Частина 2 : тестування різних алгоритмів сортування

Алгоритми – сортування комірками та сортування вставками

Очікуваний час О(n + k) та О(n^2) відповідно

Вхідні дані: випадкові числа від 0 до 1

Мова програмування: Java

IDE: IntelliJ Idea

Код:

|  |
| --- |
| import java.util.Arrays; import java.util.Random;   public class secondTask {   public static void main(String[] args) {   int increment = 20000;   int numSteps = 20;   int start = 50000;   for (int numToCheck = start;   numToCheck < numSteps \* increment + start;   numToCheck += increment) {   System.*out*.print(numToCheck + "\t");   double[] ar = new double[numToCheck];   Random r = new Random();   for (int i = 0; i < ar.length; i++) {   ar[i] = r.nextDouble();   }   double[] copyArray = Arrays.*copyOf*(ar, ar.length);   long startTime = System.*nanoTime*();   *insertionSort*(copyArray);   double estTime = (double) (System.*nanoTime*() - startTime) /   1\_000\_000\_000;   System.*out*.printf("%.9f\t", estTime);   copyArray = Arrays.*copyOf*(ar, ar.length);   startTime = System.*nanoTime*();   *bucketSort*(copyArray);   estTime = (double) (System.*nanoTime*() - startTime) /   1\_000\_000\_000;   System.*out*.println(estTime);   }  }  private static void insertionSort(double []ar){  int N = ar.length;  for (int i = 1; i < N; i++) {  int j = i - 1;  double key = ar[i];  while (j >= 0 && ar[j] > key) {  ar[j + 1] = ar[j];  j--;  }  ar[j + 1] = key;  }  }  private static void bucketSort(double []arr){  int nBuckets = arr.length;   // Step1: create an array of empty linked lists (buckets)  DLNode[] buckets = new DLNode[nBuckets];  for (int i = 0; i < buckets.length; i++) {  buckets[i] = null;  }   // Step2: Distribute the n input numbers into the buckets  for (int i = 0; i < arr.length; i++) {  int b = (int) (nBuckets \* arr[i]);  DLNode newNode = new DLNode();  newNode.data = arr[i];  buckets[b] = DLListOperations.*addFirst*(buckets[b], newNode);  }   // Step3: Sort each list B[i] individually by insertion sort  for (int i = 0; i < buckets.length; i++) {  buckets[i] = *insertionSortList*(buckets[i]);  }   // Step4: Concatenate the lists B[0], B[1], . . B[n-1] together in order  for (int i = 0, j = 0; i < buckets.length; i++) {  while (buckets[i] != null) {  arr[j++] = buckets[i].data;  buckets[i] = DLListOperations.*removeFirst*(buckets[i]);  }  }  }  private static DLNode insertionSortList(DLNode h) {   if (h == null) {  return null;  }   // Make the first node the head of the sorted list.  DLNode node = h;  h = h.next;  node.clearLinks();  DLNode sortedList = node;   // Traverse the given linked list and insert every  // node to sorted list  while (h != null) {  DLNode insertedEl = h;  h = h.next;  insertedEl.clearLinks();  sortedList = *insertToSortedList*(sortedList, insertedEl);  }  return sortedList;  }    private static DLNode insertToSortedList(DLNode h, DLNode n) {  if (h == null) {  return n;  }  // Special case if node should be inserted at the beginning of the list  if (n.data < h.data) {  return DLListOperations.*addFirst*(h, n);  }  // Search list for correct position of n node.  DLNode search = h;  while (search.next != null && n.data > search.next.data) {  search = search.next;  }  // n node goes after search.  DLListOperations.*addAfter*(search, n);   return h;   } } |

Середовище Dell 15R-5521  
Операційна система: Windows 7 х64

Висновок: алгоритми мають час виконання приблизно О(n + k) та О(n^2) відповідно , при цьому результати залежать від ключа пошуку