# Design and Analysis of Algorithms Lab Assessment-4

Name: Armaan Indani

Reg. No.: 22BCE3347

Slot: L41+42

Faculty: Dr. Joshva Devdas T – 16700

# Question 1:

SCOPE / BCSE204P\_VL2023240504902 / ASSESSMENT LAB 4



### **ASSESSMENT LAB 4**

Back

#### Question 1

Correct

Marked out of 2.50

▼ Flag question

Write a program to read the coordinates of the polygon (given as input ). Determine the boundary of the polygon using Graham Scan Algorithm

Test	Input	Result
1	0 3	The Boundary Coordinates are
	1 1	0 3
	2 2	4 4
	4 4	3 1
	0 0	0 0
	1 2	
	3 1	
	3 3	

```
Answer: (penalty regime: 0 %)
      #include <stdio.h>
       #include <limits.h>
   2
   3
       #include <float.h>
   4
      struct point
   5 🔻
           int x;
   6
   7
           int y;
   8
       };
   9
      float slope(struct point p1, struct point p2)
  10
  11 •
           if (p1.x == p2.x)
  12
  13
  14
               if (p1.y < p2.y)
  15
  16
                   return FLT MAX;
               }
  17
               else
  18
               {
  19
  20
                   return FLT_MIN;
  21
  22
  23
           float t = (float)(p1.y - p2.y) / (p1.x - p2.x);
  24
           return t;
  25
  26
       void swap(struct point *p1, struct point *p2)
  27
  28 ▼ {
           int temp_x = p1->x;
  29
  30
           int temp_y = p1->y;
  31
  32
           p1->x = p2->x;
  33
           p1->y = p2->y;
   34
   35
           p2->x = temp_x;
   36
           p2->y = temp_y;
   37
   38
   39
       int orientation(struct point prev, struct point curr, struct point next
   40 ▼
   41
           int val = (curr.y - prev.y) * (next.x - curr.x) - (curr.x - prev.x)
   42
   43
            if (val == 0)
   44
   45
               return val;
   46
   47
            return (val > 0) ? 1 : -1;
   48
   49
   50
       int main()
   51 ▼ {
   52
            int n = 8;
   53
            struct point p[n + 1];
   54
            for (int i = 0; i < n; i++)
   55 ₹
```

```
56
             scanf("%d%d", &p[i].x, &p[i].y);
57
         }
         int s = 0;
58
         for (int i = 0; i < n; i++)
59
60
             if (p[i].x < p[s].x)</pre>
61
62
             {
                 s = i;
63
64
65
             else if (p[i].x == p[s].x \&\& p[i].y < p[s].y)
66
             {
                 s = i;
67
68
69
70
         // Starting point at 0 index in array
71
         struct point temp1 = p[0];
72
73
         p[0] = p[s];
74
         p[s] = temp1;
75
         for (int i = 0; i < n - 1; i++)
76
77 1
             for (int j = 1; j < n - i - 1; j++)
78
79 1
                 if (slope(p[0], p[j]) > slope(p[0], p[j + 1]))
80
81
                      // printf("Swapping [%d %d]\n(%f) and [%d %d]\n(%f)\n\
82
                      swap(&p[j], &p[j + 1]);
83
84
85
             // printf("***\n");
86
87
88
89
         p[n] = p[0];
         int border[n];
90
91
         border[0] = 0;
92
 93
         int bd index = 1;
 94
 95
         int prev = 0, curr = 1, next = 2;
 96
         while (next < n + 1)
 97
 98
 99
             if (orientation(p[prev], p[curr], p[next]) == -1 || orientatio
100 •
                  // printf("{%d %d %d} Point %d - [%d %d] accepted %d\n", p
101
102
                  border[bd_index] = curr;
103
                  bd_index++;
104
                  prev = curr;
105
                  curr = next;
106
                  next = next + 1;
107
                  continue;
108
             else if (orientation(p[prev], p[curr], p[next]) == 1) // i.e.
109
110 •
```

```
111
                  // printf("{%d %d %d} Point %d - [%d %d] rejected\n", prev
 112
                  curr = prev;
 113
                  bd_index--;
 114
                  prev = border[bd_index - 1];
 115
116
              // printf("%d %d %d <%d>\n", prev, curr, next, orientation(p[p
117
118
119
          printf("The Boundary Coordinates are\n");
 120
          for (int i = bd_index-1; i > 0; i--)
 121 •
122
              printf("%d %d\n", p[border[i]].x, p[border[i]].y);
123
124
          printf("%d %d\n", p[border[0]].x, p[border[0]].y);
125
          return 0;
 126
Check
```

	Test	Input	Expected	Got	
<b>~</b>	1	0 3	The Boundary Coordinates are	The Boundary Coordinates are	~
		1 1	0 3	0 3	
		2 2	4 4	4 4	
		4 4	3 1	3 1	
		0 0	0 0	0 0	
		1 2			
		3 1			
		3 3			

Passed all tests! <

### **Question 2:**

SCOPE / BCSE204P\_VL2023240504902 / ASSESSMENT LAB 4



### **ASSESSMENT LAB 4**

Back

### Question 2

Correct Marked out of 2.50

ℙ Flag question

Write a Program to compute the convex Hull for the given coordinates of the polygon (input) using Jarvis' March Algorithm

Input	Result	
0 3	The Boundary Coordinates are	
2 2	0 3	
1 1	0 0	
2 1	3 0	
3 0	3 3	
0 0		
3 3		
	0 3 2 2 1 1 2 1 3 0 0 0	

```
#include <stdio.h>
    #include <limits.h>
2
3
    #include <float.h>
4
5
    struct point
6 ₹ {
7
        int x;
8
        int y;
9
   };
10
    void swap(struct point *p1, struct point *p2)
11
12 ₹ {
13
        int temp_x = p1->x;
14
        int temp_y = p1->y;
15
        p1->x = p2->x;
16
        p1->y = p2->y;
17
18
19
        p2->x = temp_x;
20
        p2->y = temp_y;
21
   }
22
    int orientation(struct point prev, struct point curr, struct point next
23
24 ▼ {
25
        int val = (curr.y - prev.y) * (curr.x - next.x) - (curr.x - prev.x)
26
27
        if (val == 0)
28 🔻
29
            return val;
30
31
        return (val > 0) ? 1 : -1;
    }
32
33
    int main()
34
35 ₹ {
36
        int n = 7;
        struct point p[n];
37
```

```
38
        for (int i = 0; i < n; i++)
39 1
             scanf("%d%d", &p[i].x, &p[i].y);
40
        }
41
42
        int s = 0;
43
44
        // Finding start point
45
        for (int i = 0; i < n; i++)
46
             if (p[i].x < p[s].x)
47
48 1
             {
49
                 s = i;
50
51
             else if (p[i].x == p[s].x \&\& p[i].y < p[s].y)
52 1
53
                 s = i;
54
55
56
57
        // Starting point at 0 index in array
58
        swap(&p[0], &p[s]);
59
        int border[n];
60
        border[0] = 0;
61
62
        int bd index;
        for (int i = 0; i < n; i++)
63
64
65
            if (i != 0 && border[i] == 0)
66
             {
67
                 bd_index = i;
68
                 break;
69
70
            int prev = border[i];
71
            int next = 1;
72
            for (int j = 0; j < n; j++)
73
74
                 if (orientation(p[prev], p[next], p[j]) == -1)
75
                 {
76
                     next = j;
77
78
79
            border[i + 1] = next;
80
81
82
        printf("The Boundary Coordinates are\n");
        printf("%d %d\n", p[border[bd_index-1]].x, p[border[bd_index-1]].y)
83
        printf("%d %d\n", p[border[0]].x, p[border[0]].y);
84
85
        printf("%d %d\n", p[border[1]].x, p[border[1]].y);
86
        printf("%d %d\n", p[border[2]].x, p[border[2]].y);
87
        return 0;
88
```

	Test	Input	Expected	Got	
<b>~</b>	1	0 3	The Boundary Coordinates are	The Boundary Coordinates are	~
		2 2	0 3	0 3	
		1 1	0 0	0 0	
		2 1	3 0	3 0	
		3 0	3 3	3 3	
		0 0			
		3 3			

Passed all tests! 🗸

### **Question 3:**

SCOPE / BCSE204P\_VL2023240504902 / ASSESSMENT LAB 4



### **ASSESSMENT LAB 4**

Back

#### Question 3

Not complete Marked out of 2.50

Flag question

Write a Program to arrange the given (N) numbers in the ascending order using Randomized Quick Sort Algorithm

Test	Input	Result
1	9 28 13 42 25 11 7 19 56 30	7 11 13 19 25 28 30 42 56
2	10 21 12 62 20 10 9 18 46 33 6	6 9 10 12 18 20 21 33 46 62

```
Answer: (penalty regime: 0, 0, 0, 0, 5, 10, 20, ... %)
```

```
#include <stdio.h>
 1
    #include <stdlib.h>
 2
 3
 4
    void swap(int *a, int *b)
 5 ▼ {
        int temp = *a;
 6
         *a = *b;
 7
 8
         *b = temp;
9
10
    int random_no(int a, int b)
11
12 v {
        int x = b - a > 0? b - a : 1;
13
        return rand() % x + a;
14
15
16
    int partition(int arr[], int low, int high)
17
18 ▼ {
         int piv_index = random_no(low, high + 1);
19
20
         int pivot = arr[piv index];
21
         // pivot = arr[high];
         // printf("low, high, pivot = (%d %d %d)\n", low, high, pivot);
22
        int i = (low - 1);
23
24
         for (int j = low; j <= high; j++)</pre>
25
26 •
             if (arr[j] < pivot)</pre>
27
28 1
29
                 i++;
                 if (i == piv_index)
30
31 •
                     piv_index = j;
32
33
                 swap(&arr[i], &arr[j]);
34
35
36
```

```
// printf("Element %d at position: %d\n", arr[piv_index], i + 1);
38
        swap(&arr[i + 1], &arr[piv_index]);
39
        return (i + 1);
40
41
    void quickSort(int arr[], int low, int high)
42
43 ▼ {
        if (low < high)</pre>
44
45 •
            int pi = partition(arr, low, high);
46
47
            quickSort(arr, low, pi - 1);
48
            quickSort(arr, pi + 1, high);
49
50
51
52
```



# **Question 4:**

SCOPE / BCSE204P\_VL2023240504902 / ASSESSMENT LAB 4



### **ASSESSMENT LAB 4**

Back

#### Question 4

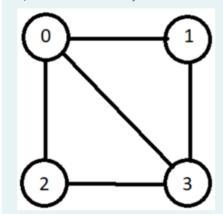
Correct

Marked out of 2.50

ℙ Flag question

Write a Program to apply the Global Minimum Cut Algorithm to

- a) find the contraction edges
- b) Find the cut found by the randomized algorithm



Test	Input	Result
1	Enter the Number of vertices in the graph : 4	Contracting edge 0-1
	Enter the Number of Edges in the graph : 5	Contracting edge 1-3
		Cut found by the randomized alg

```
Answer: (penalty regime: 0, 0, 0, 0, 5, 10, 20, ... %)
       #include <stdio.h>
       #include <stdlib.h>
    2
    3
       int random no(int a, int b)
   4
    5 ▼ {
           int x = b - a > 0? b - a : 1;
    6
    7
           return rand() % x + a;
   8
       }
   9
  10
       struct edge
  11 🔻
  12
           int v1;
  13
           int v2;
  14
       };
  15
       void swap(struct edge *e1, struct edge *e2)
  16
  17 ▼ {
           struct edge e;
  18
  19
           e.v1 = e1->v1;
  20
           e.v2 = e1->v2;
  21
  22
           e1->v1 = e2->v1;
  23
           e1->v2 = e2->v2;
  24
           e2->v1 = e.v1;
  25
  26
           e2->v2 = e.v2;
  27
       }
  28
  29
       struct edge *newEdge(int v1, int v2)
  30 ▼ {
           struct edge *e = (struct edge *)malloc(sizeof(struct edge));
  31
  32
           e \rightarrow v1 = v1;
           e \rightarrow v2 = v2;
  33
  34
           return e;
  35
   36
   37
       int removeSelfLoop(int n_e, struct edge *E[n_e])
   38 🔻
       {
   39
            int n_pe = 0;
   40
            for (int i = 0; i < n_e - n_pe; i++)
  41
                if (E[i]->v1 == E[i]->v2)
   42
   43
                {
                    swap(E[i], E[n_e - 1 - n_pe]);
   44
   45
                    n pe++;
   46
                    i--;
   47
   48
   49
            return n_e - n_pe;
   50
      |}
```

```
51
 52
     int no_of_vert_is_2(int n_e, struct edge *E[n_e])
 53 ▼ {
 54
          int arr[2] = \{-1, -1\};
 55
          for (int i = 0; i < n_e; i++)
 56
              if (E[i]->v1 == arr[0] \mid \mid E[i]->v2 == arr[1])
 57
 58
              {
 59
                   continue;
              }
 60
              else if (arr[0] == -1)
 61
 62
                   arr[0] = E[i] -> v1;
 63
 64
              else if (arr[1] == -1)
 65
 66
                   arr[1] = E[i] -> v1;
 67
              }
 68
 69
              else
 70
              {
 71
                   return 0;
 72
 73
 74
          return 1;
 75
 76
 77
     int main()
 78
 79
          int n_e = 5;
          struct edge *E[5];
 80
          E[0] = newEdge(0, 1);
 81
          E[1] = newEdge(0, 2);
 82
          E[2] = newEdge(0, 3);
 83
          E[3] = newEdge(1, 3);
          E[4] = newEdge(2, 3);
 86
          struct edge *ce = E[0];
 87
          printf("Contracting edge %d-%d\n", ce->v1, ce->v2);
 88
 89
          int a = ce->v1, b = ce->v2;
          for (int i = 0; i < n_e; i++)
 90
 91
 92
              if (E[i]->v1 == a)
 93
               {
 94
                   E[i] \rightarrow v1 = b;
 95
              }
              else if (E[i]->v2 == a)
 96
 97
              {
 98
                   E[i] \rightarrow v2 = b;
 99
100
          n_e = removeSelfLoop(n_e, E);
101
102
          ce = E[3];
103
          printf("Contracting edge %d-%d\n", ce->v1, ce->v2);
104
          for (int i = 0; i < n_e; i++)
105
106
              if (E[i]->v1 == ce->v1)
107
108
              {
                   E[i] \rightarrow v1 = ce \rightarrow v2;
109
110
```

```
111
              else if (E[i]->v2 == ce->v1)
 112 •
 113
                    E[i] \rightarrow v2 = ce \rightarrow v2;
 114
115
116
           ce->v1 = ce->v2;
           n_e = removeSelfLoop(n_e, E);
117
118
           if (no_of_vert_is_2(n_e, E) == 1)
119
 120 •
 121
               printf("Cut found by the randomized algorithm is %d", n_e);
 122
           return 0;
 123
 124 }
Check
```

