

Tugas 2

Analisis Algoritma



Disusun oleh :

Afifah Kho'eriah (140810160008)
Baby Cattleya Gustina Permatagama (140810160048)
Muhammad Islam Taufikurahman (140810160062)

S-1 Teknik Informatika
Fakultas Matematika & Ilmu Pengetahuan Alam
Universitas Padjadjaran
Jalan Raya Bandung - Sumedang Km. 21 Jatinangor 45363

1. Program *Searching Linear*

/*

```

Nama Program : Program Linear Search
Oleh      : Baby, Afifah, Islam
Dibuat   : Senin, 25 Maret 2018
*/

#include <iostream>
#include <ctime>
#include <cstdlib>
#include <chrono>

using namespace std;

int linearSearch(const int [], int, int);
int SIZE;

int main() {

    cout << "Masukkan Banyak Elemen : " ;
    cin >> SIZE;

    int array[SIZE];
    unsigned seed = time(0);
    srand(seed);

    cout << "Array Angka Random: " <<endl;

    for(int i = 0; i<100;i++)
    {
        array[i]=rand()%100+1;
        cout << array[i]<<" ";
    }

    int results;
    int input;

    cout << endl << endl <<"Masukkan Angka yang Dicari: ";
    cin >> input;

    auto start = chrono::steady_clock::now();

    results = linearSearch(array, SIZE, input);

    if (results == -1){
        cout << "Angka Tidak Ditemukan di Array\n";
    }
    else {
        cout << "Angka Ditemukan di indeks ke " << results;
        cout << " pada array.\n" <<endl;
    }

    auto end = chrono::steady_clock::now();
    auto diff = end - start;
    cout << "Running Time: "<<chrono::duration <double, milli>
(diff).count() << " ms" << endl;

    return 0;
}

```

```

int linearSearch(const int array[], int size, int value){
    int lokasi = 0;
    bool found = false;

    while (!found && lokasi < size) {
        if (array[lokasi] == value){
            found = true;
        }
        else {
            lokasi=lokasi+1;
        }
    }

    return lokasi;
}

```

2. Buat Program *Binary Search*

```

/*
    Nama Program : Program Binary Search
    Oleh      : Baby, Afifah, Islam
    Dibuat   : 6 April 2018
*/

#include <iostream>
#include <chrono>
using namespace std;

main () {
    int n, i, search, first, last, middle;
    cout << "Masukkan Jumlah Elemen: ";
    cin >> n;

    int arr[n];
    unsigned seed = time(0);
    srand(seed);

    cout << "Array Angka Random: " << endl;

    for(int i = 0; i<n;i++) //to get random numbers inside the array.
    {
        arr[i]=rand()%1000+1;
        cout << arr[i]<<" ";
    }

    cout<<endl<<"Masukkan angka yang akan dicari :";
    cin>>search;

    auto start = chrono::steady_clock::now();

    int posisi;
    for (int i=0; i<n-1; i++) {
        posisi=i;
        for (int j=i+1;j<n;j++) {
            if (arr[posisi]>arr[j]) {

```

```

        posisi=j;
    }
    swap(arr[i], arr[posisi]);
}

cout << endl << "Array Angka Sorted: " <<endl;
for(int i = 0; i<n;i++) //to get random numbers inside the array.
{
    cout << arr[i]<<" ";
}

first = 0;
last = n-1;
middle = (first+last)/2;

while (first <= last)
{
    if(arr[middle] < search)
    {
        first = middle + 1;
    }
    else if(arr[middle] == search)
    {
        cout<< endl <<"Angka " << search<<" ditemukan"<<endl;
        break;
    }
    else
    {
        last = middle - 1;
    }
    middle = (first + last)/2;
}

if(first > last)
{
    cout<<endl<<"Error! " <<search<<" tidak ditemukan dalam Array" <<endl;
}

auto end = chrono::steady_clock::now();
auto diff = end - start;

    cout << "waktu program : " <<chrono::duration <double, milli>
(diff).count() << " ms" << endl;
}

```

3. Program Maksimum Minimum Sort

```

// Nama: Maksimum Minimum Sort (C++)
// Kelompok: Afifah 140810160008, Baby 140810160048, Muhammad Islam
140810160062
// Mata Kuliah: Analisis Algoritma

```

```

#include<iostream>

```

```

#include<conio.h>
#include <ctime>
#include <cstdlib>
#include <chrono>

using namespace std;

int main()
{

    int n;
    int i,temp,j;

    cout << "Masukan panjang array :";
    cin >> n;

    int a[n];

    unsigned seed = time(0);
    srand(seed);

    // Buat Random Array
    for(int i = 0; i<n; i++)
    {
        a[i]=rand()%1000+1;
    }

    cout << "Data array sebelum sort :";
    for(j=0;j<n;j++)
    {
        cout<<a[j]<<" ";
    }

    auto start = chrono::steady_clock::now();

    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            if(a[j]>a[j+1])
            {
                temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
            }
        }
    }

    auto end = chrono::steady_clock::now();
    auto diff = end - start;

    cout<<endl;

    cout<<"\nData setelah sorting (ASC): ";
    for(j=0;j<n;j++)
    {
        cout<<a[j]<<" ";
    }
}

```

```

    cout<<"\nData setelah sorting (DESC): ";
    for(j=n-1;j>=0;j--)
    {
        cout<<a[j]<<" ";
    }

    cout << endl <<endl <<"Runtime : " << chrono::duration <double, milli>
(diff).count() << " ms" << endl;

}

```

4. Quick Sort (menggunakan pointer linked list)

```

/*
    Nama Program : C++ Quick Sort Singly Linked List
    Oleh      : Baby, Afifah, Islam
    Dibuat   : 6 April 2018
*/

#include <iostream>
#include <cstdio>
#include <ctime>
#include <stdlib>
#include <chrono>

using namespace std;

/* node singly linked list */
struct Node
{
    int data;
    struct Node *next;
};

/* function insert first linked list */
void push(struct Node** head_ref, int new_data)
{
    /* buat node baru */
    struct Node* new_node = new Node;

    /* masukan node di data */
    new_node->data = new_data;

    /* link node baru ke head */
    new_node->next = (*head_ref);

    /* pindahkan head ke node baru */
    (*head_ref) = new_node;
}

/* function cetak linked list */
void printList(struct Node *node)
{
    while (node != NULL)
    {
        printf("%d ", node->data);
    }
}

```

```

        node = node->next;
    }
    printf("\n");
}

// mengambil nilai terakhir dari list
struct Node *getTail(struct Node *cur)
{
    while (cur != NULL && cur->next != NULL)
        cur = cur->next;
    return cur;
}

// Partisi list mengambil last element sebagai pivot
struct Node *partition(struct Node *head, struct Node *end,
                      struct Node **newHead, struct Node
**newEnd)
{
    struct Node *pivot = end;
    struct Node *prev = NULL, *cur = head, *tail = pivot;

    // Saat Partisi, head dan tail mengalami perubahan jadi newHead dan
    newEnd
    while (cur != pivot)
    {
        if (cur->data < pivot->data)
        {
            // Node pertama yang memiliki value kurang dari pivot
            menjadi head baru

            if ((*newHead) == NULL)
                (*newHead) = cur;

            prev = cur;
            cur = cur->next;
        }
        else // Jika node lebih besar dari pivot
        {
            // Pindahkan node ke next dari tail, dan ubah tail
            if (prev)
                prev->next = cur->next;
            struct Node *tmp = cur->next;
            cur->next = NULL;
            tail->next = cur;
            tail = cur;
            cur = tmp;
        }
    }

    // Jika pivot adalah element terkecil di list,
    // pivot jadi head
    if ((*newHead) == NULL)
        (*newHead) = pivot;

    // Update newEnd
    (*newEnd) = tail;

    // Return node pivot
    return pivot;
}

```

```

}

//here the sorting happens exclusive of the end node
struct Node *quickSortRecur(struct Node *head, struct Node *end)
{
    // base condition
    if (!head || head == end)
        return head;

    Node *newHead = NULL, *newEnd = NULL;

    // Partisi list, newHead dan newEnd akan diupdate berdasarkan function
    partisi
    struct Node *pivot = partition(head, end, &newHead, &newEnd);

    // Jika pivot adalah element terkecil - tidak perlu recur bagian
    kiri.
    if (newHead != pivot)
    {
        // Set node sebelum pivot = NULL
        struct Node *tmp = newHead;
        while (tmp->next != pivot)
            tmp = tmp->next;
        tmp->next = NULL;

        // Recur for the list before pivot
        newHead = quickSortRecur(newHead, tmp);

        // Ubah next dari node terakhir dari bagian kiri pivot
        tmp = getTail(newHead);
        tmp->next = pivot;
    }

    // Recur for the list after the pivot element
    pivot->next = quickSortRecur(pivot->next, newEnd);

    return newHead;
}

// main function untuk quick sort.
// function quickSortRecur()
void quickSort(struct Node **headRef)
{
    (*headRef) = quickSortRecur(*headRef, getTail(*headRef));
    return;
}

int main()
{
    struct Node *a = NULL;

    int n;
    cout << "Masukan panjang array :";
    cin >> n;

    int A[n];

    unsigned seed = time(0);

```



```

    srand(seed);

    // Buat Random Array
    for(int i = 0; i<n; i++)
    {
        A[i]=rand()%1000+1;
    }

    for(int i = 0; i<n; i++)
    {
        push(&a, A[i]);
    }

    cout << endl << "List sebelum sorting \n";
    printList(a);

    auto start = chrono::steady_clock::now();

    quickSort(&a);

    auto end = chrono::steady_clock::now();
    auto diff = end - start;

    cout << "List setelah sorting \n";
    printList(a);

    cout << endl << "Runtime : " << chrono::duration <double, milli>
(diff).count() << " ms" << endl;

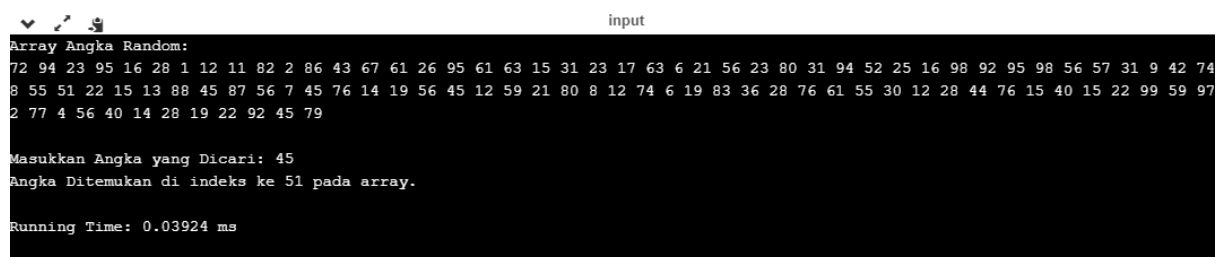
    return 0;
}

```

5. Analisis kompleksitas dan perbandingan waktu running time tiap metode dan jumlah data

A. Linear Search

Running Program :



```

input
Array Angka Random:
72 94 23 95 16 28 1 12 11 82 2 86 43 67 61 26 95 61 63 15 31 23 17 63 6 21 56 23 80 31 94 52 25 16 98 92 95 98 56 57 31 9 42 74
8 55 51 22 15 13 88 45 87 56 7 45 76 14 19 56 45 12 59 21 80 8 12 74 6 19 83 36 28 76 61 55 30 12 28 44 76 15 40 15 22 99 59 97
2 77 4 56 40 14 28 19 22 92 45 79

Masukkan Angka yang Dicari: 45
Angka Ditemukan di indeks ke 51 pada array.

Running Time: 0.03924 ms

```

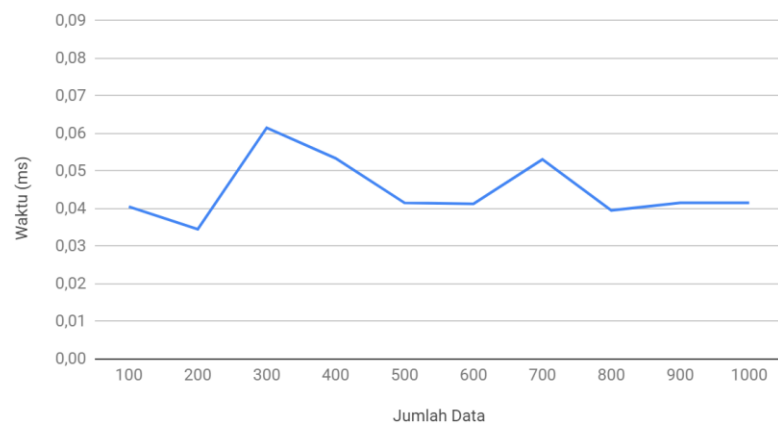
Waktu Running Program :

Jumlah Data	Waktu (ms)
100	0,04053
200	0,03453
300	0,06153
400	0,05341
500	0,04155
600	0,04130
700	0,05311
800	0,03955
900	0,04155
1000	0,04155

Kompleksitas :

Grafik :

Grafik Kompleksitas Linear Search



B. Binary Search

Running Program :

```

input
Masukkan Jumlah Elemen: 100
Array Angka Random:
815 449 629 948 999 214 4 928 227 410 43 12 676 422 94 224 322 59 775 228 797 287 768 9 762 912 577 805 442 445 481 256 894 109
03 892 322 558 172 548 967 566 911 643 988 357 218 661 415 344 888 563 630 656 923 743 919 499 548 360 944 28 615 189 136 169 4
809 78 955 356 44 521 618 38 860 974 607 520 740 951 407 302 932 414 225 675 684 723 222 43 18 601 9 206 88 529 638 896 606
Masukkan angka yang akan dicari :520

Array Angka Sorted:
4 9 9 12 18 28 38 43 43 44 59 78 88 94 109 136 169 172 189 203 206 214 218 222 224 225 227 228 256 287 302 322 322 344 356 357
0 407 410 414 415 422 432 442 445 449 481 499 520 521 529 548 548 558 563 566 577 601 606 607 615 618 629 630 638 643 656 661 6
676 684 723 740 743 762 768 775 797 805 809 815 860 888 892 894 896 911 912 919 923 928 932 944 948 951 955 967 974 988 999
Angka 520 ditemukan
waktu program :0.05834 ms

...Program finished with exit code 0
Press ENTER to exit console.

```

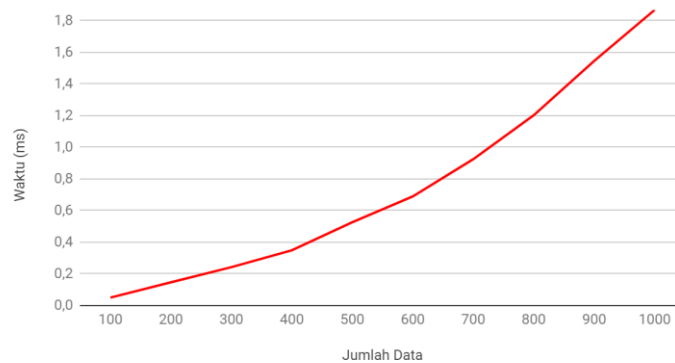
Waktu Running Program :

Jumlah Data	Waktu (ms)
100	0,05017
200	0,14670
300	0,24231
400	0,3491
500	0,52653
600	0,6893
700	0,9247
800	1,2024
900	1,54560
1000	1,86750

Kompleksitas :

Grafik :

Grafik Kompleksitas Binary Search



C. Max Min Search

Running Program :

```
input
Masukan panjang array :100
Data array sebelum sort :68 10 79 23 31 39 76 44 48 64 28 38 86 35 16 21 50 4 81 58 81 78 99 99 17 6 12 38 38 8 46 5 17 25 80
9 15 7 94 14 23 21 4 60 7 19 80 8 75 12 66 55 41 64 53 10 21 16 47 58 75 92 14 91 68 93 41 82 100 34 48 22 54 51 81 12 69 13
95 76 84 49 17 99 53 78 19 20 76 76 94 67 42 84 87 34 76 20 85

Data setelah sorting (ASC): 4 4 5 6 7 7 8 8 10 10 12 12 12 13 14 14 15 16 16 17 17 17 19 19 19 20 20 21 21 21 22 23 23 25 28
34 34 35 38 38 38 39 41 41 42 44 46 47 48 48 49 50 51 53 53 54 55 58 58 60 64 64 66 67 68 68 69 75 75 76 76 76 76 76 78 78 7
80 80 81 81 81 82 84 84 85 86 87 91 92 93 94 94 95 99 99 99 99 100

Data setelah sorting (DESC): 100 99 99 99 99 95 94 94 93 92 91 87 86 85 84 84 82 81 81 81 80 80 79 78 78 76 76 76 76 75 75
9 68 68 67 66 64 64 60 58 58 55 54 53 53 51 50 49 48 48 47 46 44 42 41 41 39 38 38 38 35 34 34 31 28 25 23 23 22 21 21 21 20
19 19 19 17 17 17 16 16 15 14 14 13 12 12 12 10 10 8 8 7 7 6 5 4 4

Runtime : 0.060252 ms
```

Waktu Running Program :

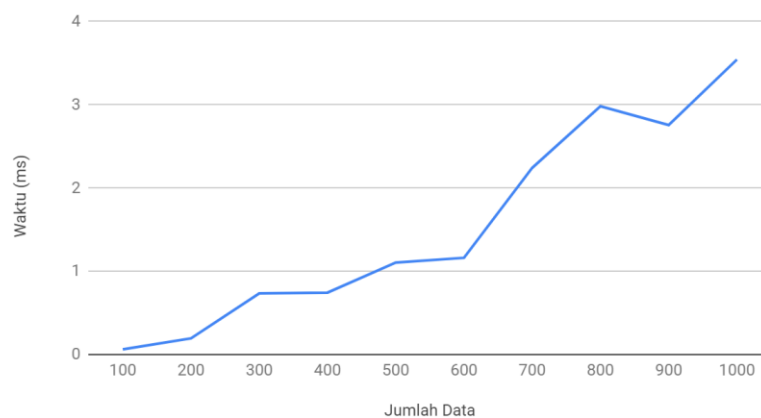
Jumlah	Waktu (ms)
--------	------------

Data	
100	0,06202
200	0,19345
300	0,7345
400	0,7431
500	1,1045
600	1,1613
700	2,242
800	2,982
900	2,7564
1000	3,5432

Kompleksitas :

Grafik :

Grafik Kompleksitas Max-Min Sort



D. Quick Sort

Running Program :

```

input
Masukan panjang array :100

List sebelum sorting
8 780 949 478 485 677 45 937 136 141 483 93 869 116 502 515 566 64 621 450 481 604 622 312 416 97 861 91 595 773 414 179 296 92
0 434 197 190 552 455 43 921 15 592 3 551 86 894 765 232 18 829 169 837 174 452 326 151 737 678 64 478 143 746 748 379 530 742
148 632 89 452 918 465 155 110 435 855 525 937 64 830 304 366 512 23 364 297 736 260 184 319 379 413 865 6 232 899 302 345 363

List setelah sorting
3 6 8 15 18 23 43 45 64 64 64 86 89 91 93 97 110 116 136 141 143 148 151 155 169 174 179 184 190 197 232 232 260 296 297 302 30
4 312 319 326 345 363 364 366 379 379 413 414 416 434 435 450 452 452 455 465 478 478 481 483 485 502 512 515 525 530 551 552 5
66 592 595 604 621 622 632 677 678 736 737 742 746 748 765 773 780 829 830 837 855 861 865 869 894 899 918 920 921 937 937 949

Runtime : 0.014588 ms

...Program finished with exit code 0
Press ENTER to exit console.

```

Waktu Running Program :

Jumlah Data	Waktu (ms)
100	0,0145
200	0,0307
300	0,0453
400	0,0642
500	0,0845
600	0,1653
700	0,1199
800	0,1456
900	0,1705
1000	0,1785

Kompleksitas :

Grafik :

