Studi Kasus 1: MERGE SORT

#include <iostream>

#include <chrono>

using namespace std;

using namespace std::chrono;

void merge (int \*a, int low, int high, int mid){

int i, j, k, temp[high-low+1];

i = low;

k = 0;

j = mid + 1;

while (i <= mid && j <= high){

if (a[i] < a[j]){

temp[k] = a[i];

k++;

i++;

}

else {

temp[k] = a[j];

k++;

j++;

}

}

while (i <= mid){

temp[k] = a[i];

k++;

i++;

}

while (j <= high){

temp[k] = a[j];

k++;

j++;

}

for (i = low; i <= high; i++){

a[i] = temp[i-low];

}

}

void mergeSort(int \*a, int low, int high){

int mid;

if (low < high){

mid=(low+high)/2;

mergeSort(a, low, mid);

mergeSort(a, mid+1, high);

merge(a, low, high, mid);

}

}

int main(){

int n, i;

high\_resolution\_clock::time\_point t1 = high\_resolution\_clock::now();

cout<<"Masukkan jumlah elemen data yang ingin diurutkan: ";

cin>>n;

int arr[n];

for(i = 0; i < n; i++){

cout<<"Masukkan elemen ke-"<<i+1<<": ";

cin>>arr[i];

}

mergeSort(arr, 0, n-1);

cout<<"\nArray yang telah diurutkan: ";

for (i = 0; i < n; i++) cout<<" "<<arr[i];

high\_resolution\_clock::time\_point t2 = high\_resolution\_clock::now();

auto duration = duration\_cast<microseconds>( t2 - t1 ).count();

cout<<endl<<duration<<" microseconds"<<endl;

}

Output:

Masukkan jumlah elemen data yang ingin diurutkan: 20

Masukkan elemen ke-1: 7

Masukkan elemen ke-2: 8

Masukkan elemen ke-3: 1

Masukkan elemen ke-4: 2

Masukkan elemen ke-5: 10

Masukkan elemen ke-6: 3

Masukkan elemen ke-7: 4

Masukkan elemen ke-8: 5

Masukkan elemen ke-9: 6

Masukkan elemen ke-10: 7

Masukkan elemen ke-11: 8

Masukkan elemen ke-12: 1

Masukkan elemen ke-13: 1

Masukkan elemen ke-14: 5

Masukkan elemen ke-15: 11

Masukkan elemen ke-16: 14

Masukkan elemen ke-17: 2

Masukkan elemen ke-18: 12

Masukkan elemen ke-19: 13

Masukkan elemen ke-20: 14

Array yang telah diurutkan: 1 1 1 2 2 3 4 5 5 6 7 7 8 8 10 11 12 13 14 14

46962755 microseconds

Kompleksitas waktu:

Durasi waktu yang dibutuhkan untuk 20 input: 46962755 ms = 46.962755 s

Studi Kasus 2: SELECTION SORT

#include <iostream>

#include <chrono>

using namespace std;

using namespace std::chrono;

void selectionSort (int arr[], int n){

int i, j;

for (i = 0; i < n; ++i){

for (j = i+1; j < n; ++j){

if (arr[i] > arr[j]){

arr[i] = arr[i]+arr[j];

arr[j] = arr[i]-arr[j];

arr[i] = arr[i]-arr[j];

}

}

}

}

int main(){

int n, i;

high\_resolution\_clock::time\_point t1 = high\_resolution\_clock::now();

cout<<"Masukkan jumlah elemen data yang ingin diurutkan: ";

cin>>n;

int arr[n];

for(i = 0; i < n; i++){

cout<<"Masukkan elemen ke-"<<i+1<<": ";

cin>>arr[i];

}

selectionSort(arr, n);

cout<<"\nArray yang telah diurutkan: ";

for (i = 0; i < n; i++) cout<<" "<<arr[i];

high\_resolution\_clock::time\_point t2 = high\_resolution\_clock::now();

auto duration = duration\_cast<microseconds>( t2 - t1 ).count();

cout<<endl<<duration<<" microseconds" <<endl;

}

Output:

Masukkan jumlah elemen data yang ingin diurutkan: 20

Masukkan elemen ke-1: 9

Masukkan elemen ke-2: 8

Masukkan elemen ke-3: 7

Masukkan elemen ke-4: 6

Masukkan elemen ke-5: 5

Masukkan elemen ke-6: 4

Masukkan elemen ke-7: 3

Masukkan elemen ke-8: 4

Masukkan elemen ke-9: 10

Masukkan elemen ke-10: 1

Masukkan elemen ke-11: 2

Masukkan elemen ke-12: 3

Masukkan elemen ke-13: 4

Masukkan elemen ke-14: 5

Masukkan elemen ke-15: 1

Masukkan elemen ke-16: 2

Masukkan elemen ke-17: 3

Masukkan elemen ke-18: 11

Masukkan elemen ke-19: 13

Masukkan elemen ke-20: 12

Array yang telah diurutkan: 1 1 2 2 3 3 3 4 4 4 5 5 6 7 8 9 10 11 12 13

57856651 microseconds

Kompleksitas waktu:

Durasi waktu yang dibutuhkan untuk 20 input: 57856651 ms = 57.856651 s

Studi Kasus 3: INSERTION SORT

#include <iostream>

#include <chrono>

using namespace std;

using namespace std::chrono;

struct list {

int data;

list \*next;

};

list\* InsertinList(list \*head, int n){

list \*newnode = new list;

list \*temp = new list;

newnode->data = n;

newnode->next = NULL;

if(head == NULL){

head = newnode;

return head;

}

else {

temp = head;

if(newnode->data < head->data){

newnode->next = head;

head = newnode;

return head;

}

while(temp->next != NULL){

if(newnode->data < (temp->next)->data)

break;

temp=temp->next;

}

newnode->next = temp->next;

temp->next = newnode;

return head;

}

}

int main(){

int n, i, num;

list \*head = new list;

head = NULL;

high\_resolution\_clock::time\_point t1 = high\_resolution\_clock::now();

cout<<"Masukkan jumlah elemen data yang ingin diurutkan: ";

cin>>n;

for(i = 0; i < n; i++){

cout<<"Masukkan elemen ke-"<<i+1<<": ";

cin>>num;

head = InsertinList(head, num);

}

cout<<"\nArray yang telah diurutkan: ";

while(head != NULL){

cout<<" "<<head->data;

head = head->next;

}

high\_resolution\_clock::time\_point t2 = high\_resolution\_clock::now();

auto duration = duration\_cast<microseconds>( t2 - t1 ).count();

cout<<endl<<duration<<" microseconds"<<endl;

}

Output:

Masukkan jumlah elemen data yang ingin diurutkan: 20

Masukkan elemen ke-1: 3

Masukkan elemen ke-2: 4

Masukkan elemen ke-3: 1

Masukkan elemen ke-4: 2

Masukkan elemen ke-5: 3

Masukkan elemen ke-6: 4

Masukkan elemen ke-7: 0

Masukkan elemen ke-8: 9

Masukkan elemen ke-9: 8

Masukkan elemen ke-10: 7

Masukkan elemen ke-11: 1

Masukkan elemen ke-12: 2

Masukkan elemen ke-13: 3

Masukkan elemen ke-14: 4

Masukkan elemen ke-15: 5

Masukkan elemen ke-16: 15

Masukkan elemen ke-17: 14

Masukkan elemen ke-18: 13

Masukkan elemen ke-19: 12

Masukkan elemen ke-20: 11

Array yang telah diurutkan: 0 1 1 2 2 3 3 3 4 4 4 5 7 8 9 11 12 13 14

15

42577513 microseconds

Kompleksitas waktu:

Durasi waktu yang dibutuhkan untuk 20 input: 42577513 ms = 42.577513 s

Studi Kasus 4: BUBBLE SORT

#include <iostream>

#include <chrono>

using namespace std;

using namespace std::chrono;

void bubbleSort (int arr[], int n){

int i, j;

for (i = 0; i < n; ++i){

for (j = 0; j < n-i-1; ++j){

if (arr[j] > arr[j+1]){

arr[j] = arr[j]+arr[j+1];

arr[j+1] = arr[j]-arr[j + 1];

arr[j] = arr[j]-arr[j + 1];

}

}

}

}

int main(){

int n, i;

high\_resolution\_clock::time\_point t1 = high\_resolution\_clock::now();

cout<<"Masukkan jumlah elemen data yang ingin diurutkan: ";

cin>>n;

int arr[n];

for(i = 0; i < n; i++){

cout<<"Masukkan elemen ke-"<<i+1<<": ";

cin>>arr[i];

}

bubbleSort(arr, n);

cout<<"\nArray yang telah diurutkan: ";

for (i = 0; i < n; i++){

cout<<" "<<arr[i];

}

high\_resolution\_clock::time\_point t2 = high\_resolution\_clock::now();

auto duration = duration\_cast<microseconds>( t2 - t1 ).count();

cout<<endl<<duration<<" microseconds"<<endl;

}

int main(){

int n, i, num;

list \*head = new list;

head = NULL;

high\_resolution\_clock::time\_point t1 = high\_resolution\_clock::now();

cout<<"Masukkan jumlah elemen data yang ingin diurutkan: ";

cin>>n;

for(i = 0; i < n; i++){

cout<<"Masukkan elemen ke-"<<i+1<<": ";

cin>>num;

head = InsertinList(head, num);

}

cout<<"\nArray yang telah diurutkan: ";

while(head != NULL){

cout<<" "<<head->data;

head = head->next;

}

high\_resolution\_clock::time\_point t2 = high\_resolution\_clock::now();

auto duration = duration\_cast<microseconds>( t2 - t1 ).count();

cout<<endl<<duration<<" microseconds"<<endl;

}

Output:

Masukkan jumlah elemen data yang ingin diurutkan: 20

Masukkan elemen ke-1: 9

Masukkan elemen ke-2: 8

Masukkan elemen ke-3: 7

Masukkan elemen ke-4: 1

Masukkan elemen ke-5: 3

Masukkan elemen ke-6: 2

Masukkan elemen ke-7: 4

Masukkan elemen ke-8: 5

Masukkan elemen ke-9: 6

Masukkan elemen ke-10: 7

Masukkan elemen ke-11: 8

Masukkan elemen ke-12: 9

Masukkan elemen ke-13: 8

Masukkan elemen ke-14: 11

Masukkan elemen ke-15: 13

Masukkan elemen ke-16: 12

Masukkan elemen ke-17: 11

Masukkan elemen ke-18: 14

Masukkan elemen ke-19: 15

Masukkan elemen ke-20: 16

Array yang telah diurutkan: 1 2 3 4 5 6 7 7 8 8 8 9 9 11 11 12 13 14 15 16

35236342 microseconds

Kompleksitas waktu:

Durasi waktu yang dibutuhkan untuk 20 input: 35236342 ms = 35.236342 s