МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное

образовательное учреждение

высшего образования

«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ И ФИЗИКО-МАТЕМАТИЧЕСКИХ НАУК

КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Отчёт по лабораторной работе №9.2

по дисциплине «Теория алгоритмов»

Анализ рекурсивных алгоритмов. Сортировка слиянием, сортировка Хоара, бинарный поиск

Выполнил: студент группы ФИб-4301-51-00 \_\_\_\_\_\_\_\_\_\_\_ / Н.А. Слободчиков/

Проверил: доцент каф. ПМиИ \_\_\_\_\_\_\_\_\_\_\_ /Е.В.Разова/

Киров 2021

Задание 1:

сортировка слиянием

|  |  |
| --- | --- |
| Алгоритм |  |
| static int[] MergeSort(int[] arr, int l, int r)  {  if (l < r)  {  int m = (l + r) / 2;  MergeSort(arr, l, m);  MergeSort(arr, m + 1, r);  MergeMass(arr, l, m, r);  }  return arr;  } | Начало рекурсии – получение середины за Θ(1) |
| разбиение на 2 равные части  T(n/2)+T(n/2)=2·T(n/2) |
| Выход из рекурсии: Слияние 2 массивов за Θ(n) |

T(n)= Θ(1) +T(n/2) +T(n/2) + Θ(n)=2·T(n/2) +Θ(1) +Θ(n)

T(n)=2·T(n/2) + c\*n = 2(2T(n/4) + c\*n/2)+c\* n = 4T(n/4) + 2 c\*n/2 + c\*n = 4T(n/4) + 2\*c\*n = 8(T(n/8)) + 3c\*n = 2^i\*(T(n/2^i)) + i\*c\*n

T(1) = Θ(1)

i = log2n

T(n) = n\*(T(1)) + c\*log2n\*n = Θ(n\*log2n)

сортировка Хоара

|  |  |
| --- | --- |
| Алгоритм |  |
| static int[] QuickSort(int[] arr, int minInd, int maxInd)  {  if (minInd >= maxInd)  {  return arr;  }  int Index = Partition(arr, minInd, maxInd);  QuickSort(arr, minInd, Index - 1);  QuickSort(arr, Index + 1, maxInd);  return arr;  } | Начало рекурсии – получение опорного индекса и разделение массива за Θ(n) |
| разбиение на 2 части, в лучшем случае  T(n/2)+T(n/2)=2·T(n/2)  В худшем случае  T(n-1) + T(1) |
|  |

T(1) = 1

В лучшем T(n)= Θ(n) +T(n/2) +T(n/2)

Временная оценка Ω(n\*log2n) (аналогично предыдущему случаю)

В худшем T(n)= Θ(n) +T(n-1) +T(1)

T(n)= T(n-1) + T(1) + c\*n = T(n-2) + T(1) + Θ(1) + c\*(n-1) + c\*n = T(n-2) + 2\*c\*n = T(n-3) + 3\*c\*n = T(n-i) + i\*c\*n

i = n-1

T(n) = T(1) + (n-1)\*c\*n = c\*n^2 – c\*n = O(n^2)

По времени работы сортировка слиянием эффективнее, так как в худшем случае будет быстрее чем быстрая сортировка в худшем случае и одинаково эффективна в лучшем

2)

Бинарный поиск

|  |  |
| --- | --- |
| Алгоритм |  |
| static int BinarySearch(int[] arr, int value, int first, int last)  {  if (first > last) return -1;  int m = (first + last) / 2;  int midValue = arr[m];  if (midValue == value) return m;  else  {  if (midValue > value)  {  return BinarySearch(arr, value, first, m - 1);  }  else  {  return BinarySearch(arr, value, m + 1, last);  }  }  } | Начало рекурсии – получение среднего значения массива Θ(1) |
| разбиение на 2 части и продолжение в нужной  T(n/2)  Пока не найден элемент,  Если элемент найден продолжения не происходит |
|  |

T(1) = 1

В лучшем T(n)= Θ(1) – не происходит рекурсии, выходит сразу

В худшем T(n)= T(n/2)+ Θ(1)

T(n)= T(n/2) + Θ(1) = T(n/4) + 2\* Θ(1) = T(n-2) = T(n/2^i) + i\*Θ(1)

i = log2n

T(n) = T(1) + log2n \* Θ(1) = O(log2n)

3)

Бинарный поиск

|  |  |
| --- | --- |
| Алгоритм |  |
| *Procedure Soch (i : Integer);*  *Var k : Integer;*  *Begin*  *If i>n Then Print(a)*  *Else For k:=1 To n Do*  *Begin*  *a[i]:=k;*  *Soch(i+1);*  *End;*  *End;* | Начало рекурсии – получение условие и вывод элементов массива Θ(n) |
| Вызов n рекурсий  n\*T(n-1) |
|  |

А) T(n+1)=n\*T(n-1) + c\*n

n>=1 T(n)<=f(n)

предположим, f(n) = a\*nn + bn

при n=1

T(1)<=a + b – верно при a+b>c1

Предположим, что для всех k<n неравенство T(k)≤a·kk+b выполняется – индуктивное предположение.

T(n) <= n\*(a\*(n/2)n/2+bn) + c2\*n = a \* 2\*n/2 \*(n/2)n/2+bn + c2n = 2\*a\*(n/2)n/2+1 +(c2+b)n<= a\*nn+bnпри a> c2

T(n) = O(nn)

Б) T(1) = 1

T(n+1)=n\*T(n) + c\*n= n\*(n\*T(n-1)+c\*n) +c\*n = n2T(n-2) + c\*n2 = n3T(n-3) + c\*n3 =niT(i + 1) + c\*ni

i = n

T(1) = nn + c\*nn = Θ(nn)