**PERBANDINGAN ALGORITMA MERGE SORT, QUICK SORT, SELECTION SORT, DAN INSERTION SORT**

**Laporan Tugas Kecil 2**

**IF 2211 - Strategi Algoritma**



**Oleh:**

**AHMAD AIDIN**

**NIM: 13513020**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO & INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**Maret 2017**

**A. Source Code (C++)**

|  |
| --- |
| Main.cpp |
| #include "MergeSorter.hpp"  #include "QuickSorter.hpp"  #include "SelectionSorter.hpp"  #include "InsertionSorter.hpp"  #include <stdlib.h>  #include <iostream>  #include <time.h>  #include <stdio.h>  using namespace std;  int main() {  bool keluar = false;  int size;  while(!keluar){  cout << "ukuran table yang diinginkan: ";  cin >> size;  srand(time(NULL));  int oldTable[size];  for (int i = 0; i < size; i++) {  oldTable[i]=(rand()%(size\*10)+1);  }  int opt=1;  while(opt!=5){  cout<<"Jenis Sort (default : Merge Sort): "<<endl;  cout<<"1: Merge Sort"<<endl;  cout<<"2: Quick Sort"<<endl;  cout<<"3: Selection Sort"<<endl;  cout<<"4: Insertion Sort"<<endl;  cout<<"5: Pilih ukuran lain"<<endl;  cout<<"0: Keluar"<<endl;  cout<<"Pilihanmu: ";  cin>>opt;  if(opt==0){  opt=5;  keluar = true;  } else if(opt!=5){  int table[size];  for (int i = 0; i < size; i++) {  table[i]=oldTable[i];  }  cout<<endl;  cout<<"Tabel awal: "<<endl;  for (int i = 0; i < size; i++) {  cout<<table[i]<<"|";  }  clock\_t start = clock();  switch (opt) {  case 1: {  MergeSorter::sort(table,0,size-1);  }  case 2: {  QuickSorter::sort(table, 0,size-1);  }  case 3: {  SelectionSorter::sort(table, size);  }  case 4: {  InsertionSorter::sort(table, size);  }  }  clock\_t end = clock();  float seconds = (float)(end - start)\*1000 / CLOCKS\_PER\_SEC;  cout<<endl<<endl;  cout<<"Tabel hasil pengurutan: "<<endl;  for (int i = 0; i < size; i++) {  cout<<table[i]<<"|";  }  cout<<endl<<endl;  printf ("Waktu eksekusi: %.3lf ms.\n", seconds );  cout<<endl;  }  else {  keluar = false;  }  }  }  return 0;  } |

|  |
| --- |
| MergeSorter.hpp |
| #pragma once  class MergeSorter {  public:  MergeSorter() {  }  MergeSorter(MergeSorter &m) {  }  static void sort(int \*table, int i, int j) {  int k;  if (i<j) {  k = (i + j) / 2;  sort(table, i, k);  sort(table, k + 1, j);  merge(table,i,k,j);  }  }  static void merge(int\* table, int left, int mid, int right) {  int tempTable[right-left+1];  int iteratorL, iteratorR;  iteratorL = left;  iteratorR = mid + 1;  int i=0;  while ((iteratorL <= mid) && (iteratorR <= right)) {  if (table[iteratorL] <= table[iteratorR]) {  tempTable[i]=table[iteratorL];  iteratorL++;  }  else {  tempTable[i]=table[iteratorR];  iteratorR++;  }  i++;  }  while (iteratorL <= mid) {  tempTable[i]=table[iteratorL];  iteratorL++;  i++;  }//salin sisa table bagian kiri ke tempTable jika ada  while (iteratorR <= right) {  tempTable[i]=table[iteratorR];  iteratorR++;  i++;  }//salin sisa table bagian kanan ke tempTable jika ada  for (int j = left; j <= right; j++) {  table[j] = tempTable[j-left];  }//salin elemen tempTable ke table  }  }; |

|  |
| --- |
| QuickSorter.hpp |
| #pragma once  #include <algorithm>  #include <map>  #include <iostream>  using namespace std;  class QuickSorter {  public:  QuickSorter() {  }  QuickSorter(QuickSorter &m) {  }  static void sort(int \*table, int left, int right) {  int i = left;  int j = right;  //create partition  int pivot = table[(left + right) / 2];  while (i<=j){  while(table[i]<pivot)  i++;  while(table[j]>pivot)  j--;  if(i<=j){  swap(table[i],table[j]);  i++;  j--;  }  }  //recursive  if (left<j) {  sort(table, left, j);  }  if(i<right){  sort(table,i,right);  }  }  }; |

|  |
| --- |
| SelectionSorter.hpp |
| #pragma once  class SelectionSorter {  public:  SelectionSorter() {  }  SelectionSorter(SelectionSorter &m) {  }  static void sort(int \*table, int size) {  for(int i = 0; i<size-1; i++){  for(int j=i+1; j<size; j++){  if(table[i]>table[j]){  swap(table[i],table[j]);  }  }  }  }  }; |

|  |
| --- |
| InsertionSorter.hpp |
| #pragma once  class InsertionSorter {  public:  InsertionSorter() {  }  InsertionSorter(InsertionSorter &m) {  }  static void sort(int \*table, int size) {  for(int i = 1; i<size; i++){  int j=i;  while (j>0 && table[j]<table[j-1]) {  swap(table[j],table[j-1]);  j--;  }  }  }  }; |

**B. Contoh Input dan Output**

Data input dan output yang dipergunakan pada contoh hanya berjumlah 100 agar mudah dibaca

**Algoritma: Merge Sort**

Tabel awal:

5, 640, 327, 178, 331, 322, 861, 848, 935, 706, 205, 205, 562, 545, 847, 844, 570, 640, 565, 147, 738, 333, 738, 465, 657, 895, 661, 960, 731, 862, 264, 87, 501, 590, 617, 184, 263, 477, 31, 197, 182, 587, 754, 95, 131, 600, 939, 53, 592, 855, 551, 329, 540, 640, 793, 196, 887, 805, 508, 969, 667, 771, 55, 519, 361, 23, 702, 975, 851, 85, 524, 32, 671, 277, 479, 154, 228, 769, 558, 819, 975, 108, 500, 514, 747, 644, 62, 985, 449, 569, 953, 467, 339, 360, 337, 51, 382, 39, 26, 233

Tabel hasil pengurutan:

5, 23, 26, 31, 32, 39, 51, 53, 55, 62, 85, 87, 95, 108, 131, 147, 154, 178, 182, 184, 196, 197, 205, 205, 228, 233, 263, 264, 277, 322, 327, 329, 331, 333, 337, 339, 360, 361, 382, 449, 465, 467, 477, 479, 500, 501, 508, 514, 519, 524, 540, 545, 551, 558, 562, 565, 569, 570, 587, 590, 592, 600, 617, 640, 640, 640, 644, 657, 661, 667, 671, 702, 706, 731, 738, 738, 747, 754, 769, 771, 793, 805, 819, 844, 847, 848, 851, 855, 861, 862, 887, 895, 935, 939, 953, 960, 969, 975, 975, 985

Waktu eksekusi: 0.115 ms.

**Algoritma: Quick Sort**

Tabel awal:

423, 684, 133, 16, 125, 593, 814, 538, 628, 370, 222, 424, 272, 636, 563, 695, 883, 118, 183, 409, 511, 544, 524, 767, 631, 624, 580, 183, 954, 478, 507, 376, 513, 639, 744, 637, 583, 557, 527, 562, 278, 748, 986, 549, 735, 900, 244, 618, 369, 426, 378, 879, 970, 253, 997, 952, 876, 929, 134, 182, 758, 993, 909, 271, 631, 4, 259, 566, 912, 137, 127, 189, 885, 464, 738, 971, 363, 333, 588, 83, 110, 317, 961, 431, 569, 310, 382, 797, 238, 868, 330, 347, 860, 238, 617, 842, 242, 228, 407, 153,

Tabel hasil pengurutan:

4, 16, 83, 110, 118, 125, 127, 133, 134, 137, 153, 182, 183, 183, 189, 222, 228, 238, 238, 242, 244, 253, 259, 271, 272, 278, 310, 317, 330, 333, 347, 363, 369, 370, 376, 378, 382, 407, 409, 423, 424, 426, 431, 464, 478, 507, 511, 513, 524, 527, 538, 544, 549, 557, 562, 563, 566, 569, 580, 583, 588, 593, 617, 618, 624, 628, 631, 631, 636, 637, 639, 684, 695, 735, 738, 744, 748, 758, 767, 797, 814, 842, 860, 868, 876, 879, 883, 885, 900, 909, 912, 929, 952, 954, 961, 970, 971, 986, 993, 997,

Waktu eksekusi: 0.086 ms.

**Algoritma: Selection Sort**

Tabel awal:

688, 12, 249, 140, 499, 718, 543, 293, 60, 944, 397, 378, 353, 191, 963, 168, 255, 929, 954, 399, 157, 246, 962, 502, 416, 299, 248, 870, 72, 732, 202, 112, 743, 802, 251, 593, 519, 145, 885, 930, 441, 282, 308, 145, 824, 622, 664, 430, 902, 618, 828, 58, 863, 141, 559, 630, 440, 159, 851, 863, 242, 52, 974, 336, 205, 577, 928, 723, 73, 813, 653, 865, 446, 312, 9, 621, 933, 673, 50, 835, 642, 229, 244, 504, 369, 155, 133, 160, 313, 335, 23, 906, 386, 348, 241, 942, 276, 168, 16, 701

Tabel hasil pengurutan:

9, 12, 16, 23, 50, 52, 58, 60, 72, 73, 112, 133, 140, 141, 145, 145, 155, 157, 159, 160, 168, 168, 191, 202, 205, 229, 241, 242, 244, 246, 248, 249, 251, 255, 276, 282, 293, 299, 308, 312, 313, 335, 336, 348, 353, 369, 378, 386, 397, 399, 416, 430, 440, 441, 446, 499, 502, 504, 519, 543, 559, 577, 593, 618, 621, 622, 630, 642, 653, 664, 673, 688, 701, 718, 723, 732, 743, 802, 813, 824, 828, 835, 851, 863, 863, 865, 870, 885, 902, 906, 928, 929, 930, 933, 942, 944, 954, 962, 963, 974

Waktu eksekusi: 0.141 ms.

**Algoritma: Insertion Sort**

Tabel awal:

799, 870, 759, 388, 125, 584, 861, 699, 776, 967, 85, 246, 405, 219, 46, 191, 955, 130, 809, 705, 851, 8, 743, 47, 376, 927, 539, 554, 650, 480, 247, 448, 702, 5, 835, 178, 940, 695, 876, 715, 13, 312, 312, 418, 530, 357, 960, 484, 486, 768, 540, 689, 127, 282, 735, 503, 560, 625, 408, 209, 105, 654, 9, 806, 11, 195, 335, 302, 242, 210, 17, 254, 521, 328, 23, 402, 37, 982, 237, 874, 102, 776, 914, 228, 410, 648, 82, 969, 625, 489, 530, 729, 495, 890, 886, 857, 84, 220, 158, 325,

Tabel hasil pengurutan:

5, 8, 9, 11, 13, 17, 23, 37, 46, 47, 82, 84, 85, 102, 105, 125, 127, 130, 158, 178, 191, 195, 209, 210, 219, 220, 228, 237, 242, 246, 247, 254, 282, 302, 312, 312, 325, 328, 335, 357, 376, 388, 402, 405, 408, 410, 418, 448, 480, 484, 486, 489, 495, 503, 521, 530, 530, 539, 540, 554, 560, 584, 625, 625, 648, 650, 654, 689, 695, 699, 702, 705, 715, 729, 735, 743, 759, 768, 776, 776, 799, 806, 809, 835, 851, 857, 861, 870, 874, 876, 886, 890, 914, 927, 940, 955, 960, 967, 969, 982,

Waktu eksekusi: 0.087 ms.

**C. Perbandingan Waktu Eksekusi**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Banyak data | Merge Sort | Quick Sort | Selection Sort | Insertion Sort |
| 1 | 1.000 | 0,505 | 0,454 | 12,194 | 6,470 |
| 2 | 5.000 | 3,396 | 2,301 | 176,223 | 66,134 |
| 3 | 10.000 | 6,566 | 4,689 | 516,513 | 263,413 |
| 4 | 50.000 | 32,801 | 17,907 | 12.389,828 | 6.711,627 |
| 5 | 100.000 | 60,067 | 30,834 | 50.025,859 | 26.665,461 |
| 6 | 500.000 | 210,819 | 158,975 | Sangat lama | Sangat lama |
| 7 | 1.000.000 | 378,228 | 266,180 | Sangat lama | Sangat lama |

Perbedaan yang signifikan adalah pada waktu eksekusi Merge Sort dan Quick Sort dibanding dengan Selection Sort dan Insertion Sort. Merge Sort dan Quick Sort memiliki waktu eksekusi lebih cepat dibanding Selection Sort dan Insertion Sort karena Merge Sort dan Quick Sort memiliki kompleksitas waktu O(n log n) sedangkan Selection Sort dan Insertion Sort memiliki kompleksitas waktu O(n2).

Quick sort memiliki waktu eksekusi bukan lebih cepat dibandingkan Merge Sort karena Merge sort membuat array baru setiap iterasi sehingga memerlukan waktu menulis dan membaca ke RAM lebih banyak dari Quick Sort.

Insertion Sort memiliki waktu eksekusi yang lebih cepat dibanding Selection Sort karena pada Selection Sort iterasi level paling dasarnya mengharuskan hingga ujung array sedangkan Insertion Sort tidak.

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| 1. Program berhasil dikompilasi | √ |  |
| 1. Program berhasil *running* | √ |  |
| 1. Program dapat membaca koleksi data random dan menuliskan koleksi data terurut. | √ |  |
| 1. Laporan berisi hasil perbandingan kecepatan eksekusi dan analisisnya | √ |  |