Design of Algorithms Assignment 1 Written Report
By Ai Ling Chiam (1384412)

Jarvis' March Algorithm

Jarvis March involves finding the convex hull for a set of points via a comparison between the angles of points using the orientation function provided. My choice of data structure for this algorithm is a doubly linked list. There are several reasons why a doubly linked list is used for this algorithm. It is easy to insert and remove a node from the list. It provides constant-time insertion and deletion on both the head and tail. In addition, it allows forward and backward transversals.

Graham's Scan Algorithm

Graham's Scan is similar to Jarvis' March, but instead of comparing the angles between points, it compares the angles during the sort. There is an alteration that I did in my program, which is the sorting process. The time complexity of the sorting algorithm is supposed to be O(n log n), which is by using mergesort or quicksort. Instead of using that, I used selection sort as my sorting process, as it is easier to understand and debug despite having a longer time complexity O(n^2). My choices of data structures are a stack and a doubly linked list. I used a stack because it is easier to push and pop points from the stack. However, it is difficult to push at the bottom of the stack, which would result in a longer time complexity. Hence why I also used a doubly linked list to do such operations. What my program does is that I sort the two arrays pointsX and pointsY by the angle relative to the lowest point, perform the push and pop operations to the stack based on the orientation of the top 3 points, and insert the stack into the doubly linked list.

Experimental Evaluation

I will evaluate the two algorithms by calculating the total number of basic operations across three input sets of different distinct sizes, each under three differing distribution conditions: random, points on a circle, and random points contained within a set of points making up a simple hull. I made the test-case generators with the assistance of ChatGPT. I calculate the total number of basic operations for each test case by copying the points into a new text file in the test_case folder called 1a-6.txt. I used solution->operationCount to count the number of basic operations.

Design of Algorithms Assignment 1 Written Report

By Ai Ling Chiam (1384412)

Input Sets Containing Random Points

Code used to generate the test cases:

```c
// Random number generator

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define MAX_POINTS 5 // Maximum number of points to generate

typedef struct {
    int x;
    int y;
} Coordinate;

Coordinate generateRandomCoordinate(int maxX, int maxY) {
    Coordinate randomCoord;

    // Generate random x and y coordinates within the specified range
    randomCoord.x = rand() % (maxX + 1);
    randomCoord.y = rand() % (maxY + 1);

    return randomCoord;
}

int main() {
    int maxX, maxY;

    // Prompt user for the maximum values of x and y coordinates
    printf("Enter the maximum value of x coordinate: ");
    scanf("%d", &maxX);
    printf("Enter the maximum value of y coordinate: ");
    scanf("%d", &maxY);

    // Seed the random number generator using the current time
    srand(time(NULL));

    printf("%d\n", MAX_POINTS);

    // Generate and print multiple random coordinates
    for(int i = 0; i < MAX_POINTS; ++i) {
        Coordinate randomCoordinate = generateRandomCoordinate(maxX, maxY);
        printf("%d %d\n", randomCoordinate.x, randomCoordinate.y);
    }
}
```

Design of Algorithms Assignment 1 Written Report
By Ai Ling Chiam (1384412)

Test Cases Used

Input set of 5 points	Input set of 15 points	Input set of 30 points
5 65 101 100 12 2 92 42 28 46 78	15 92 57 35 41 4 41 115 120 83 105 68 46 92 6 19 43 49 68 86 100 62 28 81 104 79 74 75 62 51 76	30 69 64 117 69 90 98 92 28 65 50 19 50 118 71 103 45 79 95 116 83 84 96 78 96 43 59 95 55 96 62 90 69 87 51 112 12 78 102 16 108 107 9 41 54 83 78 106 110 67 20 36 75 86 20 46 73 97 42 61 56

Results

Input Sets	Total Basic Operations Using Jarvis’ March Algorithm	Total Basic Operations Using Graham’s Scan Algorithm
5 Points	20	6
15 Points	75	91
30 Points	240	406

Design of Algorithms Assignment 1 Written Report
By Ai Ling Chiam (1384412)

Input Sets Containing Points of a Circle

Code to generate the test cases:

```c
// Random generator for points on a circle

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

typedef struct {
    double x;
    double y;
} Point;

void generatePointsOnCircle(double cx, double cy, double radius, int numPoints, Point points[]) {
    double angleIncrement = 2 * M_PI / numPoints;
    double angle = 0.0;

    for (int i = 0; i < numPoints; ++i) {
        points[i].x = cx + radius * cos(angle);
        points[i].y = cy + radius * sin(angle);
        angle += angleIncrement;
    }
}

int main() {
    double centerX = 2.0;  // X coordinate of the circle's center
    double centerY = 1.0;  // Y coordinate of the circle's center
    double radius = 50.0;   // Radius of the circle
    int numPoints = 5;    // Number of points on the circle
    Point points[numPoints];

    generatePointsOnCircle(centerX, centerY, radius, numPoints, points);

    printf("%d\n", numPoints);

    for (int i = 0; i < numPoints; ++i) {
        printf("%.2f %.2f\n", points[i].x, points[i].y);
    }

    return 0;
}
```

I made the circle radius constant, as it would be fair for all test cases.

Test Cases Used

Input set of 5 points	Input set of 15 points	Input set of 30 points
5 52.00 1.00	15 52.00 1.00	30 52.00 1.00

17.45 48.55 -38.45 30.39 -38.45 -28.39 17.45 -46.55	47.68 21.34 35.46 38.16 17.45 48.55 -3.23 50.73 -23.00 44.30 -38.45 30.39 -46.91 11.40 -46.91 -9.40 -38.45 -28.39 -23.00 -42.30 -3.23 -48.73 17.45 -46.55 35.46 -36.16 47.68 -19.34	50.91 11.40 47.68 21.34 42.45 30.39 35.46 38.16 27.00 44.30 17.45 48.55 7.23 50.73 -3.23 50.73 -13.45 48.55 -23.00 44.30 -31.46 38.16 -38.45 30.39 -43.68 21.34 -46.91 11.40 -48.00 1.00 -46.91 -9.40 -43.68 -19.34 -38.45 -28.39 -31.46 -36.16 -23.00 -42.30 -13.45 -46.55 -3.23 -48.73 7.23 -48.73 17.45 -46.55 27.00 -42.30 35.46 -36.16 42.45 -28.39 47.68 -19.34 50.91 -9.40

Results

Input Sets	Total Basic Operations Using Jarvis' March Algorithm	Total Basic Operations Using Graham's Scan Algorithm
5 Points	25	6
15 Points	225	91
30 Points	900	406

Input Sets Containing Random Points Within A Set of Points Making Up A Simple Hull

Design of Algorithms Assignment 1 Written Report
By Ai Ling Chiam (1384412)

A simplification that I would like to add is that the simple hull that I will be using would be a trapezium or similar to a trapezium. This might lead to similar input sets as the first 3 test cases. The convex hull that I will be using is (0, 0), (5, 0), (8, 3), (5, 5), (2, 3).

<u>Code to generate the test cases:</u>

```
// Generator for random points within the convex hull
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

#define MAX_POINTS 50 // Maximum number of points to generate

typedef struct {
    double x;
    double y;
} Point;

// Calculate the cross product of vectors (p1, p2) and (p1, p3)
double crossProduct(Point p1, Point p2, Point p3) {
    return (p2.x - p1.x) * (p3.y - p1.y) - (p2.y - p1.y) * (p3.x - p1.x);
}

// Calculate the distance between two points
double distance(Point p1, Point p2) {
    return sqrt((p2.x - p1.x) * (p2.x - p1.x) + (p2.y - p1.y) * (p2.y - p1.y));
}

// Check if a point is inside the convex hull
int isInsideConvexHull(Point hull[], int hullSize, Point p) {
    int i;
    for (i = 0; i < hullSize - 1; i++) {
        if (crossProduct(hull[i], hull[i+1], p) < 0) {
            return 0;
        }
    }
    if (crossProduct(hull[i], hull[0], p) < 0) {
        return 0;
    }
    return 1;
}
```

Design of Algorithms Assignment 1 Written Report

By Ai Ling Chiam (1384412)

```
// Generate random points inside the convex hull
void generateRandomPointsInConvexHull(Point hull[], int hullSize, int numPoints, Point randomPoints[]) {
    int i = 0;
    srand(time(NULL));
    while (i < numPoints) {
        double minX = hull[0].x, maxX = hull[0].x, minY = hull[0].y, maxY = hull[0].y;
        for (int j = 1; j < hullSize; j++) {
            if (hull[j].x < minX) minX = hull[j].x;
            if (hull[j].x > maxX) maxX = hull[j].x;
            if (hull[j].y < minY) minY = hull[j].y;
            if (hull[j].y > maxY) maxY = hull[j].y;
        }
        double randX = minX + ((double)rand() / RAND_MAX) * (maxX - minX);
        double randY = minY + ((double)rand() / RAND_MAX) * (maxY - minY);
        Point randomPoint = {randX, randY};
        if (isInsideConvexHull(hull, hullSize, randomPoint)) {
            randomPoints[i++] = randomPoint;
        }
    }
}
```

```
int main() {
    // Example points for the convex hull
    Point convexHull[] = {
        {0, 0},
        {5, 0},
        {8, 3},
        {5, 5},
        {2, 3}
    };
    int hullSize = 5;

    // Generate random points within the convex hull
    Point randomPoints[MAX_POINTS];
    int numPoints = 10; // Number of random points to generate
    generateRandomPointsInConvexHull(convexHull, hullSize, numPoints, randomPoints);

    printf("%d\n", numPoints);

    // Print the random points
    for (int i = 0; i < numPoints; i++) {
        printf("%.2f %.2f\n", randomPoints[i].x, randomPoints[i].y);
    }

    return 0;
}
```

Design of Algorithms Assignment 1 Written Report
By Ai Ling Chiam (1384412)

Test Cases Used

Input set of 5 points	Input set of 15 points	Input set of 30 points
5 0.57 0.46 2.67 2.78 6.04 2.39 4.65 2.32 3.65 2.24	15 5.77 3.93 4.30 2.42 3.92 1.58 2.71 1.30 3.87 1.47 7.32 2.62 3.36 0.24 2.80 1.34 4.30 3.28 5.38 3.42 1.79 0.05 5.38 1.47 1.53 0.54 3.47 3.89 5.67 1.04	30 5.77 1.68 4.38 2.21 3.95 0.07 3.29 0.28 2.68 1.14 4.31 4.53 1.22 0.33 5.69 1.31 2.27 0.47 4.14 3.15 2.50 1.37 3.93 3.22 5.45 0.88 0.27 0.34 2.21 2.73 3.58 1.82 6.54 1.86 5.32 0.46 4.71 4.21 1.74 1.06 3.57 3.13 3.05 3.21 0.52 0.17 7.15 2.66 5.07 0.28 1.45 2.14 1.53 0.25 3.61 1.43 5.44 1.45 0.77 0.56

Results

Input Sets	Total Basic Operations Using Jarvis' March Algorithm	Total Basic Operations Using Graham's Scan Algorithm
5 Points	15	6
15 Points	105	91
30 Points	270	406

Design of Algorithms Assignment 1 Written Report
By Ai Ling Chiam (1384412)

Conclusion

Based on the 9 test cases, it was proven that there are some cases where Jarvis March has fewer number of basic operations performed compared to Graham's Scan. I also noticed that the total number of basic operations Graham's Scan does is the same for all 3 input sets of different sizes. This is because, during the sorting process based on the polar angles relative to the small point, it goes through every point to check whether the angle is smaller than the current angle. This leads to the number of angle comparisons being the same. Based on the results produced, six out of 9 test cases resulted in Graham Scan having the least number of basic operations. Despite the 2 algorithms having the same time complexity* O(n^2), Graham's Scan algorithm is more efficient compared to Jarvis' March algorithm.

* = My implementation of Graham's Scan has the time complexity of O(n^2) due to the inefficient sorting algorithm I used, which is selection sort.