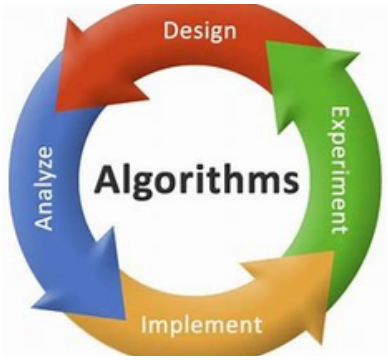
UPES



DAA LAB-2 Report File

Name: Divyansh Sundriyal

Batch : B-33

Sap Id: 590014264

REPOSITRY

https://github.com/Divxcode-177/DAA_LAB_DIVYANSH_SUNDRIYAL_590014264

LAB-2

Lab Experiment 2 – Merge Sort Algorithm Performance Analysis

Objective:

Implement merge sort using any of the programming language (C, C++, Java). Make one pdf, including code, 10 different test cases and their screenshots. Also attach plagiarism report in the end.

CODE :

```
public class MergeSortCode{

 public static void Merging(int mainArray[], int start, int middle, int end) {
  int leftSize = middle - start + 1;
  int rightSize = end - middle;

  int left[] = new int[leftSize];
  int right[] = new int[rightSize];

  for (int left1 = 0; left1 < leftSize; left1++)
   left[left1] = mainArray[start + left1];
  for (int right1 = 0; right1 < rightSize; right1++)
   right[right1] = mainArray[middle + 1 + right1];

  int left1 = 0, right1 = 0, k = start;

  while (left1 < leftSize && right1 < rightSize) {
   if (left[left1] <= right[right1]) {
    mainArray[k++] = left[left1++];
   } else {
    mainArray[k++] = right[right1++];
   }
  }

  while (left1 < leftSize) mainArray[k++] = left[left1++];
  while (right1 < rightSize) mainArray[k++] = right[right1++];
 }

 public static void Mergesortcode(int mainArray[], int start, int end) {
  if (start < end) {
   int middle = (end - start) / 2 + start;
   Mergesortcode(mainArray, start, middle);
   Mergesortcode(mainArray, middle + 1, end);
   Merging(mainArray, start, middle, end);
  }
 }
```

CODE :

```
public class MergeSortExample {


  static void Merging(int arr[], int start, int middle, int end) {
     int n1 = middle - start + 1;
     int n2 = end - middle;

     int left[] = new int[n1];
     int right[] = new int[n2];

     for (int left1 = 0; left1 < n1; left1++)
        left[left1] = arr[start + left1];
     for (int right1 = 0; right1 < n2; right1++)
        right[right1] = arr[middle + 1 + right1];

     int left1 = 0, right1 = 0, k = start;

     while (left1 < n1 && right1 < n2) {
        if (left[left1] <= right[right1]) {
           arr[k++] = left[left1++];
        } else {
           arr[k++] = right[right1++];
        }
     }

     while (left1 < n1) arr[k++] = left[left1++];
     while (right1 < n2) arr[k++] = right[right1++];
  }
```

CODE :

```
static void Mergesortcode(int arr[], int start, int end) {
if (start < end) {
int middle = (end - start) / 2 + start;
Mergesortcode(arr, start, middle);
Mergesortcode(arr, middle + 1, end);
Merging(arr, start, middle, end);
}
}


static void ArrayPrintingFunction(int arr[]) {
for (int i = 0; i < arr.length; i++) {
System.out.print(arr[i] + "\t");
}
System.out.println();
}

public static void main(String[] args) {


int TestCaseArray1[] = {1, 2, 3, 4, 5, 6};
System.out.println("Test Case 1 - Already Sorted Array");
System.out.print("Before: ");
ArrayPrintingFunction(TestCaseArray1);
Mergesortcode(TestCaseArray1, 0, TestCaseArray1.length - 1);
System.out.print("After: ");
ArrayPrintingFunction(TestCaseArray1);
```

CODE :

```
int TestCaseArray2[] = {9, 8, 7, 6, 5, 4};
System.out.println("Test Case 2 - Reverse Order Array");
System.out.print("Before: ");
ArrayPrintingFunction(TestCaseArray2);
Mergesortcode(TestCaseArray2, 0, TestCaseArray2.length - 1);
System.out.print("After: ");
ArrayPrintingFunction(TestCaseArray2);
```

```
int TestCaseArray3[] = {10, 1, 14, 17, 2, 3};
System.out.println("Test Case 3 - Random Order Array");
System.out.print("Before: ");
ArrayPrintingFunction(TestCaseArray3);
Mergesortcode(TestCaseArray3, 0, TestCaseArray3.length - 1);
System.out.print("After: ");
ArrayPrintingFunction(TestCaseArray3);

int TestCaseArray4[] = {5, 1, 3, 5, 2, 5};
System.out.println("Test Case 4 - Array With Duplicates");
System.out.print("Before: ");
ArrayPrintingFunction(TestCaseArray4);
Mergesortcode(TestCaseArray4, 0, TestCaseArray4.length - 1);
System.out.print("After: ");
ArrayPrintingFunction(TestCaseArray4);
```

CODE :

```
int TestCaseArray5[] = {42};
System.out.println("Test Case 5 - Single Element Array");
System.out.print("Before: ");
ArrayPrintingFunction(TestCaseArray5);
Mergesortcode(TestCaseArray5, 0, TestCaseArray5.length - 1);
System.out.print("After: ");
ArrayPrintingFunction(TestCaseArray5);
```

```
int TestCaseArray6[] = {99, 11};
System.out.println("Test Case 6 - Two Elements Array");
System.out.print("Before: ");
ArrayPrintingFunction(TestCaseArray6);
Mergesortcode(TestCaseArray6, 0, TestCaseArray6.length - 1);
System.out.print("After: ");
ArrayPrintingFunction(TestCaseArray6);
```

```
int TestCaseArray7[] = {7, 7, 7, 7, 7, 7};
System.out.println("Test Case 7 - All Same Elements Array");
System.out.print("Before: ");
ArrayPrintingFunction(TestCaseArray7);
Mergesortcode(TestCaseArray7, 0, TestCaseArray7.length - 1);
System.out.print("After: ");
ArrayPrintingFunction(TestCaseArray7);
```

CODE :

```
int TestCaseArray8[] = {1000, 500, 2000, 1500, 2500};
System.out.println("Test Case 8 - Large Numbers Array");
System.out.print("Before: ");
ArrayPrintingFunction(TestCaseArray8);
Mergesortcode(TestCaseArray8, 0, TestCaseArray8.length - 1);
System.out.print("After: ");
ArrayPrintingFunction(TestCaseArray8);



int TestCaseArray9[] = {-5, -10, -3, -1, -7};
System.out.println("Test Case 9 - Negative Numbers Array");
System.out.print("Before: ");
ArrayPrintingFunction(TestCaseArray9);
Mergesortcode(TestCaseArray9, 0, TestCaseArray9.length - 1);
System.out.print("After: ");
ArrayPrintingFunction(TestCaseArray9);

int TestCaseArray10[] = {-2, 4, 0, -9, 7, 3};
System.out.println("Test Case 10 - Mix Positive & Negative Array");
System.out.print("Before: ");
ArrayPrintingFunction(TestCaseArray10);
Mergesortcode(TestCaseArray10, 0, TestCaseArray10.length - 1);
System.out.print("After: ");
ArrayPrintingFunction(TestCaseArray10);

 }
}
```

OUTPUT:

```
PS C:\Users\dell\Documents\GitHub\DAA_LAB_DIVYANSH_SUNDRIYAL_590014264\Lab2\Code> java MergeSortCode.java
Test Case 1 - Already Sorted Array
Before: 1       2       3       4       5       6
After:  1       2       3       4       5       6
Test Case 2 - Reverse Order Array
Before: 9       8       7       6       5       4
After:  4       5       6       7       8       9
Test Case 3 - Random Order Array
Before: 10      1       14      17      2       3
After:  1       2       3       10      14      17
Test Case 4 - Array With Duplicates
Before: 5       1       3       5       2       5
After:  1       2       3       5       5       5
Test Case 5 - Single Element Array
Before: 42
After:  42
Test Case 6 - Two Elements Array
Before: 99      11
After:  11      99
Test Case 7 - All Same Elements Array
Before: 7       7       7       7       7       7
After:  7       7       7       7       7       7
Test Case 8 - Large Numbers Array
Before: 1000    500     2000    1500    2500
After:  500     1000    1500    2000    2500
Test Case 9 - Negative Numbers Array
Before: -5      -10     -3      -1      -7
After:  -10     -7      -5      -3      -1
Test Case 10 - Mix Positive & Negative Array
Before: -2      4       0       -9      7       3
After:  -9      -2      0       3       4       7
```

PLAGIARISM SCAN REPORT

Report Generation Date: 30-08-25

Words: 769

Characters: 8724

Excluded URL : N/A

5%	95%
Plagiarism	Unique
2	**36**
Plagiarized Sentences	Unique Sentences

Content Checked for Plagiarism

DAA lAB-2
Report File
Name: Divyansh Sundriyal
Batch : B-33
Sap Id: 590014264
https://github.com/Divxcode-177/DAA_LAB_DIVYANSH_SUNDRIYAL_590014264
repositry
Objective:
Implement merge sort using any of the
programming language (C, C++, Java).
Make one pdf, including code, 10 different
test cases and their screenshots. Also
attach plagiarism report in the end.

LAB-2
Lab Experiment 2 – Merge Sort
Algorithm Performance Analysis

CODE :

```
public class MergeSortCode{
public static void Merging(int mainArray[], int start, int middle, int end) {
int leftSize = middle - start + 1;
int rightSize = end - middle;
int left[] = new int[leftSize];
int right[] = new int[rightSize];
for (int left1 = 0; left1 < leftSize; left1++)
left[left1] = mainArray[start + left1];
for (int right1 = 0; right1 < rightSize; right1++)
right[right1] = mainArray[middle + 1 + right1];
int left1 = 0, right1 = 0, k = start;
while (left1 < leftSize && right1 < rightSize) {
if (left[left1] <= right[right1]) {
mainArray[k++] = left[left1++];
} else {
mainArray[k++] = right[right1++];
}
}
while (left1 < leftSize) mainArray[k++] = left[left1++];
while (right1 < rightSize) mainArray[k++] = right[right1++];
}
public static void Mergesortcode(int mainArray[], int start, int end) {
if (start < end) {
int middle = (end - start) / 2 + start;
Mergesortcode(mainArray, start, middle);
Mergesortcode(mainArray, middle + 1, end);
Merging(mainArray, start, middle, end);
}
}
```

CODE :

```
public class MergeSortExample {
// Renamed function merge -> Merging
static void Merging(int arr[], int start, int middle, int end) {
int n1 = middle - start + 1;
int n2 = end - middle;
int left[] = new int[n1];
int right[] = new int[n2];
for (int left1 = 0; left1 < n1; left1++)
left[left1] = arr[start + left1];
for (int right1 = 0; right1 < n2; right1++)
right[right1] = arr[middle + 1 + right1];
int left1 = 0, right1 = 0, k = start;
while (left1 < n1 && right1 < n2) {
if (left[left1] <= right[right1]) {
arr[k++] = left[left1++];
} else {
arr[k++] = right[right1++];
}
}
while (left1 < n1) arr[k++] = left[left1++];
while (right1 < n2) arr[k++] = right[right1++];
}
```

CODE :

```
// Renamed function mergeSort -> Mergesortcode
static void Mergesortcode(int arr[], int start, int end) {
if (start < end) {
int middle = (end - start) / 2 + start;
Mergesortcode(arr, start, middle);
Mergesortcode(arr, middle + 1, end);
Merging(arr, start, middle, end);
}
```

```
}
// Array printing function
static void ArrayPrintingFunction(int arr[]) {
for (int i = 0; i < arr.length; i++) {
System.out.print(arr[i] + "\t");
}
System.out.println();
}
public static void main(String[] args) {
// Test Case 1: Already Sorted
int TestCaseArray1[] = {1, 2, 3, 4, 5, 6};
System.out.println("Test Case 1 - Already Sorted Array");
System.out.print("Before: ");
ArrayPrintingFunction(TestCaseArray1);
Mergesortcode(TestCaseArray1, 0, TestCaseArray1.length - 1);
System.out.print("After: ");
ArrayPrintingFunction(TestCaseArray1);
System.out.println("-----------------------------------");
```

CODE :

```
// Test Case 2: Reverse Order
int TestCaseArray2[] = {9, 8, 7, 6, 5, 4};
System.out.println("Test Case 2 - Reverse Order Array");
System.out.print("Before: ");
ArrayPrintingFunction(TestCaseArray2);
Mergesortcode(TestCaseArray2, 0, TestCaseArray2.length - 1);
System.out.print("After: ");
ArrayPrintingFunction(TestCaseArray2);
System.out.println("-----------------------------------");
// Test Case 3: Random Order
int TestCaseArray3[] = {10, 1, 14, 17, 2, 3};
System.out.println("Test Case 3 - Random Order Array");
System.out.print("Before: ");
ArrayPrintingFunction(TestCaseArray3);
Mergesortcode(TestCaseArray3, 0, TestCaseArray3.length - 1);
```

```
System.out.print("After: ");
ArrayPrintingFunction(TestCaseArray3);
System.out.println("-----------------------------------");
// Test Case 4: Array With Duplicates
int TestCaseArray4[] = {5, 1, 3, 5, 2, 5};
System.out.println("Test Case 4 - Array With Duplicates");
System.out.print("Before: ");
ArrayPrintingFunction(TestCaseArray4);
Mergesortcode(TestCaseArray4, 0, TestCaseArray4.length - 1);
System.out.print("After: ");
ArrayPrintingFunction(TestCaseArray4);
System.out.println("-----------------------------------");
```

CODE :

```
// Test Case 5: Single Element
int TestCaseArray5[] = {42};
System.out.println("Test Case 5 - Single Element Array");
System.out.print("Before: ");
ArrayPrintingFunction(TestCaseArray5);
Mergesortcode(TestCaseArray5, 0, TestCaseArray5.length - 1);
System.out.print("After: ");
ArrayPrintingFunction(TestCaseArray5);
System.out.println("-----------------------------------");
// Test Case 6: Two Elements
int TestCaseArray6[] = {99, 11};
System.out.println("Test Case 6 - Two Elements Array");
System.out.print("Before: ");
ArrayPrintingFunction(TestCaseArray6);
Mergesortcode(TestCaseArray6, 0, TestCaseArray6.length - 1);
System.out.print("After: ");
ArrayPrintingFunction(TestCaseArray6);
System.out.println("-----------------------------------");
// Test Case 7: All Same Elements
int TestCaseArray7[] = {7, 7, 7, 7, 7, 7};
System.out.println("Test Case 7 - All Same Elements Array");
System.out.print("Before: ");
```

```
ArrayPrintingFunction(TestCaseArray7);
Mergesortcode(TestCaseArray7, 0, TestCaseArray7.length - 1);
System.out.print("After: ");
ArrayPrintingFunction(TestCaseArray7);
System.out.println("-----------------------------------");
```

CODE :

```
// Test Case 8: Large Numbers
int TestCaseArray8[] = {1000, 500, 2000, 1500, 2500};
System.out.println("Test Case 8 - Large Numbers Array");
System.out.print("Before: ");
ArrayPrintingFunction(TestCaseArray8);
Mergesortcode(TestCaseArray8, 0, TestCaseArray8.length - 1);
System.out.print("After: ");
ArrayPrintingFunction(TestCaseArray8);
System.out.println("-----------------------------------");
// Test Case 9: Negative Numbers
int TestCaseArray9[] = {-5, -10, -3, -1, -7};
System.out.println("Test Case 9 - Negative Numbers Array");
System.out.print("Before: ");
ArrayPrintingFunction(TestCaseArray9);
Mergesortcode(TestCaseArray9, 0, TestCaseArray9.length - 1);
System.out.print("After: ");
ArrayPrintingFunction(TestCaseArray9);
System.out.println("-----------------------------------");
// Test Case 10: Mix of Positive & Negative Numbers
int TestCaseArray10[] = {-2, 4, 0, -9, 7, 3};
System.out.println("Test Case 10 - Mix Positive & Negative Array");
System.out.print("Before: ");
ArrayPrintingFunction(TestCaseArray10);
Mergesortcode(TestCaseArray10, 0, TestCaseArray10.length - 1);
System.out.print("After: ");
ArrayPrintingFunction(TestCaseArray10);
System.out.println("-----------------------------------");
```

```
}
}
```

OUTPUT:

Matched Sources :

Pastebinpastebin.com › KGawzXa8void _Merge (int* pArray, int start, int middle, int end ...

void _Merge (int* pArray, int start, int middle, int end) { // initialize the left array int leftSize = middle - start +1; // +1 because of sentinal int* pLeft ...

50%

https://pastebin.com/KGawzXa8/

java中for (int i : arr) 的含义_for (int i : arr)-CSDN博客

Aug 15, 2023 · public class Apptest { public static void main(String[] args) { int [] arr = {4, 5, 6, 7}; for (int i = 0; i < arr.length; i++){ System.out.println(" 数组元素 : " + arr[i]); } }

6%

https://blog.csdn.net/weixin_51220009/article/details/132291531