**Laporan Tugas Kecil 2**

**Implementasi Convex Hull untuk Visualisasi Tes Linear**

**Separability Dataset dengan Algoritma Divide and Conquer**

Mata Kuliah IF2211 - Strategi Algoritma



Oleh :

Nama : Roby Purnomo

NIM : 13520106

Kelas : 01

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021/2022

Daftar Isi

[**Bab I : Teori Dasar dan Spesifikasi** 2](#_Toc93946440)

[**Algoritma Brute Force** 2](#_Toc93946441)

[**Bab II : Source Program (Bahasa C)** 4](#_Toc93946442)

[**Modularity Program** 4](#_Toc93946443)

[**main.c** 4](#_Toc93946444)

[**matcher.c** 6](#_Toc93946445)

[**Bab III : Testing** 14](#_Toc93946446)

[**1.txt (Small)** 14](#_Toc93946447)

[**2.txt (Medium)** 17](#_Toc93946448)

[**3.txt (Medium)** 22](#_Toc93946449)

[**bigeurope.txt (Medium)** 27](#_Toc93946450)

[**bigstates.txt (Medium)** 35](#_Toc93946451)

[**genshin.txt (Small)** 40](#_Toc93946452)

[**smalllangs.txt (Small)** 44](#_Toc93946453)

[**big.txt (Big)** 48](#_Toc93946454)

[**big2.txt (Big)** 53](#_Toc93946455)

[**Bab IV : Kesimpulan dan Saran** 59](#_Toc93946456)

[**Kesimpulan** 59](#_Toc93946457)

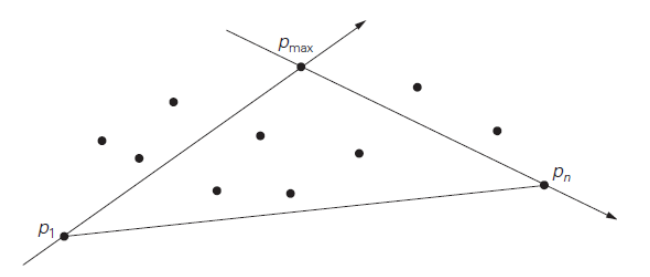
[**Saran** 59](#_Toc93946458)

# **Bab I : Teori Dasar dan Spesifikasi**

## **Algoritma Divide and Conquer**

Algoritma yang digunakan pada program ini yaitu mengimplementasikan strategi Divide and Conquer dengan langkah-langkah sebagai berikut.

1. Mula-mula cari titik yang terletak paling kiri dan paling kanan pada himpunan titik S dan simpan sebagai p1 dan pn
2. Lalu bagi menjadi 2 persoalan yaitu Convex Hull untuk titik di atas garis p1-pn dan titik di bawah garis p1-pn
3. Pada Convex Hull Top dan Convex Hull Bot, cari titik ekstrem, yaitu titik terjauh dari garis p1-pn yang terletak di atas/kiri dari garis p1-pn untuk Convex Hull Top dan di bawah/kanan untuk Convex Hull Bot. Jika terdapat titik dengan jarak ke garis p1-pn sama, maka pilih titik yang memiliki sudut pada gambar di bawah ini yang paling besar. Simpan titik ekstrem tersebut sebagai pmax.



1. Lalu dengan menggunakan prinsip divide and conquer, bagi permasalahan menjadi tingkat yang lebih kecil yaitu lakukan Convex Hull untuk bagian kiri dan kanan, yakni untuk garis p1-pmax dan pmax-pn dan menyimpan hasil return sebagai *left* dan *right*.
2. Lalu setelah melakukan divide, langkah terakhir adalah dengan melakukan merge yakni *left* + pmax + *right*, dan jika tidak ada nilai ekstremnya maka akan mengembalikan array kosong (sebagai basis)

# **Bab II : Source Program (Bahasa C)**

Dapat dilihat secara full pada alamat github di bawah ini.

[*https://github.com/robypurnomo/tucil\_2\_stima*](https://github.com/robypurnomo/tucil_2_stima)

## **Modularity Program**

├───lib

│   ├───charmachine

│   │   ├───charmachine.c

│   │   └───charmachine.h

│   ├───filereader

│   │   └───filereader.c

│   ├───matcher

│   │   └───matcher.c

│   ├───matrix

│   │   ├───matrix.c

│   │   └───matrix.h

│   └───boolean.h

└───main.c

## **main.c**

#include <stdio.h>

#include "lib/filereader/filereader.c"

#include "lib/matcher/matcher.c"

#include <string.h>

#define endl printf("\n")

FILE \*fp;

int main () {

    char path[100];

    char x[] = "exit";

    Matrix puzzle;

    List kata;

    while (true) {

        start :

        printf("========================== Tucil 1 =========================\n");

        endl;

        printf("Masukkan path file (isi \"exit\" untuk keluar) : ");

        scanf("%s", &path);

        if (strcmp(path, x) == 0){

            goto exit;

        }

        fp=fopen(path, "a");

        char str[] = "\n\n";

        fwrite(&str, sizeof(str), 1, fp);

        fclose(fp);

        fp=fopen(path, "r");

        if (fp == NULL) {

            endl;

            printf("File tidak ditemukan !!!\n");

            endl;

            goto start;

        }

        readFile(fp, &puzzle, &kata);

        endl;

        printf("========================= Puzzle ===========================\n");

        endl;

        displayMatrix(puzzle);

        endl;

        endl;

        printf("========================== Kata ============================\n");

        endl;

        displayWord(kata);

        endl;

        endl;

        printf("========================== Hasil ===========================\n");

        endl;

        result(&puzzle, &kata);

        endl;

    }

    exit :

        endl;

        printf("=========================== Exit ===========================\n");

        endl;

        printf("Terimakasih telah mencoba program saya :D\n");

        endl;

        printf("============================================================\n");

    fclose(fp);

    return 0;

}

## **matcher.c**

#include <stdio.h>

#include "../matrix/matrix.c"

#include "time.h"

void displayResult(Matrix m, int found, int len, int a, int b) {

    int i, j;

    int indexi[len];

    int indexj[len];

    switch (found) {

    case 0:

        printf("Kata tidak ditemukan !");

        break;

    case 1:

        for (i=0; i<len; i++) {

            indexi[i] = a;

            indexj[i] = b+i;

        }

        for(i=0; i<ROWS(m); i++){

            for(j=0; j<COLS(m); j++){

                if (isIndex(m, indexi, indexj, len, i, j)) {

                    printf("%c", CHAR(m, i, j));

                } else {

                    printf("-");

                }

                if(j != COLS(m)-1){

                    printf(" ");

                }

            }

            if(i != ROWS(m)-1){

                printf("\n");

            }

        }

        break;

    case 2:

        for (i=0; i<len; i++) {

            indexi[i] = a;

            indexj[i] = b-i;

        }

        for(i=0; i<ROWS(m); i++){

            for(j=0; j<COLS(m); j++){

                if (isIndex(m, indexi, indexj, len, i, j)) {

                    printf("%c", CHAR(m, i, j));

                } else {

                    printf("-");

                }

                if(j != COLS(m)-1){

                    printf(" ");

                }

            }

            if(i != ROWS(m)-1){

                printf("\n");

            }

        }

        break;

    case 3 :

        for (i=0; i<len; i++) {

            indexi[i] = a+i;

            indexj[i] = b;

        }

        for(i=0; i<ROWS(m); i++){

            for(j=0; j<COLS(m); j++){

                if (isIndex(m, indexi, indexj, len, i, j)) {

                    printf("%c", CHAR(m, i, j));

                } else {

                    printf("-");

                }

                if(j != COLS(m)-1){

                    printf(" ");

                }

            }

            if(i != ROWS(m)-1){

                printf("\n");

            }

        }

        break;

    case 4 :

        for (i=0; i<len; i++) {

            indexi[i] = a-i;

            indexj[i] = b;

        }

        for(i=0; i<ROWS(m); i++){

            for(j=0; j<COLS(m); j++){

                if (isIndex(m, indexi, indexj, len, i, j)) {

                    printf("%c", CHAR(m, i, j));

                } else {

                    printf("-");

                }

                if(j != COLS(m)-1){

                    printf(" ");

                }

            }

            if(i != ROWS(m)-1){

                printf("\n");

            }

        }

        break;

    case 5 :

        for (i=0; i<len; i++) {

            indexi[i] = a+i;

            indexj[i] = b+i;

        }

        for(i=0; i<ROWS(m); i++){

            for(j=0; j<COLS(m); j++){

                if (isIndex(m, indexi, indexj, len, i, j)) {

                    printf("%c", CHAR(m, i, j));

                } else {

                    printf("-");

                }

                if(j != COLS(m)-1){

                    printf(" ");

                }

            }

            if(i != ROWS(m)-1){

                printf("\n");

            }

        }

        break;

    case 6 :

        for (i=0; i<len; i++) {

            indexi[i] = a-i;

            indexj[i] = b-i;

        }

        for(i=0; i<ROWS(m); i++){

            for(j=0; j<COLS(m); j++){

                if (isIndex(m, indexi, indexj, len, i, j)) {

                    printf("%c", CHAR(m, i, j));

                } else {

                    printf("-");

                }

                if(j != COLS(m)-1){

                    printf(" ");

                }

            }

            if(i != ROWS(m)-1){

                printf("\n");

            }

        }

        break;

    case 7 :

        for (i=0; i<len; i++) {

            indexi[i] = a-i;

            indexj[i] = b+i;

        }

        for(i=0; i<ROWS(m); i++){

            for(j=0; j<COLS(m); j++){

                if (isIndex(m, indexi, indexj, len, i, j)) {

                    printf("%c", CHAR(m, i, j));

                } else {

                    printf("-");

                }

                if(j != COLS(m)-1){

                    printf(" ");

                }

            }

            if(i != ROWS(m)-1){

                printf("\n");

            }

        }

        break;

    case 8 :

        for (i=0; i<len; i++) {

            indexi[i] = a+i;

            indexj[i] = b-i;

        }

        for(i=0; i<ROWS(m); i++){

            for(j=0; j<COLS(m); j++){

                if (isIndex(m, indexi, indexj, len, i, j)) {

                    printf("%c", CHAR(m, i, j));

                } else {

                    printf("-");

                }

                if(j != COLS(m)-1){

                    printf(" ");

                }

            }

            if(i != ROWS(m)-1){

                printf("\n");

            }

        }

        break;

    default:

        break;

    }

}

void matcher(Matrix m, char word[], int len, int \*countout) {

    int i, j, k;

    int found = 0;

    int count = 0;

    for (i=0; i<ROWS(m); i++) {

        for (j=0; j<COLS(m); j++) {

            // printf("%d %d\n", i, j);

            count++;

            if (CHAR(m, i, j) == word[0]) {

                // printf("%c %c\n", CHAR(m,i,j), word[0]);

                int a,b;

                boolean keep;

                /\* Horizontal \*/

                /\* Kanan \*/

                if (j+len <= COLS(m)) {

                    keep = true;

                    a=i, b=j+1, k=1;

                    while(k < len && keep) {

                        count++;

                        if (CHAR(m, a, b) != word[k]) {

                            keep = false;

                        }

                        b++;k++;

                    }

                    if (keep) {

                        found=1;

                        goto result;

                    }

                }

                /\* Kiri \*/

                if (j-len > -2) {

                    keep = true;

                    a=i, b=j-1, k=1;

                    while(k < len && keep) {

                        count++;

                        if (CHAR(m, a, b) != word[k]) {

                            keep = false;

                        }

                        b--;k++;

                    }

                    if (keep) {

                        found=2;

                        goto result;

                    }

                }

                /\* Vertikal \*/

                /\* Bawah \*/

                if (i+len <= ROWS(m)  && !found) {

                    keep = true;

                    a=i+1, b=j, k=1;

                    while(k < len && keep) {

                        count++;

                        if (CHAR(m, a, b) != word[k]) {

                            keep = false;

                        }

                        a++;k++;

                    }

                    if (keep) {

                        found=3;

                        goto result;

                    }

                }

                /\* Atas \*/

                if (i-len > -2) {

                    keep = true;

                    a=i-1, b=j, k=1;

                    while(k < len && keep) {

                        count++;

                        if (CHAR(m, a, b) != word[k]) {

                            keep = false;

                        }

                        // printf("%c %c %d\n", CHAR(m,a,b), word[k], keep);

                        a--;k++;

                    }

                    if (keep) {

                        found=4;

                        goto result;

                    }

                }

                /\* Diagonal \*/

                /\* Kanan bawah \*/

                if (j+len <= COLS(m) && i+len <= ROWS(m)) {

                    keep = true;

                    a=i+1, b=j+1, k=1;

                    while(k < len && keep) {

                        count++;

                        if (CHAR(m, a, b) != word[k]) {

                            keep = false;

                        }

                        a++;b++;k++;

                    }

                    if (keep) {

                        found=5;

                        goto result;

                    }

                }

                /\* Kiri atas \*/

                if (j-len > -2 && i-len > -2) {

                    keep = true;

                    a=i-1, b=j-1, k=1;

                    while(k < len && keep) {

                        count++;

                        if (CHAR(m, a, b) != word[k]) {

                            keep = false;

                        }

                        a--;b--;k++;

                    }

                    if (keep) {

                        found=6;

                        goto result;

                    }

                }

                /\* Kanan atas \*/

                if (j+len <= COLS(m) && i-len > -2) {

                    keep = true;

                    a=i-1, b=j+1, k=1;

                    while(k < len && keep) {

                        count++;

                        if (CHAR(m, a, b) != word[k]) {

                            keep = false;

                        }

                        a--;b++;k++;

                    }

                    if (keep) {

                        found=7;

                        goto result;

                    }

                }

                /\* Kiri bawah \*/

                if (j-len > -2 && i+len <= ROWS(m)) {

                    keep = true;

                    a=i+1, b=j-1, k=1;

                    while(k < len && keep) {

                        count++;

                        if (CHAR(m, a, b) != word[k]) {

                            keep = false;

                        }

                        a++;b--;k++;

                    }

                    if (keep) {

                        found=8;

                        goto result;

                    }

                }

            }

        }

    }

    result :

        printf("Jumlah perbandingan yang dilakukan : %d\n", count);

        \*countout = count;

        displayResult(m, found, len, i, j);

}

void result(Matrix \*m, List \*l) {

    int i,j;

    clock\_t begin = clock();

    int min = 100;

    int count = 0; int sum = 0;

    for (i = 0; i < ROWS(\*l); i++) {

        if (COLS(\*l)[i] < min) {

            min = COLS(\*l)[i];

        }

    }

    i=0;

    while (i < ROWS(\*l)) {

        printf("%d. %s\n", i+1, WORD(\*l, i));

        matcher(\*m, WORD(\*l, i), COLS(\*l)[i], &count);

        sum += count;

        printf("\n\n");

        i++;

    }

    clock\_t end = clock();

    double time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC;

    printf("Total perbandingan yang diperlukan : %d kali\n", sum);

    printf("Total waktu yang diperlukan : %.2f detik\n", time\_spent);

    clearMatrix(m);

    clearList(l);

}

# **Bab III : Testing**

# **Bab IV : Kesimpulan dan Saran**

## **Kesimpulan**

Saya berhasil membuat Implementasi Convex Hull untuk **“Visualisasi Tes Linear Separability Dataset dengan Algoritma Divide and Conquer”** dengan bahasa pemrograman python dengan spesifikasi dan sejauh ini belum ada solusi dari program yang terlewat dan/atau tidak tepat. Dan juga saya melakukan testing untuk sampel darta lain yaitu iris, wine, breast\_cancer, digits, dan linnerud

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| 1. Pustaka myConvexHull berhasil dibuat  dan tidak ada kesalahan | ✓ | ⨉ |
| 2. Convex hull yang dihasilkan sudah  benar | ✓ | ⨉ |
| 3. Pustaka myConvexHull dapat  digunakan untuk menampilkan convex  hull setiap label dengan warna yang  berbeda. | ✓ | ⨉ |
| 4. **Bonus** : program dapat menerima input  dan menuliskan output untuk dataset  lainnya | ✓ | ⨉ |

## **Saran**

Kedepannya mungkin dapat ditemukan algoritma yang lebih baik daripada algoritma Divide and Conquer yang diimplementasikan pada program kali ini dalam segi kompleksitas maupun efektifitas.