1. Using list data structure

**Pseudo code:**

function convex\_hull(polygon)

leftmost = find leftmost vertex of polygon

hull = [leftmost]

current = leftmost

while True

next\_vertex = None

for vertex in polygon

if vertex is current

continue

else if next\_vertex is None

next\_vertex = vertex

else

cross\_product = calculate cross product of (vertex - current) and (next\_vertex - current)

if cross\_product > 0

next\_vertex = vertex

else if cross\_product == 0

dist1 = calculate distance between vertex and current

dist2 = calculate distance between next\_vertex and current

if dist1 > dist2

next\_vertex = vertex

add next\_vertex to hull

current = next\_vertex

if current is leftmost

break

return hull

function read\_txt\_file(file\_path)

lines = read contents of file into list of strings

data = []

for line in lines

columns = split line into columns and convert to appropriate data type

data.append(columns)

return data

points = read\_txt\_file('ban5000w-0.01-adjlist.txt')

before = get current time in milliseconds

convex\_hull(points)

after = get current time in milliseconds

print "Time Used = ", after - before, " MilleSeconds"

1. Using queue data structure

**Pseudo Code:**

IMPORT math

FUNCTION convex\_hull(polygon):

# Find the leftmost vertex of the polygon

leftmost = minimum value in polygon with key as the first element of the vertex

# Create a list to store the convex hull vertices

hull = [leftmost]

# Start at the leftmost vertex and iterate clockwise

current = leftmost

WHILE True:

next\_vertex = None

FOR vertex in polygon:

IF vertex == current:

continue

ELIF next\_vertex is None:

next\_vertex = vertex

ELSE:

# Compare the angle between the current vertex and the next candidate vertex

# with the angle between the current vertex and the next\_vertex

cross\_product = (vertex[0] - current[0]) \* (next\_vertex[1] - current[1]) - (vertex[1] - current[1]) \* (next\_vertex[0] - current[0])

IF cross\_product > 0:

next\_vertex = vertex

ELIF cross\_product == 0:

# If the cross product is 0, choose the vertex that is farther away

dist1 = square root of (vertex[0] - current[0])^2 + (vertex[1] - current[1])^2

dist2 = square root of (next\_vertex[0] - current[0])^2 + (next\_vertex[1] - current[1])^2

IF dist1 > dist2:

next\_vertex = vertex

# Add the next vertex to the convex hull

add next\_vertex to hull

# Update the current vertex

current = next\_vertex

# If we have completed a loop and returned to the leftmost vertex, exit the loop

IF current == leftmost:

break

RETURN hull

FUNCTION read\_txt\_file(file\_path):

# Read the contents of the file into a list of strings

open file with file\_path in read mode and assign to file

lines = read all lines of file

# Create an empty list to store the data

data = []

# Iterate through the lines and split them into columns

FOR line in lines:

columns = split line by space and remove the third element

columns[0] = convert first element of columns to integer

columns[1] = convert second element of columns to integer

add columns to data

RETURN data

points = read\_txt\_file('./ban5000w-0.01-adjlist.txt')

IMPORT time

before = current time in milliseconds

convex\_hull(points)

after = current time in milliseconds

print("Time Used = ", (after - before), " MilleSeconds")

1. Using stack data structure

**Code**

function convex\_hull(points):

xmin = ymin = float('inf')

xmax = ymax = float('-inf')

for x, y in points:

if x < xmin:

xmin = x

if y < ymin:

ymin = y

if x > xmax:

xmax = x

if y > ymax:

ymax = y

points = [p for p in points if not (p[0] == xmin or p[0] == xmax or p[1] == ymin or p[1] == ymax)]

sort points by x-coordinate

hull = []

for p in points:

while len(hull) > 1 and cross(hull[-2], hull[-1], p) <= 0:

remove last element from hull

add p to hull

return hull

function cross(p1, p2, p3):

return (p2[0] - p1[0]) \* (p3[1] - p1[1]) - (p2[1] - p1[1]) \* (p3[0] - p1[0])

FUNCTION read\_txt\_file(file\_path):

# Read the contents of the file into a list of strings

open file with file\_path in read mode and assign to file

lines = read all lines of file

# Create an empty list to store the data

data = []

# Iterate through the lines and split them into columns

FOR line in lines:

columns = split line by space and remove the third element

columns[0] = convert first element of columns to integer

columns[1] = convert second element of columns to integer

add columns to data

RETURN data

points = read\_txt\_file('./ban5000w-0.01-adjlist.txt')

IMPORT time

before = current time in milliseconds

convex\_hull(points)

after = current time in milliseconds

print("Time Used = ", (after - before), " MilleSeconds")

1. Using priority queue data structure

**Pseudo Code**

import math

function convex\_hull(points)

xmin, ymin, xmax, ymax = infinity, infinity, negative infinity, negative infinity

for (x, y) in points:

if x < xmin then xmin = x

if y < ymin then ymin = y

if x > xmax then xmax = x

if y > ymax then ymax = y

points = [p for p in points if p[0] ≠ xmin and p[0] ≠ xmax and p[1] ≠ ymin and p[1] ≠ ymax]

p0 = min(points, key=lambda p: (p[1], p[0]))

sort(points, key=lambda p: (angle(p0, p), distance(p0, p)))

hull = []

for p in points[:3]:

while length(hull) > 1 and cross(hull[-2], hull[-1], p) <= 0:

hull.pop()

hull.append(p)

for p in points[3:]:

while length(hull) > 1 and cross(hull[-2], hull[-1], p) <= 0:

hull.pop()

hull.append(p)

return hull

function angle(p1, p2)

return atan2(p2[1] - p1[1], p2[0] - p1[0])

function distance(p1, p2)

return sqrt((p2[1] - p1[1])^2 + (p2[0] - p1[0])^2)

function cross(p1, p2, p3)

return (p2[0] - p1[0]) \* (p3[1] - p1[1]) - (p2[1] - p1[1]) \* (p3[0] - p1[0])

function read\_txt\_file(file\_path)

open file\_path for reading and store the file in file

lines = read the lines of file

data = []

for line in lines:

columns = split line by white spaces and remove the 3rd column

convert the first and second columns to integers

append columns to data

return data

points = read\_txt\_file('ban5000w-0.01-adjlist.txt')

import time

before = current time in milliseconds

convex\_hull(points)

after = current time in milliseconds

print("Time Used = ", after - before, " MilleSeconds")