**Giả sử sử dụng hàm Random trong C/C++ để phát sinh ngẫu nhiên dãy số nguyên có kích thước 100, 1.000, 5.000, 10.000, 50.000 số. Hãy cài đặt các giải thuật sắp xếp để sắp dãy số nguyên đã cho theo thứ tự tăng dần và nhận xét về thời gian thực hiện của các giải thuật.**

**\*Lưu ý:** Do sử dụng hàm clock() nên thời gian chạy giữa các máy là khác nhau tuỳ thuộc vào CPU và các yếu tố khác, các bài test dưới đây đều được thực hiện trên cùng 1 máy. Thông số máy như sau:

- Model: Apple Macbook Air 2019.

- Memory: 8GB of 2133MHz LPDDR3 onboard memory.

- CPU: 1.6GHz dual-core Intel Core i5 8210Y, Turbo Boost up to 3.6GHz, with 4MB L3 cache

- GPU: Intel UHD Graphics 617

- OS: MacOS 13.2.1 Ventura

- Phần mềm code: Xcode

**\*Code:**

|  |
| --- |
| #include <iostream>  #include <stdio.h>  #include <stdlib.h>  #include <time.h>  #include <iomanip>    **using** **namespace** std;  **int** random(**int** minN, **int** maxN)  {  **return** minN + rand() % (maxN + 1 - minN);  }  **void** input\_arr(**int** a[], **int** &n)  {  cin >> n;  cout << "n = " << n;  **int** r;  srand((**int**)time(0));  **for** (**int** i = 0; i < n; i++)  {  r = random(-10000,10000);  a[i] = r;  }  }  **void** swap(**int** &x, **int** &y)  {  **int** temp = x;  x = y;  y = temp;  }    **void** SelectionSort(**int** arr[], **int** n)  {  **int** i, j, min\_idx;  **for** (i = 0; i < n-1; i++)  {  min\_idx = i;  **for** (j = i+1; j < n; j++)  **if** (arr[j] < arr[min\_idx])  min\_idx = j;  swap(arr[min\_idx], arr[i]);  }  }  **void** InterchangeSort(**int** a[], **int** n)  {  **int** i,j;  **for**(i=0; i<n-1; i++)  {  **for**(j=i+1; j<n; j++)  {  **if**(a[i] > a[j])  swap(a[i],a[j]);  }  }  }  **void** InsertionSort(**int** arr[], **int** n)  {  **int** i, key, j;  **for** (i = 1; i < n; i++)  {  key = arr[i];  j = i - 1;  **while** (j >= 0 && arr[j] > key)  {  arr[j + 1] = arr[j];  j = j - 1;  }  arr[j + 1] = key;  }  }  **void** BubbleSort(**int** arr[], **int** n)  {  **int** i, j;  **for** (i = 0; i < n - 1; i++)  **for** (j = n - 1; j > i; j--)  **if** (arr[j] < arr[j-1])  swap(arr[j], arr[j-1]);  }  **void** heapify(**int** arr[], **int** n, **int** i)  {  **int** max =i;  **int** l = i\*2 +1;  **int** r = l+1;  **if**(l < n **and** arr[l] > arr[max]){  max = l;  }    **if** (r < n **and** arr[r] > arr[max]){  max = r;  }    **if**(max != i){  swap(arr[i], arr[max]);  heapify(arr ,n , max);  }    }  **void** HeapSort(**int** arr[], **int** n){      **for**(**int** i = n/2 - 1; i>=0; i--)  heapify(arr,n, i);    **for**(**int** j = n-1 ; j>0; j--){  swap(arr[0], arr[j] );  heapify(arr, j, 0);  }  }  **void** ShellSort(**int** a[], **int** n)  {  **int** interval, i, j, temp;  **for**(interval = n/2; interval > 0; interval /= 2){  **for**(i = interval; i < n; i++){  temp = a[i];  **for**(j = i; j >= interval && a[j - interval] > temp; j -= interval){  a[j] = a[j - interval];  }  a[j] = temp;  }  }  }  **void** QuickSort(**int** a[], **int** left, **int** right)  {  **int** i, j, x;  x = a[(left+right)/2];  i = left;  j = right;  **while** (i < j)  {  **while** (a[i] < x)  i++;  **while** (a[j] > x)  j--;  **if** (i <= j)  {  swap(a[i], a[j]);  i++;  j--;  }  }  **if** (left < j)  QuickSort(a, left, j);  **if** (i < right)  QuickSort(a, i, right);  }  **void** merge(**int** arr[], **int** l, **int** m, **int** r)  {  **int** i, j, k;  **int** n1 = m - l + 1;  **int** n2 = r - m;  **int** L[n1], R[n2];  **for** (i = 0; i < n1; i++)  L[i] = arr[l + i];  **for** (j = 0; j < n2; j++)  R[j] = arr[m + 1+ j];  i = 0;  j = 0;  k = l;  **while** (i < n1 && j < n2)  {  **if** (L[i] <= R[j])  {  arr[k] = L[i];  i++;  }  **else**  {  arr[k] = R[j];  j++;  }  k++;  }  **while** (i < n1)  {  arr[k] = L[i];  i++;  k++;  }  **while** (j < n2)  {  arr[k] = R[j];  j++;  k++;  }  }    **void** MergeSort(**int** arr[], **int** l, **int** r)  {  **if** (l < r)  {    **int** m = l+(r-l)/2;  MergeSort(arr, l, m);  MergeSort(arr, m+1, r);  merge(arr, l, m, r);  }  }  **int** main()  {  **int** a[1000000], n;  **double** time\_use = 0.0;  clock\_t start, end;  input\_arr(a, n);    **int** options;  cout << "Chon thuat toan:" << endl << "1. Selection sort" << endl << "2. Insertion sort" << endl << "3. Interchange sort" << endl << "4. Bubble sort" << endl << "5. Quick sort" << endl << "6. Heap srot" << endl << "7. Shell sort" << endl << "8. Merge sort" << endl;  cout << endl;  cout << "Options: ";  start = clock();  **do**  {  cin >> options;  **if** (options < 1 || options > 8) cout << "Loi, chon lai: ";  } **while** (options < 1 || options > 8);  **switch** (options)  {  **case** 1:  SelectionSort(a, n);  **break**;  **case** 2:  InsertionSort(a, n);  **break**;  **case** 3:  InterchangeSort(a, n);  **break**;  **case** 4:  BubbleSort(a, n);  **break**;  **case** 5:  QuickSort(a, 0, n-1);  **break**;  **case** 6:  HeapSort(a, n);  **break**;  **case** 7:  ShellSort(a, n);  **break**;  **case** 8:  MergeSort(a, 0, n-1);  **break**;  }  end = clock();  time\_use = (**double**)(end - start) / CLOCKS\_PER\_SEC;  cout << endl;  cout << "Thoi gian chay chuong trinh: " << fixed << time\_use << " giây" <<endl;  } |

**\*Với A[100]:**

- Selection Sort:

Text

Description automatically generated with low confidence

- Interchange Sort:

Text

Description automatically generated

- Insertion Sort:

Text

Description automatically generated

- Bubble Sort:

Text

Description automatically generated

- Heap Sort:

Text

Description automatically generated

- Shell Sort:

Text

Description automatically generated

- Quick Sort:

Text

Description automatically generated

- Merge Sort:

**Text

Description automatically generated**

**\*Với A[1000]:**

- Selection Sort:

Text

Description automatically generated

- Interchange Sort:

Text

Description automatically generated

- Insertion Sort:

Text

Description automatically generated

- Bubble Sort:

Text

Description automatically generated

- Heap Sort:

Text

Description automatically generated

- Shell Sort:

Text

Description automatically generated

- Quick Sort:

Text

Description automatically generated

- Merge Sort:

**Text

Description automatically generated**

**\*Với A[5000]:**

- Selection Sort:

Text

Description automatically generated

- Interchange Sort:

Text

Description automatically generated

- Insertion Sort:

Text

Description automatically generated

- Bubble Sort:

Text

Description automatically generated

- Heap Sort:

Text

Description automatically generated

- Shell Sort:

Text

Description automatically generated

- Quick Sort:

Text

Description automatically generated

- Merge Sort:

**Text

Description automatically generated**

**\*Với A[10000]:**

- Selection Sort:

Text

Description automatically generated

- Interchange Sort:

Text

Description automatically generated

- Insertion Sort:

Text

Description automatically generated

- Bubble Sort:

Text

Description automatically generated

- Heap Sort:

Text

Description automatically generated

- Shell Sort:

Text

Description automatically generated

- Quick Sort:

Text

Description automatically generated

- Merge Sort:

Text

Description automatically generated

**\*Với A[50000]:**

- Selection Sort:

Text

Description automatically generated

- Interchange Sort:

Text

Description automatically generated

- Insertion Sort:

Text

Description automatically generated

- Bubble Sort:

Text

Description automatically generated

- Heap Sort:

Text

Description automatically generated

- Shell Sort:

Text

Description automatically generated

- Quick Sort:

Text

Description automatically generated

- Merge Sort:

**Text

Description automatically generated**