Final Paper Data Struct

FERLANGGA WISPRA ARDHANA - 2501993030
FIRDAUS RABBY MOHAMAD RAFHAEL - 2502049473
IRHAM ZHARFAN ATMOKO - 2502007880
NABIILAH PUTRI AVIARTA - 2502018133
ZAKI WIDYADHANA WIRAWAN - 2501992261

< (1st data structure type)> and <(2nd data structure type)> Data Structure for Simple Image Processing

1. Pendahuluan
    a. Apa itu data structure? (secara singkat)
    b. Apa itu Image Processing? (secara singkat)
    c. Apa peran dari data structure di image Processing?
2. Literature review
    a. Penelitian tentang operasi sederhana yang ada di Image Processing terkait data structure
    b. Penelitian Jenis -jenis data structure yang digunakan pada Image Processing
    c. Penelitian Perbandingan kelebihan dan kelemahan jenis-jenis data structure yang disebut pada poin (a), dalam hal:
        i. Kompleksitas
        ii. Running time
3. Eskperimen dan Analisa
    a. Data set yang digunakan, contoh data yang bisa digunakan:
        i. https://imerit.net/blog/22-free-image-datasets-for-computer-vision-all-pbm/
    b. Proses implementasi (Pseudocode/Flowchart) data struktur dengan image processing
4. Kesimpulan
    a. Hasil eksperimen, data struktur mana yang lebih baik?

A. sebuah struktur data adalah cara penyimpanan, penyusunan dan pengaturan data di dalam media penyimpanan komputer sehingga data tersebut dapat digunakan secara efisien dan tertata dalam menyusun sesuatu seperti data atau media penyimpanannya. Kemudian struktur data dalam teknik pemprograman berarti tata letak dat**a** yang berisi kolom-kolom data, baik itu kolom yang tampak oleh pengguna (user) ataupun kolom yang hanya digunakan untuk keperluan pemrograman yang tidak tampak oleh pengguna.

B. Image processing merupakan bidang ilmu yang mempelajari tentang bagaimana suatu citra itu dibentuk, diolah, dan dianalisis sehingga menghasilkan informasi yang dapat dipahami oleh manusia.

C. Peranan data structure di image processing adalah terbagi kedalam beberapa level, seperti contoh ini terdapat data-data original, matriks integer dengan data tentang kecerahan pixel. Gambar dalam level memiliki output yang terdiri atas operasi-operasi pemrosesan awal yang digunakan untuk menyoroti aspek-aspek dalam gambar yang penting untuk perlakuan yang lebih lanjut. Dan hal ini dinamakan iconic images

2. A. Dalam image processing dengan data structure

Contoh operasi sederhana dari image processing:

- Putar/rotasi image
- Mengatur true tone dari image
- Copy and preview sebuah image, ini ialah operasi dasar dalam image processing
- Cut and paste, merupakan operasi yang dapat memotong sebuah area difoto dan menempelkan di file/tempat baru

Dalam operasi Aritmetika ini kita dapat menambah / mengurangi intesitas, mengurangi / menghilangkan noise, dan mendeteksi perubahan gambar.

B. **Analog Image Processing**

Jenis tipe yang pertama yaitu pengolahan citra analog atau *analog image processing*. Ini merupakan teknik pemrosesan gambar analog atau visual dapat digunakan untuk *hard copy* seperti cetakan dan foto. Analis gambar seringkali menggunakan berbagai dasar interpretasi saat menggunakan teknik visual ini.Asosiasi, ini juga merupakan alat penting lainnya dalam pemrosesan gambar melalui teknik visual.Jadi, analis menerapkan kombinasi pengetahuan pribadi dan data jaminan untuk pemrosesan gambar.Contoh khas dari pemrosesan gambar atau *image processing* adalah televisi.

QUEUE

antrean adalah koleksi dari data-data yang memiliki urutan dan hanya bisa diubah dengan menambahkan data di satu ujung dan mengeluarkan data di ujung lainnya. Untuk 2 b aing baru nemu segini, kalo ada asupan langsung dimasukin - FER

C. i. dalam kompleksitas sebuah algoritma haruslah ada sebuah efisiensi (kemangkusan) algoritma karena algoritma yang bagus —> yang efektif, diukur dari seberapa lama waktu dan memori yang dibutuhkan untuk menjalankan. Kebutuhan waktu dan ruang tergantung pada ukuran masukan yang biasanya adalah banyaknya jumlah data yang akan diproses. Perbandingan antara kelebihan dan kelemahan data structure

Kelebihan:

- Array mempunyai penyimpanan data yang sangat mudah
- Linked list mempunyai kompleksitas dan mampu bekerja lebih baik dari array

Kekurangan:

- Array & matrix memilki sebuah kelemahan didalam kompleksitas, yaitu array memiliki kinerja lebih lambat dari linked list

- Linked list memiliki kompleksitas dan dapat bekerja lebih cepat, namun memiliki sebuah kelemahan yang sangat sulit untuk digunakan dan diakses tidak seperti array

Ii. running time dari suatu algoritma untuk input tertentu tergantung pada jumlah operasi yang dieksekusi. Semakin besar jumlah operasi, semakin lama waktu berjalan dari suatu algoritma.biasanya kami ingin tahu berapa banyak operasi yang akan dijalankan oleh suatu algoritma secara proporsional dengan ukuran inputnya.

3. A. Bivariat artinya adalah dua variabel. Jadi, jenis dataset ini menggambarkan hubungan antara dua variabel saja. Contohnya, misalkan Anda ingin menentukan bonus untuk masing-masing anggota tim sales. Jadi, bonus yang diberikan akan dihitung berdasarkan dua variabel berikut:
Jumlah produk yang dijual
Jumlah keuntungan yang diraih per bulannya
Nah, karena kedua data tersebut saling berhubungan, maka kumpulan data tersebut masuk ke dalam kategori bivariate dataset. Selanjutnya pemahaman bivariat adalah analisis **data** yang dilakukan untuk mencari korelasi atau pengaruh antara 2 variabel atau lebih yang diteliti. Pada penelitian ini sebelum dilakukan analisis **data**, terlebih dahulu dilakukan uji normalitas **data** untuk mengetahui normal atau tidaknya **data** yang ada.

B.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


struct node{
	int iso;
	int size;
	char name[40];
	char type[10];
	char nikon[20];
	char strSpeed[10];
	char lensa[10];
	char loc[50];
	char putihabu[4];


	struct node *left;
	struct node *right;
};
```

```
struct node *createNewNode(struct node *root,char name[], char type[], char nikon[]
,char strSpeed[],int iso,char lensa[],char putihabu[],char loc[], int size){
	struct node *newNode = (struct node*)malloc(sizeof(struct node));

		newNode->size = size;
	newNode->iso = iso;
	newNode->left = NULL;
	newNode->right = NULL;

	strcpy(newNode->name, name);
	strcpy(newNode->nikon, nikon);
	strcpy(newNode->strSpeed, strSpeed);
	strcpy(newNode->lensa, lensa);
	strcpy(newNode->putihabu, putihabu);
	strcpy(newNode->loc, loc);
	strcpy(newNode->type, type);


	return newNode;
}

struct node* minvalnode(struct node* node){
   struct node* current = node;

   while (current && current->left != NULL)
      current = current->left;

   return current;
}

struct node * insert(struct node *root,char name[], char type[], char nikon[],  char strSpeed[],int iso,char
lensa[],char putihabu[],char loc[], int size){
	if(root == NULL){
		return createNewNode(root, name, type,  nikon, strSpeed, iso, lensa, putihabu, loc, size);
	}
	else if(size < root->size){
		root->left = insert(root->left, name, type,  nikon, strSpeed, iso, lensa, putihabu, loc, size);
	}
	else if(size >= root->size){
```

```
                root->right = insert(root->right, name, type,  nikon, strSpeed, iso, lensa, putihabu, loc,
size);
        }
        return root;
}

//display
void inOrder(struct node *root){
        if(root == NULL){
                return;
        }
        else{
                inOrder(root->left);
                printf("%s.%s, %d KB - %s\nnikon: %s, Shutter Speed: %s, ISO: %d, Aperture: %s,
Grayscale: %s \n", root->name, root->type, root->size, root->loc,
                root->nikon, root->strSpeed, root->iso, root->lensa, root->putihabu);
                inOrder(root->right);
        }
}


struct node* deletenode(struct node* root, char name[]){
        if(root == NULL) {
                return;
        }
        else if(strcmp(root->name, name) != 0){
                root->left = deletenode(root->left, name);
                root->right = deletenode(root->right, name);
        }
        else if(strcmp(root->name, name) == 0) {
      if (root->left == NULL) {
        struct node* temp = root->right;
        free(root);
        return temp;
      }
      else if (root->right == NULL) {
        struct node* temp = root->left;
        free(root);
        return temp;
      }
```

```
        struct node* temp = minvalnode(root->right);

        root = temp;

        root->right = deletenode(root->right, temp->name);
            return root;
            }
}

struct node* searchname(struct node* root, char name[]){
            if(root == NULL) {
                        return;
            }
            else if(strcmp(root->name, name) != 0){
                        root->left = searchname(root->left, name);
                        root->right = searchname(root->right, name);
            }
            else if(strcmp(root->name, name) == 0) {
                        printf("\n%s.%s, %d KB - %s\nnikon: %s, Shutter Speed: %s, ISO: %d, Aperture: %s,
Grayscale: %s \n", root->name, root->type, root->size, root->loc,
                                    root->nikon, root->strSpeed, root->iso, root->lensa, root->putihabu);
            return root;
             }
}

int main(){


            struct node* root = NULL;
            char name[100];
            char nikon[100];
            char type[5];
            char strSpeed[100];
            int iso;
            char lensa[100];
            char putihabu[100];
            char loc[100];
            int size;
            int i;
```

```
	FILE *fp;
	fp=fopen("imagesize.txt","r");

	i=0;
	while(!feof(fp)){
		fscanf(fp,"%[^.].%[^#]#%[^#]#%[^#]#%d#%[^#]#%[^#]#%[^#]#%d\n",name, type,
nikon, strSpeed, &iso, lensa, putihabu, loc, &size);
		root = insert(root, name,type, nikon, strSpeed, iso, lensa, putihabu, loc, size);
		i++;
	}

	fclose(fp);
	printf("Deleting foto_60.bmp...\n");
	deletenode(root, "foto_39");
	printf("Searching for foto_69.jpg...\n");
	searchname(root, "foto_65");
	printf("Displaying list image:\n\n");
	inOrder(root);


	return 0;
}
```