n*n)=10^6
10,000	0.05689	0.03930	O(nlogn)=40,000 O(n*n)= 10^8
100,000	0.71644	0.55157	O(nlogn)=500,000 O(n*n)=10^10
1,000,000	5.52458	5.60349	O(nlogn)=6,000,000 O(n*n)=10^12
2,000,000	11.87036	11.54690	O(nlogn)=12,602,060 O(n*n)=4*10^12
3,000,000	X	17.12558	O(nlogn)=19,431,363 O(n*n)=9*10^12
4,000,000	X	X	O(nlogn)=26,408,240 O(n*n)=16*10^12
5,000,000	X	X	O(nlogn)=33,494,850 O(n*n)=25*10^12
10,000,000	X	X	O(nlogn)=70,000,000 O(n*n)=10^14

Merge Hull	Uniform(s)	Clustered(s)	Expected
1,000	0.00414	0.00391	O(nlogn)=3,000
10,000	0.04171	0.03664	O(nlogn)=40,000
100,000	0.52161	0.44887	O(nlogn)=500,000
1,000,000	6.18138	6.00312	O(nlogn)=6,000,000
2,000,000	12.24851	13.15488	O(nlogn)=12,602,060
3,000,000	17.88851	26.40263	O(nlogn)=19,431,363
4,000,000	25.77283	31.05477	O(nlogn)=26,408,240
5,000,000	32.26809	42.65405	O(nlogn)=33,494,850
10,000,000	X	X	O(nlogn)=70,000,000

Clustered	Graham	Jarvis	Quickhull	Merge
1,000,000	7.25312	69.90606 h = 41	5.60349	6.00312
2,000,000	14.30834	115.76362 h = 44	11.54690	13.15488
3,000,000	23.92488	295.58773 h = 48	17.12558	26.40263
4,000,000	30.92443	414.12098 h = 48	X	31.05477
5,000,000	38.05435	469.60794 h = 44	X	42.65405
10,000,000	80.77861	1145.9719 h=66	X	X

Uniform	Graham	Jarvis	Quickhull	Merge
1,000,000	7.10247	66.89897 h = 33	5.52458	6.18138
2,000,000	14.91750	174.11069 h = 47	11.87036	12.24851
3,000,000	23.81629	244.66016 h = 45	X	17.88851
4,000,000	31.19035	340.58623 h = 40	X	25.77283
5,000,000	40.74763	353.88278 h = 47	X	32.26809
10,000,000	87.29779	356.32224 h=50	X	X

Demo