ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - IF184802

IMPLEMENTASI REDUKSI POLIGON MENGGUNAKAN ALGORITMA MELKMAN CONVEX HULL YANG DIMODIFIKASI DENGAN STUDI KASUS LL AND ERBAO(ISUN1) PADA SPHERE ONLINE JUDGE

MICHAEL JULIAN ALBERTUS
NRP 05111640000097

Dosen Pembimbing 1
Rully Soelaiman, S.Kom., M.Kom.

Dosen Pembimbing 2
Yudhi Purwananto, S.Kom., M.Kom.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2019

[*Halaman ini sengaja dikosongkan*]

TUGAS AKHIR - IF184802

IMPLEMENTASI REDUKSI POLIGON MENGGUNAKAN ALGORITMA MELKMAN CONVEX HULL YANG DIMODIFIKASI DENGAN STUDI KASUS LL AND ERBAO(ISUN1) PADA SPHERE ONLINE JUDGE

MICHAEL JULIAN ALBERTUS
NRP 05111640000097

Dosen Pembimbing 1
Rully Soelaiman, S.Kom., M.Kom.

Dosen Pembimbing 2
Yudhi Purwananto, S.Kom., M.Kom.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2019

[*Halaman ini sengaja dikosongkan*]

UNDERGRADUATE THESES - IF184802

IMPLEMENTATION OF POLYGON REDUCTION USING MODIFIED MELKMAN CONVEX HULL ALGORITHM WITH CASE STUDY LL AND ERBAO FROM SPHERE ONLINE JUDGE

MICHAEL JULIAN ALBERTUS
NRP 05111640000097

Supervisor 1
Rully Soelaiman, S.Kom., M.Kom.

Supervisor 2
Yudhi Purwananto, S.Kom., M.Kom.

INFORMATICS DEPARTMENT
Faculty of Information Technology and Communication
Institut Teknologi Sepuluh Nopember
Surabaya, 2019

[*Halaman ini sengaja dikosongkan*]

LEMBAR PENGESAHAN

IMPLEMENTASI REDUKSI POLIGON MENGGUNAKAN ALGORITMA MELKMAN CONVEX HULL YANG DIMODIFIKASI DENGAN STUDI KASUS LL AND ERBAO(ISUN1) PADA SPHERE ONLINE JUDGE

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Algoritma Pemrograman
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:
Michael Julian Albertus
NRP. 05111640000097

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Rully Soelaiman, S.Kom., M.Kom. .....

NIP. 19700213199402100 (Pembimbing 1)

Yudhi Purwananto, S.Kom., M.Kom. .....

NIP. 197007141997031002 (Pembimbing 2)

SURABAYA
KAPAN

[*Halaman ini sengaja dikosongkan*]

ABSTRAK

IMPLEMENTASI REDUKSI POLIGON MENGGUNAKAN ALGORITMA MELKMAN CONVEX HULL YANG DIMODIFIKASI DENGAN STUDI KASUS LL AND ERBAO(ISUN1) PADA SPHERE ONLINE JUDGE

Nama	:	Michael Julian Albertus
NRP	:	05111640000097
Departemen	:	Departemen Informatika, Fakultas Teknologi Informasi dan Komunikasi, ITS
Pembimbing I	:	Rully Soelaiman, S.Kom., M.Kom.
Pembimbing II	:	Yudhi Purwananto, S.Kom., M.Kom.

Abstrak

[TBA]

Kata Kunci: geometri; convex hull; melkman algorithm; relative poligon;

[*Halaman ini sengaja dikosongkan*]

ABSTRACT

IMPLEMENTATION OF POLYGON REDUCTION USING MODIFIED MELKMAN CONVEX HULL ALGORITHM WITH CASE STUDY LL AND ERBAO FROM SPHERE ONLINE JUDGE

Name	:	Michael Julian Albertus
Student ID	:	05111640000097
Department	:	Informatics Department, Faculty of Information Technology and Communication, ITS
Supervisor I	:	Rully Soelaiman, S.Kom., M.Kom.
Supervisor II	:	Yudhi Purwananto, S.Kom., M.Kom.

Abstract

[TBA]

Keywords: geometry; convex hull; melkman algorithm; relative poligon;

[*Halaman ini sengaja dikosongkan*]

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa. Atas rahmat dan kasih sayangNya, penulis dapat menyelesaikan tugas akhir dan laporan akhir dalam bentuk buku ini.

Pengerjaan buku ini penulis tujukan untuk mengeksplorasi lebih mendalam topik-topik yang tidak diwadahi oleh kampus, namun banyak menarik perhatian penulis. Selain itu besar harapan penulis bahwa pengerjaan tugas akhir sekaligus pengerjaan buku ini dapat menjadi batu loncatan penulis dalam menimba ilmu yang bermanfaat.

Penulis ingin menyampaikan rasa terima kasih kepada banyak pihak yang telah membimbing, menemani dan membantu penulis selama masa pengerjaan tugas akhir maupun masa studi.

1. Bapak Rully Soelaiman S.Kom.,M.Kom., selaku pembimbing penulis. Ucapan terima kasih juga penulis sampaikan atas segala perhatian, didikan, pengajaran, dan nasihat yang telah diberikan oleh beliau selama masa studi penulis.

Penulis menyadari bahwa buku ini jauh dari kata sempurna. Maka dari itu, penulis memohon maaf apabila terdapat salah kata maupun makna pada buku ini. Akhir kata, penulis mempersembahkan buku ini sebagai wujud nyata kontribusi penulis dalam ilmu pengetahuan.

Surabaya, KAPAN

Michael Julian Albertus

[*Halaman ini sengaja dikosongkan*]

DAFTAR ISI

DAFTAR GAMBAR

[*Halaman ini sengaja dikosongkan*]

DAFTAR TABEL

DAFTAR PSEUDOCODE

[*Halaman ini sengaja dikosongkan*]

DAFTAR KODE SUMBER

[*Halaman ini sengaja dikosongkan*]

DAFTAR NOTASI

$\mathbb{Z}$	Himpunan bilangan bulat.
$\mathbb{Z}_n$	Himpunan bilangan bulat positif hingga n eksklusif.
$\mathbb{Z}/p\mathbb{Z}$	Himpunan kongruensi dalam modulo p, dimana p merupakan bilangan prima dalam *multiplicative group*.
$\phi(n)$	*Euler Totient Function* atau *Euler Phi*. Menotasikan banyaknya nilai yang koprima dengan n.
$R[x]/R$	Himpunan *ring* yang merupakan struktur aljabar bilangan pada pertambahan dan perkalian.

[*Halaman ini sengaja dikosongkan*]

BAB I
DASAR TEORI

Pada bab ini, akan dijelaskan dasar teori yang digunakan sebagai landasaan pengerjaan Tugas Akhir ini.

1.1. Deskripsi Permasalahan

Permasalahan yang dibahas pada Tugas Akhir ini adalah perhitungan untuk mencari nilai x yang didefinisikan oleh persamaan (1.1).

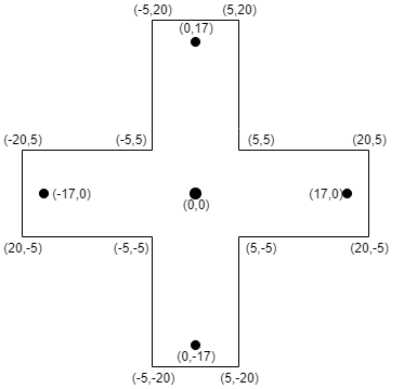
$$x = \sum_{i=0}^{n-1} \mathrm{RCH}_i \tag{1.1}$$

RCH_i pada persamaan (1.1) menyatakan sisi polygon dari RCH yang merupakan *relative convex hull* yang didapatkan dari sekumpulan titik yang dibatasi di dalam polygon sederhana[1]. Permasalahan pada tugas akhir ini adalah mencari *relative convex hull* dari sekumpulan titik yang dibatasi oleh polygon sederhana. Gambar 1.1 dan 1.2 merupakan contoh dari permasalahan ISUN1.

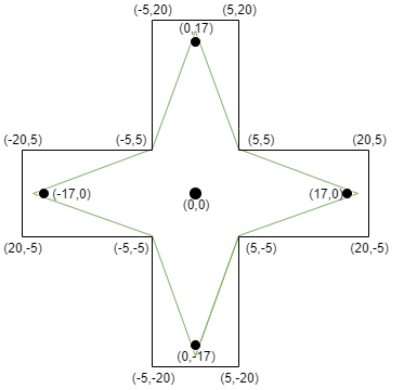
1.2. Convex Polygon

Convex polygon merupakan sebuah polygon sederhana yang memiliki sudut maksimal 180 derajat pada tiap edgenya. *Convex polygon* memiliki beberapa properti, yaitu:

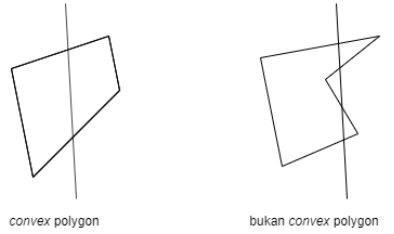
1. Sebuah garis lurus yang di gambar melewati sebuah *convex* polygon akan berpotongan maksimal 2 kali. Ilustrasi dapat dilihan pada gambar 1.3
2. Jika dua titik sembarang diambil dan ditarik garis antara keduanya, tidak ada bagian dari garis yang berada di luar polygon. Ilustrasi dapat dilihat pada gambar 1.4



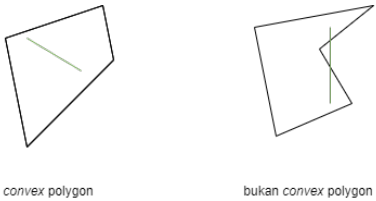
Gambar 1.1: Ilustrasi Contoh Kasus Tanpa Solusi



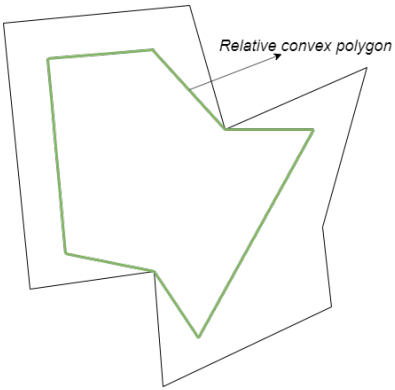
Gambar 1.2: Ilustrasi Contoh Kasus



Gambar 1.3: Ilustrasi Properti Convex Polygon 1



Gambar 1.4: Ilustrasi Properti Convex Polygon 2



Gambar 1.5: Ilustrasi Relative Convex Polygon

1.2.1. Relative Convex Polygon

Relative convex polygon merupakan penurunan dari convex Polygon tetapi ada beberapa sisi dari polygon tersebut berbentuk convace atau cekung kedalam dikarenakan adanya batasan dari luar seperti polygon atau segmen garis lainnya. Ilustrasi relative convex polygon dapat dilihat pada gambar 1.5.

1.3. Strategi Penyelesaian Permasalahan

Pada subbab ini akan dipaparkan mengenai strategi penyelesaian masalah klasik pada daring SPOJ dengan kode ISUN1 menggunakan algoritma reduksi polygon. Secara singkat, strategi penye-

lesaian masalah dari ISUN1 menggunakan algoritma reduksin polygon menjadi 2 bagian besar, yaitu :

1. Pemrosesan titik pembentuk polygon yang membentuk *Convex*.
2. *Convex Hull* dari titik yang berada didalam polygon

Sebagai contoh, pada subbab ini akan menggunakan P sebagai polygon luar yang mempunyai n vertex, dimana $P = \langle p_1, p_2, ..., p_n \rangle$ yang mempunyai titik sebanyak m ($S = \langle s_1, s_2, ..., s_m \rangle$), dan $D(A)$ merupakan sebuah deque (*doubly-ended queue*) yang menampung vertex dari polygon P. Reduksi polygon didasari dari algoritma Melkman dengan sedikit modifikasi. Modifikasi yang dilakukan adalah ketika 3 buak titik pembentuk polygon yang konsekutif membuat *convex* maka titik tengan dari ketiga titik tersebut dibuang, dan jika *concave* maka titik tengahnya tetap disimpan. Pada saat sebuah titik dibuang, maka luas dari polygon akan tereduksi. Langkah-langkah reduksi dilakukan dengan mengulangi 2 langkah yang akan dijelaskan pada subbab 1.3.1 dan 1.3.2.

1.3.1. Pemrosesan Titik Pembentuk Polygon yang Membentuk Convex

Pemrosesan titik pembentuk polygon dapat dilakukan dengan cara melakukan *traversing* terhadap semua vertex pembentuk polygon. Untuk setiap vertex p_i yang di cek, hitung orientasi(secara berlawanan jarum jam) titik p_i dengan p_{i-1} dan p_{i+1}. Jika orientasinya membentuk *convex*, maka titik p_i akan dibuang.

Sebelum membuang titik p_i, kita akan membuat sebuah segitiga ABC dimana $A = p_i$, $B = p_{i-1}$, dan $C = p_{i+1}$ karena triangulation of polygon(Teorema 1).

Teorema 1 (Triangulation of Polygon) *Semua polygon dapat di buat dari beberapa segitiga.*

Kemudian cari $T(ABC)$ dimana $T(ABC)$ merupakan semua titik S yang berada di dalam segitiga ABC dengan menggunakan algoritma *Point Inside Polygon* (dapat dilihat pada subbab 1.5). Pencarian titik yang berada di dalam segitiga ABC berguna untuk mencari pengganti vertex p_i sebagai pembentuk polygon luarnya.

1.3.2. Convex Hull dari Titik yang Berada di Dalam Polygon

Melanjutkan dari subbab 1.3.1, ketika sudah mendapatkan T, lakukan pencarian *Convex Hull* dari titik - titik tersebut menggunakan algoritma *monotone chain* (dapat dilihat pada subbab 1.4.2.2). Kemudian sisipkan semua titik yang membentuk *Convex Hull* diantara vertex p_{i-1}, p_{i+1} untuk me-rekonstruksi polygon luar yang sudah di reduksi.

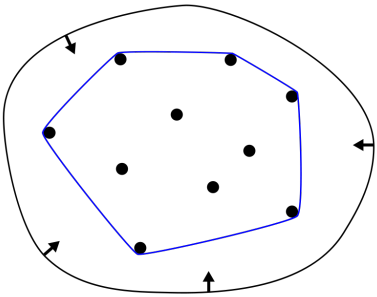
1.4. Convex Hull

Convex Hull dari sekumpulan titik S adalah sebuah set dari semua kombinasi *convex* dari titik - titik tersebut. Setiap titik s_i pada S diberikan sebuah koefisien a_i dimana a_i merupakan bialangan non negatif dan jika semua a_i dijumlahkan hasilnya satu. Dan koefisien ini digunakan untuk menghitung berat rata - rata untuk setiap titik. Untuk setiap koefisien yang dipilih akan dikombinasikan dan menghasilkan *convex hull*. Set *convex hull* ini dapat di ekspresikan dengan formula (1.2) dan ilustrasi *convex hull* ada pada gambar 1.6.

$$Conv(S) = \left\{ \sum_{i=1}^{|S|} a_i s_i \Big| (\forall i : a_i \geq 0 \wedge \sum_{i=1}^{|S|} a_i = 1) \right\} \tag{1.2}$$

1.4.1. Relative Convex Hull

Relative convex hull merupakan penurunan dari *convex hull*. *Relative convex hull* merupakan *convex hull* yang mempunyai



Gambar 1.6: Ilustrasi Convex Cull

cavity (cekungan kedalam) yang diakibatkan atau relatif terhadap sesuatu yang membatasi *convex hull tersebut*. ilustrasi *relative convex hull* dapat dilihat pada gambar 1.7.

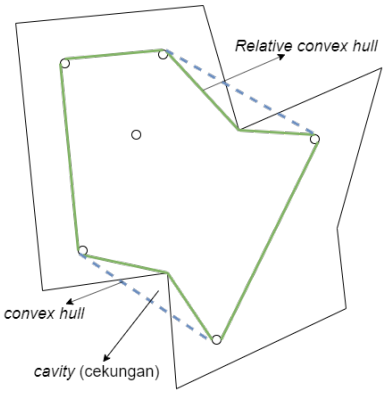
Penentuan untuk mengetahui sebuah polygon merupakan *convex* atau *concave* dapat menggunakan orientasi. Apabila orientasi dari tiga titik yang berurutan adalah positif berlawanan jarum jam maka tiga titik tersebut adalah *convex*. Sebaliknya apabila negatif maka tiga titik tersebut adalah *concave*. Untuk mencari orientasi antara tiga titik dapat digunakan persamaan 1.3.

$$\begin{aligned} \vec{u} &= (B_x - A_x)x + (B_y - A_y)y \\ \vec{v} &= (C_x - A_x)x + (C_y - A_y)y \end{aligned} \tag{1.3}$$

$$Orientasi = u_x * v_y - u_y * v_x$$

1.4.2. Algoritma Convex Hull

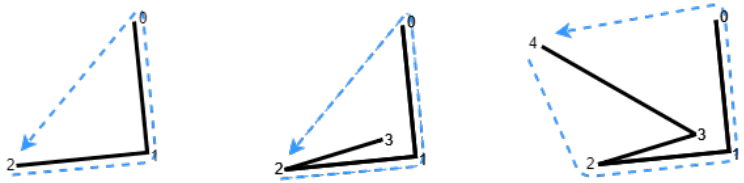
Ada beberapa algoritma yang dapat digunakan untuk mencari sebuah *convex hull*, untuk melihat perbandingan dari beberapa algoritma dapat dilihat pada tabel 1.1. Berdasarkan tabel 1.1, penulis memilih 2 algoritma yang akan digunakan pada buku ini.



Gambar 1.7: Ilustrasi Relative Convex Hull

Tabel 1.1: Tabel Perbandingan Algoritma Convex Hull

Algoritma Convex Hull	Implementasi	Kompleksitas	Kode Sumber	Jenis Input
Jarvis's Algorithm	Mudah	$\mathcal{O}(n^2)$	Singkat	Kumpulan Titik
Graham's Scan	Sedikit Mudah	$\mathcal{O}(n \log(n))$	Singkat	Kumpulan Titik
Quick Hull	Kompleks	$\mathcal{O}(n \log(n))$	Panjang	Kumpulan Titik
Monotone Chain	Mudah	$\mathcal{O}(n \log(n))$	Singkat	Kumpulan Titik
Melkman's Algorithm	Mudah	$\mathcal{O}(n)$	Singkat	polygon Sederhana



Gambar 1.8: Ilustrasi Algoritma Melkman

1.4.2.1. Algoritma Melkman Convex Hull

Algoritma melkman *convex hull* merupakan algoritma untuk menghitung rantai polygonal ataupun polygon sederhana dengan waktu linear $\mathcal{O}(n)$[2]. Asumsikan sebuah polygon sederhana P, dengan vertex p_i dan edge $p_i p_{i+1}$. Algoritma ini menggunakan deque, $D = \langle d_1, d_2, ..., d_n \rangle$, untuk mereprentasikan *convex hull* , $Q_i = CH(P_i)$, dimana $P_i = (p_0, p_1, ..., p_i)$. Deque mempunyai fungsi *push* dan *pop* dari atas/depan dan *insert* dan *remove* dari bawah/belakang. Secara spesifiknya yang dilakukan *push* v ke deque melakukan $(l \leftarrow l+1; d_t \leftarrow v)$, untuk *pop* d_t dari deque melakukan $(t \leftarrow t-1)$, untuk insert v ke deque melakukan $(b \leftarrow b-1; d_b \leftarrow v)$, dan *remove* d_b dari deque melakukan $(b \leftarrow b+1)$.

Algoritma ini menggunakan konvensi dimana vertexnya berurutan secara berlawanan jarum jam di sekitar *convex hull* Q.

Setiap d_t dan d_b mengacu kepada vertex yang sama pada rantai polygon C, dan vertex ini akan selalu menjadi vertex yang kita tambahkan terakhir pada *convex hull* . Pseudocode Melkman Convex Hull dapat dilihat pada pseudocode 1.1.

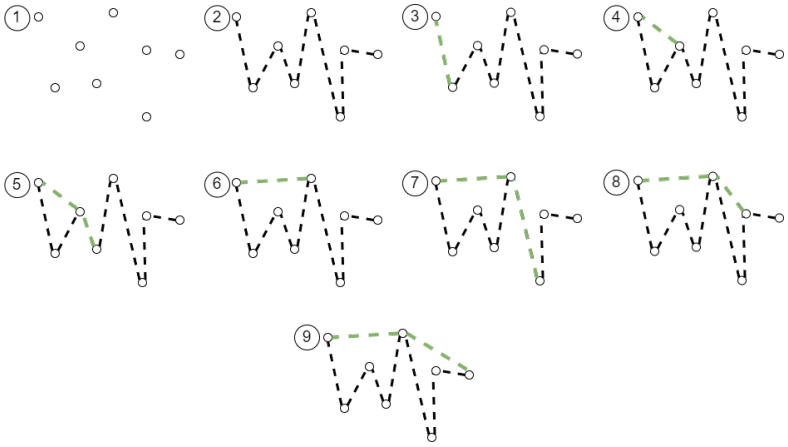
1.4.2.2. Algoritma Monotone Chain

Algoritma *monotone chain* merupakan proses pembentukan *convex hull* dari sekumpulan titik dengan kompleksitas $\mathcal{O}(n \log(n))$[3]. Asumsikan sekumpulan titik S sejumlah n ,$S = \langle s_1, s_2, ..., s_n \rangle$ algoritma ini menggunakan list untuk membentuk

Pseudocode 1.1: Melkman Convex Hull

Input: P
Output: Q

Inisialisasi: D
if LEFT(p_0, p_1, p_2) **then**
    $D \leftarrow \langle p_2, p_0, p_1, p_2 \rangle$
else
    $D \leftarrow \langle p_2, p_1, p_0, p_2 \rangle$
end if
$i = 3$
while $i < n$ **do**
    while LEFT(d_{t-1}, d_t, p_i) dan LEFT(d_b, d_{b+1}, p_i)) **do**
        $i \leftarrow i + 1$
    end while
    while !LEFT(d_{t-1}, d_t, p_i) **do**
        pop d_t
    end while
    push p_i
    while !LEFT(p_i, d_b, d_{b+1}) **do**
        remove d_b
    end while
    insert p_i
    $i \leftarrow i + 1$
end while



Gambar 1.9: Ilustrasi Algoritma Monotone Chain

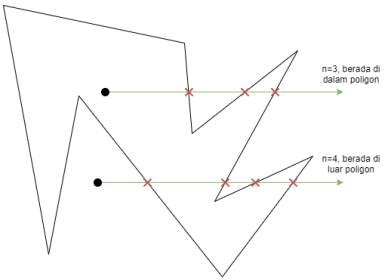
sebuah rantai (*monotone chain*), dimana list $L(S)$ menampung semua titik yang ada di S yang terurut berdasarkan nilai koordinatnya terhadap sumbu x. algoritma ini memeriksa setiap 3 vertex yang berurutan, jika 3 vertex tersebut membuat *convex* makan ketiga vertex tersebut disimpan, dan sebaliknya jika ketiga vertex tersebut membuat *concave* maka vertex ke 2 akan dibuang dari vertex penyusun *convex hull*. lalu lakukan hal yang sama dengan membalikan urutan pada L untuk mendapatkan *lower hull*. Pseudocode algoritma *Monotone Chain* dapat dilihat pada pseudocode 1.2.

1.5. Point Inside Polygon

Point Inside Polygon merupakan algoritma untuk menentukan apakah suatu polygon berada di dalam sebuah polygon atau tidak [4]. Ide utama dari algoritma ini adalah dengan cara menarik garis sejajar dengan sumbu x dimana garis tersebut berujung pada titik yang ingin dicari lokasinya kemudian hitung ada berapa edge dari polygon yang berpotongan dengan garis tersebut. Jika jumlah edge polygon yang berpotongan adalah ganjil, maka titik tersebut bera-

Pseudocode 1.2: Monotone Chain Algorithm

Input: S
Output: $CH(S)$
1: Inisialisasi: L
2: Sort S
3: $L \leftarrow S$
4: Inisialisasi $CH(S)$
5: **for** $i = 0; i < 2; i++$ **do**
6: **for** $j = 0; j < Size(L); j++$ **do**
7: **while** $Size(CH) \geq 2$ and $right(CH[Size(CH) - 1], CH[Size(CH) - 2], S[j])$ **do**
8: Delete CH last element
9: **end while**
10: push pt to CH
11: **end for**
12: reverse L
13: **end for**



Gambar 1.10: Ilustrasi Algoritma Point Inside Polygon

da dalam polygon, dan sebaliknya, jika jumlahnya genap maka titik tersebut berada di luar polygon.

BAB II
DESAIN

Pada bab ini akan dijelaskan desain algoritma yang akan digunakan untuk menyelesaikan permasalahan.

2.1. Desain Umum Sistem

Pada subbab ini akan dijelaskan mengenai gambaran secara umum dari algoritma yang dirancang. Sistem diawali dengan menerima masukan 2 buah bilangan bulat N yang merupakan banyaknya vertex pembentuk poligon luar dan M yang merupakan banyaknya titik yang ada di dalam poligon tersebut. N baris berikutnya berisikan 2 buah bilangan bulat x_i, y_i yang merupakan koordinat dari vertex pembentuk poligon luar terurut berlawanan arah jarum jam. M baris berikutnya berisikan dua buah bilangan bulat $x1_i$, $y1_i$ yang merupakan koordinat dari titik yang ada di dalam poligon.

2.2. Desain Fungsi Main

Fungsi MAIN merupakan fungsi yang bertanggung jawab untuk menerima masukan yang sudah dijelaskan pada 2.1 untuk dilakukan proses sekanjutnya. Pseudocode fungsi MAIN dapat dilihat pada pseudocode 2.1. Fungsi INPUT merupakan fungsi untuk menerima masukan, dan fungsi PRINT merupakan fungsi untuk menampilkan hasil.

2.3. Desain Class Point

Class POINT adalah class untuk menyimpan titik dalam diagram Kartesius. Pseudocode 2.2 merupakan pseudocode dari class POINT. Nantinya pada implementasi, class ini akan melakukan *override* terhadap operator perbandingan.

Pseudocode 2.1: Fungsi MAIN

```
1:  while (N ← INPUT()) and N ≠ EOF do
2:      M ← INPUT()
3:      perimeter ← POLYGON
4:      trees ← Array POINT
5:      for i ← 1, N do
6:          x_i, y_i ← INPUT()
7:          perimeter.P[i] ← POINT(x_i, y_i, false)
8:      end for
9:      if M = 1 or M = 0 then
10:         PRINT(0)
11:         CONTINUE
12:     end if
13:     for i ← 1, M do
14:         x1_i, y1_i ← INPUT()
15:         trees ← POINT(x1_i, y1_i, true)
16:     end for
17:     ans ←SOLVE(perimeter, trees)
18:     PRINT (ans)
19: end while
```

Nama Variabel	Fungsi Variabel
x	Menyimpan ordinat dari titik tersebut
y	Menyimpan absis dari titik tersebut
$fixed$	untuk membedakan antara titik pembentuk poligon P dan titik yang ada di dalam kumpulan titik S

Tabel 2.1: Nama dan Fungsi Variabel dalam class POINT

Pseudocode 2.2: Class POINT

1: $x, y \leftarrow$ **double**
2: $fixed \leftarrow$ **boolean**
3: **constructor** POINT()
4: **constructor** POINT($\_x, \_y, \_fixed$)

Nama Variabel	Fungsi Variabel
x	Menyimpan arah vektor sejajar dengan sumbu x
y	Menyimpan arah vektor sejajar dengan sumbu y

Tabel 2.2: Nama dan Fungsi Variabel dalam class VEC

Class POINT tidak memiliki fungsi karena class ini memang hanya untuk menyimpan suatu titik yang akan digunakan nanti.

Fungsi *Constructor* dari class ini terdiri dari dua jenis. Fungsi *constructor* yang pertama adalah fungsi dengan tanpa parameter, pada *constructor* ini, semua variabel yang ada di dalam class POINT akan di set 0. Fungsi *constructor* kedua adalah fungsi dengan parameter $\_x, \_y, \_fixed$, menyatakan nilai $x, y, fixed$ secara berurutan.

2.4. Desain Class Vec

Class VEC merupakan class yang menyimpan vector dari dua buah titik pada diagram kartesian. Pseudocode 2.3 merupakan pseudocode dari class VEC.

Class VEC tidak memiliki fungsi karena class ini hanya untuk menyimpan vector dari dua titik yang akan digunakan nanti.

Fungsi *Constructor* dari class ini terdiri dari 3 jenis. Fungsi *constructor* yang pertama adalah fungsi dengan tanpa parameter, pada *constructor* ini, semua variabel yang ada di dalam class VEC

Pseudocode 2.3: Class VEC

```
1: x, y ← double
2: constructor VEC()
3: constructor VEC(_x, _y)
4: constructor VEC(A, B)
```

Nama Variabel	Fungsi Variabel
a	Menyimpan nilai a pada persamaan $ax+by+c=0$
b	Menyimpan nilai b pada persamaan $ax+by+c=0$
c	Menyimpan nilai c pada persamaan $ax+by+c=0$

Tabel 2.3: Nama dan Fungsi Variabel dalam class LINE

akan di set 0. Fungsi *constructor* kedua adalah fungsi dengan parameter $\_x, \_y$, menyatakan nilai x, y secara berurutan. Fungsi *constructor* ketiga adalah fungsi dengan parameter A, B, menyatakan POINT dari titik A dan POINT dari titik B, dimana nantinya nilai x dan y akan didapatkan dari pengurangan koordinat dari POINT A dan POINT B.

2.5. Desain Class Line

Class LINE merupakan class yang bertanggung jawab untuk melakukan operasi-operasi pada garis dalam diagram kartesian. Pseudocode 2.4 merupakan pseudocode dari class LINE.

Class LINE tidak memiliki fungsi karena class ini hanya untuk menyimpan nilai dari fungsi $ax + by + c = 0$ yang akan digunakan nanti.

Fungsi *Constructor* dari class ini terdiri dari 3 jenis. Fungsi *constructor* yang pertama adalah fungsi dengan tanpa parameter,

Pseudocode 2.4: Class LINE

```
1: a, b, c ← double
2: constructor LINE()
3: constructor LINE(_a, _b, _c)
4: constructor LINE(A, B)
```

Nama Variabel	Fungsi Variabel
P	Menyimpan POINT yang merupakan ujung awal dari sebuah segment garis
Q	Menyimpan POINT yang merupakan ujung akhir dari sebuah segment garis
L	Menyimpan fungsi dari garis yang melalui dua titik tersebut

Tabel 2.4: Nama dan Fungsi Variabel dalam class SEGMENT

pada *constructor* ini, semua variabel yang ada di dalam class LINE akan di set 0. Fungsi *constructor* kedua adalah fungsi dengan parameter $\_a, \_b, \_c$, menyatakan nilai a, b, c secara berurutan. Fungsi *constructor* ketiga adalah fungsi dengan parameter A, B, menyatakan POINT dari titik A dan POINT dari titik B, dimana nantinya nilai a, b dan c akan didapatkan dengan mencari fungsi garis yang melewati POINT A dan POINT B.

2.6. Desain Class Segment

Class SEGMENT merupakan class yang bertanggung jawab untuk menyimpan dan melakukan operasi-operasi pada segment garis dalam diagram kartesian. Pseudocode 2.5 merupakan pseudocode dari class SEGMENT.

Class SEGMENT tidak memiliki fungsi karena class ini hanya untuk menyimpan data dari sebuah segmen garis yang akan digunakan nanti.

Pseudocode 2.5: Class SEGMENT

```
1: P, Q ← POINT
2: L ← LINE
3: constructor SEGMENT()
4: constructor SEGMENT(_P, _Q)
```

Nama Variabel	Fungsi Variabel
P	Menyimpan array dari POINT yang membentuk poligon tersebut

Tabel 2.5: Nama dan Fungsi Variabel dalam class POLYGON

Fungsi *Constructor* dari class ini terdiri dari 2 jenis. Fungsi *constructor* yang pertama adalah fungsi dengan tanpa parameter, pada *constructor* ini, semua variabel yang ada di dalam class SEGMENT akan di set 0. Fungsi *constructor* kedua adalah fungsi dengan parameter $\_P, \_Q$, menyatakan POINT dari titik P dan POINT dari titik Q, yang merupakan titik POINT A dan POINT B secara berturut, dan LINE L didapar dengan menggunakan *constructor* LINE dengan parameter P dan Q.

2.7. Desain Class Polygon

Class POLYGON merupakan class yang bertanggung jawab untuk menyimpan dan melakukan operasi-operasi pada poligon pada diagram kartesian. Pseudocode 2.6 merupakan pseudocode dari class POLYGON.

Fungsi-fungsi yang terkandung dalam class ini adalah PREV, NEXT, PERIMETER. Tabel 2.5 menjelaskan variabel dan kegunaannya dalam class POLYGON.

Fungsi *Constructor* dari class ini terdiri dari 2 jenis. Fungsi *constructor* yang pertama adalah fungsi dengan tanpa parameter, pada *constructor* ini, variabel P yang ada di dalam class POLYGON

Pseudocode 2.6: Class POLYGON

```
1: P ← Array POINT
2: constructor POLYGON()
3: constructor POLYGON(_P)
4: function PREV(idx)
5: function NEXT(idx)
6: function PERIMETER()
```

akan di inisialisasi. Fungsi *constructor* kedua adalah fungsi dengan parameter $\_P$, menyatakan array POINT dari titik pembentuk poligon tersebut.

Masukan	Proses	Keluaran
Suatu bilangan bulat idx yang menyatakan index saat ini	mencari index selanjutnya	Suatu bilangan bulat yang menyatakan index selanjutnya

Tabel 2.6: Masukan, Proses, dan Keluaran dari Fungsi NEXT Class POLYGON

Fungsi *next* bertanggung jawab untuk mencari index selanjutnya dari titik yang membentuk polygon. Masukan, proses dan keluaran dari fungsi ini tercantum pada tabel 2.6. Pseudocode fungsi ini dapat dilihat pada pseudocode 2.7.

Pseudocode 2.7: Fungsi NEXT pada class POLYGON

Input: idx

```
1: if idx =SIZE(P)−1 then
2:     return 0
3: else
4:     return idx + 1
5: end if
```

Masukan	Proses	Keluaran
Suatu bilangan bulat idx yang menyatakan index saat ini	mencari index sebelumnya	Suatu bilangan bulat yang menyatakan index sebelumnya

Tabel 2.7: Masukan, Proses, dan Keluaran dari Fungsi PREV Class POLYGON

Fungsi *prev* bertanggung jawab untuk mencari index sebelumnya dari titik yang membentuk polygon. Masukan, proses dan keluaran dari fungsi ini tercantum pada tabel 2.7. Pseudocode fungsi ini dapat dilihat pada pseudocode 2.8.

Pseudocode 2.8: Fungsi PREV pada class POLYGON

Input: idx
1: **if** $idx = 0$ **then**
2: **return** SIZE(P)-1
3: **else**
4: **return** $idx - 1$
5: **end if**

Masukan	Proses	Keluaran
-	Mencari keliling dengan mencari jarak eulerian dari semua titik pembentuk polygon	Suatu bilangan berkoma yang menyatakan keliling dari poligon tersebut

Tabel 2.8: Masukan, Proses, dan Keluaran dari Fungsi PERIMETER Class POLYGON

Fungsi *perimeter* bertanggung jawab untuk mencari keliling dari sebuah polygon. Masukan, proses, dan keluaran dari fungsi ini tercantum pada tabel 2.8. Pseudocode funsi ini dapat dilihat pada pseudocode 2.9.

Pseudocode 2.9: Fungsi PERIMETER pada class POLYGON

```
1: ret ← 0
2: for i ← 0 to SIZE(P)−1 do
3:     ret ← ret EDIST(P[i], P[ NEXT(i)])
4: end for
5: return ret
```

Masukan	Proses	Keluaran
tiga buah bilangan x, l, r, dimana bilangan x merupakan bilangan yang ingin diketahui apakah berada diantara titik l dan r	Mencari tahu apakan bilangan x berada diantara bilangan l dan r	Sebuah boolean yang menyatakan apakah x berada diantara l dan r

Tabel 2.9: Masukan, Proses, dan Keluaran dari Fungsi BETWEEND

2.8. Fungsi BetweenD

Fungsi BETWEEND bertanggung jawab untuk mencari tahu apakah suatu bilangan x berada diantara bilangan l dan bilangan$r$. Pseudocode dari fungsi BETWEEND dapat dilihat pada pseudocode 2.10. Masukan, proses, dan keluaran dari fungsi ini tercantum pada tabel 2.9.

Pseudocode 2.10: Fungsi BETWEEND

Input: x, l, r

```
1: if MIN(l, r) ≤ x + EPS and x ≤ MAX(l, r) + EPS then
2:     return TRUE
3: else
4:     return FALSE
5: end if
```

Masukan	Proses	Keluaran
Dua buah POINT A dan POINT B yang akan dicari jaraknya	Mencari jarak antara POINT A dan POINT B	Sebuah bilangan berkoma yang menyatakan jarak POINT A dan POINT B

Tabel 2.10: Masukan, Proses, dan Keluaran dari Fungsi EDIST

2.9. Fungsi EDist

Fungsi EDIST bertanggung jawab untuk mencari jarak antara dua titik POINT A dan POINT B dengan menggunakan rumus Pythagoras. Rumus Pythagoras dapat di lihat pada persamaan 2.1. Pseudocode fungsi EDIST dapat dilihat pada pseudocode 2.11. Masukan, proses, dan keluaran dari fungsi ini tercantum pada tabel 2.10.

$$C = \sqrt{A^2 + B^2} \tag{2.1}$$

Pseudocode 2.11: Fungsi EDIST

Input: A, B

```
1: return SQRT((A * A) + (B * B))
```

2.10. Fungsi Cross

Fungsi CROSS bertanggung jawab untuk mencari nilai perkalian *cross* dari dua buah vector. Rumus Pythagoras dapat di lihat pada persamaan 2.2. Pseudocode fungsi CROSS dapat dilihat pada pseudocode 2.12. Masukan, proses, dan keluaran dari fungsi ini tercantum pada tabel 2.11.

$$C = (u_x * v_y) - (u_y * v_x) \tag{2.2}$$

Masukan	Proses	Keluaran
Dua buah VEC U dan VEC V yang akan dicari hasil perkalian silangnya	Mencari nilai perkalian silang dari VEC U dan VEC V	Sebuah bilangan yang menyatakan hasil perkalian silang VEC U dan VEC V

Tabel 2.11: Masukan, Proses, dan Keluaran dari Fungsi CROSS

Pseudocode 2.12: Fungsi CROSS

Input: U, V

1: **return** $(U.x * V.y) - (U.y * V.x)$

2.11. Fungsi Orientation

Fungsi ORIENTATION bertanggung jawab untuk mencari orientasi dari tiga titik. Pseudocode fungsi ORIENTATION dapat dilihat pada pseudocode 2.13. Masukan, proses, dan keluaran dari fungsi ini tercantum pada tabel 2.12.

2.12. Fungsi OnSegment

Fungsi ONSEGMENT bertanggung jawab untuk mencari tahu apakah sebuah titik POINT P bersentuhan dengan sebuah segment

Masukan	Proses	Keluaran
Tiga buah POINT O, POINT P dan POINT Q yang akan dicari orientasinya	Mencari orientasi antara POINT O, POINT P dan POINT Q	Sebuah bilangan yang menyatakan orientasi antara POINT O, POINT P dan POINT Q

Tabel 2.12: Masukan, Proses, dan Keluaran dari Fungsi ORIENTATION

Pseudocode 2.13: Fungsi ORIENTATION

Input: O, P, Q
1: $OP \leftarrow$ VEC(O, P)
2: $OQ \leftarrow$ VEC(O, Q)
3: **return** CROSS(OP, OQ)

Masukan	Proses	Keluaran
Dua buah POINT P dan SEGMENT S yang akan dicari tahu apakah POINT P berada di SEGMENT S	Mencari tahu apakah POINT P berada di SEGMENT S	Sebuah boolean yang menyatakan apakah POINT P berada di dalam SEGMENT S

Tabel 2.13: Masukan, Proses, dan Keluaran dari Fungsi ONSEGMENT

garis SEGMENT S atau tidak. Pseudocode fungsi ONSEGMENT dapat dilihat pada pseudocode 2.14. Masukan, proses, dan keluaran dari fungsi ini tercantum pada tabel 2.13.

Pseudocode 2.14: Fungsi ONSEGMENT

Input: P, S
1: **if** ORIENTATION$(S.P, S.Q, P)$ **then**
2: **return** FALSE
3: **else**
4: **return** (BETWEEND$(P.x, S.P.x, S.Q.x)$ and BETWEEND$(P.y, S.P.y, S.Q.y)$)
5: **end if**

Masukan	Proses	Keluaran
Sebuah array POINT pts yang merupakan sekumpulan titik	Mencari POLYGON dengan keliling terkecil dari sekumpulan titik	Sebuah POLYGON yang mengelilingin kumpulan POINT pts

Tabel 2.14: Masukan, Proses, dan Keluaran dari Fungsi CONVEXHULL

2.13. Fungsi ConvexHull

Fungsi CONVEXHULL bertanggung jawab untuk mencari *convex hull* dari sekumpulan titik pts. Algoritma yang digunakan oleh fungsi ini adalah algoritma *Monotone Chain* yang dapat dilihat pada bagian 1.4.2.2. Pseudocode dari fungsi CONVEXHULL yang digunakan dapat dilihat pada Pseudocode 2.15. Masukan, proses, dan keluaran dari fungsi ini tercantum pada tabel 2.14.

2.14. Fungsi InSimplePolygon

Fungsi INSIMPLEPOLYGON bertanggung jawab untuk mencari tahu apakah sebuah titik POINT P berada di dalam POLYGON A atau tidak. Algoritma yang digunakan pada fungsi ini adalah algoritma *point inside polygon* yang dapat dilihat pada bagian 1.5. Pada desain fungsi INSIMPLEPOLYGON ada 3 macam keluaran yaitu -1 untuk menandakan bahwa POINT P berada di dalam POLYGON A, 0 untuk menandakan bahwa POINT P berada di salah satu sisi POLYGON A, dan 1 untuk menandakan bahwa POINT P berada di luar POLYGON A. Pseudocode fungsi INSIMPLEPOLYGON dapat dilihat pada pseudocode 2.16. Masukan, proses, dan keluaran dari fungsi ini tercantum pada tabel 2.15.

Pseudocode 2.15: Fungsi CONVEXHULL

Input: pts

```
1: SORT(pts)
2: hull ← Array POINT
3: for i ← 0 to 2 do
4:     start ← SIZE(hull)
5:     for pt in pts do
6:         while (SIZE(hull)≥ start + 2) and (ORIENTATION(hull[SIZE(hull)−1], hull[SIZE(hull)−2], pt)≤ 0) do
7:             hull.ERASE(hull[SIZE(hull)−1])
8:         end while
9:         hull.INSERT(pt)
10:    end for
11:    hull.ERASE(hull[SIZE(hull)−1])
12:    REVERSE(pts)
13: end for
14: return POLYGON(hull);
```

Masukan	Proses	Keluaran
Sebuah POINT A dan sebuah POLYGON P yang akan dicari tahu apakah POINT A berada di dalam POLYGON P	Mencari tahu apakah POINT A berada di dalam POLYGON P	Sebuah bilangan yang apakah POINT A berada di dalam POLYGON P

Tabel 2.15: Masukan, Proses, dan Keluaran dari Fungsi INSIMPLEPOLYGON

Pseudocode 2.16: Fungsi INSIMPLEPOLYGON

Input: P, A

```
1:  ret ← INTEGER
2:  for i ← 0 to SIZE(A.P) do
3:      if P = A.P[i] then
4:          return 0
5:      end if
6:      j ← A.NEXT(i)
7:      if ONSEGMENT(P, SEGMENT(A.P[i], A.P[j])) then
8:          return 0
9:      end if
10:     below ← (A.P[i].y < P.y)
11:     if below ≠ (A.P[j].y < P.y) then
12:         o ← ORIENTATION(P, A.P[i], A.P[j])
13:         if o = 0 then
14:             return 0
15:         end if
16:         if below = (o > 0) and below = TRUE then
17:             ret+ = 1
18:         else
19:             if below = (o > 0) and below = FALSE then
20:                 ret− = 1
21:             end if
22:         end if
23:     end if
24: end for
25: if ret = 0 then
26:     return 1
27: else
28:     return −1
29: end if
```

Masukan	Proses	Keluaran
Sebuah POLYGON $triangle$ yang merupakan segitiga, QUEUE POINT q yang merupakan pembentuk polygon luar, dan array POINT $trees$ yang merupakan titik yang berada di dalam polygon luar	Mencari POINT mana saja yang akan menggantikan POINT $triangle_1$ yang akan dibuang	Sebuah LIST POINT yang berisikan daftar POINT yang akan menggantikan POINT $triangle_1$

Tabel 2.16: Masukan, Proses, dan Keluaran dari Fungsi GETBETWEEN

2.15. Fungsi GetBetween

Fungsi GETBETWEEN bertanggung jawab untuk mencari list POINT yang akan menggantikan POINT yang akan dibuang. Pseudocode fungsi GETBETWEEN dapat dilihat pada pseudocode 2.17. Masukan, proses, dan keluaran dari fungsi ini tercantum pada tabel 2.16.

2.16. Fungsi Solve

Fungsi SOLVE bertanggung jawab untuk mencari *relative convex polygon* yang mengelilingi semua titik yang ada di dalam polygon luar. Pseudocode fungsi SOLVE dapat dilihat pada pseudocode 2.18. Masukan, proses, dan keluaran dari fungsi ini tercantum pada tabel 2.17.

Pseudocode 2.17: Fungsi GETBETWEEN

Input: $triangle, q, trees$

```
points ← Array POINT
while q not EMPTY do
    if INSIMPLEPOLYGON(q.FRONT(), triangle) ≠ 1 then
        points.INSERT(q.FRONT())
    end ifq.POP
end while
for pt in trees do
    if INSIMPLEPOLYGON(pt, triangle) ≠ 1 then
        points.INSERT(pt)
    end if
end for
P ← CONVEXHULL(points)
pts ← Array POINT
i ← 0
while TRUE do
    if P.P[i] = triangle.P[0] then
        if P.P[P.NEXT(i)] = triangle.P[2] then
            i ← P.PREV(i)
            while P.P[i] ≠ triangle.P[2] do
                pts.INSERT(P.P[i])
                i = P.PREV(i)
            end while
        else
            i ← P.NEXT(i)
            while P.P[i] ≠ triangle.P[2] do
                pts.INSERT(P.P[i])
                i = P.NEXT(i)
            end while
        end if
        BREAK
    end if
    i ← P.NEXT(i)
end while
return pts
```

Pseudocode 2.18: Fungsi SOLVE

Input: $perimeter, trees$

```
1: q ← QUEUE
2: for pt in perimeter.P do
3:     q.PUSH pt
4: end for
5: bef ← perimeter.P[perimeter.SIZE() − 1]
6: while TRUE do
7:     erased ← FALSE
8:     count ← q.SIZE()
9:     while count– do
10:        cur ← q.FRONT() q.POP()
11:        if cur.fixed =FALSE and (FINE(q, cur) or cur = bef
   or cur = q.FRONT()) then
12:            erased ← TRUE
13:            triangle ← POLYGON
14:            triangle.P.INSERT(bef, cur, q.FRONT())
15:            points ← GETBETWEEN(triangle, q, trees)
16:            for pt in points do
17:                q.PUSH(pt); bef ← pt
18:            end for
19:        else
20:            q.PUSH(cur); bef ← cur
21:        end if
22:    end while
23:    if erased = FALSE then BREAK
24:    end if
25: end while
26: hull ← array of POINT
27: while q not empty do
28:    hull.INSERT(q.FRONT())
29:    q.POP()
30: end while
31: return POLYGON(hull)
```

Masukan	Proses	Keluaran
Sebuah POLYGON $perimeter$ yang merupakan polygon sederhana yang merupakan poligon pembatas, dan array POINT $trees$ yang merupakan titik yang berada di dalam polygon pembatas	Mencari *relative convex hull* dari semua titik $trees$ di dalam POLYGON $perimeter$	Sebuah POLYGON yang merupakan *relative convex hull* dari semua titik $trees$

Tabel 2.17: Masukan, Proses, dan Keluaran dari Fungsi SOLVE

[*Halaman ini sengaja dikosongkan*]

DAFTAR PUSTAKA

[1] SPOJ. (2009). LL and ErBao, **url**: https://www.spoj.com/problems/ISUN1/.

[2] A. A. Melkman, **?**On-line construction of the convex hull of a simple polyline**?**, *Information Processing Letters 25*, **pages** 11–12, 1987.

[3] A. Andrew., **?**Another Efficient Algorithm for Convex Hulls in Two Dimensions**?**, *Information Processing Letters 9*, **pages** 216–219, 1979.

[4] geeksforgeeks. (2019). How to check if a given point lies inside or outside a polygon, **url**: https://www.geeksforgeeks.org/how-to-check-if-a-given-point-lies-inside-a-polygon/.