Министерство образования  и науки, молодежи и спорта Украины

Одесский национальный политехнический  университет

Институт компьютерных систем

Кафедра системного программного обеспечения

Лабораторная  работа №2

На тему: «Расчет метрических  характеристик качества

разработки программ  по метрикам Холстеда»

по дисциплине «Управление  качеством ПО»

Выполнил   ст. гр. АС-081                                                       Дворник А.И.

 Проверил:

Куприянов А. Б.

Одеcса 2012

**Тема:** расчет метрических характеристик качества разработки программ  по метрикам Холстеда

**Цель работы:** Получение навыков расчета показателей качества программ.

**Вариант:** Пирамидальная сортировка

Пирамидальная сортировка — алгоритм сортировки, работающий в худшем, в среднем и в лучшем случае (то есть гарантированно) за Θ(n log n) операций при сортировке n элементов. Количество применяемой служебной памяти не зависит от размера массива (то есть, O(1)).

Может рассматриватъся как усовершенствованная Bubblesort, в которой элемент всплывает (min-heap) / тонет (max-heap) по многим путям....

**Алгоритм**

Сортировка пирамидой  использует сортирующее дерево. Сортирующее  дерево — это такое двоичное дерево, у которого выполнены условия:

* Каждый лист имеет глубину либо d либо d − 1, d — максимальная глубина дерева.
* Значение в любой вершине больше, чем значения её потомков.

Удобная структура данных для сортирующего дерева — такой  массив Array, что Array[1] — элемент в корне, а потомки элемента Array[i] — Array[2i] и Array[2i+1].

Алгоритм сортировки будет  состоять из двух основных шагов:

* Выстраиваем элементы массива в виде сортирующего дерева:

Array[i]>= Array[2i]

Array[i]>= Array[2i+1]

при 1 <= i < n/2.

**Ход работы:**

**Программа на java**

package laba2;

public class heap\_Sort{

      public static void main(String a[]){

        int i;

          int arr[] = {1,3,4,5,2};

        System.out.println("\n  Heap Sort\n---------------\n");

        System.out.println("\n  Unsorted Array\n\n");

        for (i = 0; i < arr.length; i++)09

          System.out.print(" "+arr[i]);

        for(i=arr.length; i>1; i--){

          fnSortHeap(arr, i - 1);

        }

        System.out.println("\n  Sorted array\n---------------\n");

        for (i = 0; i < arr.length; i++)

          System.out.print(" "+arr[i]);

      }

      public static void fnSortHeap(int array[], int arr\_ubound){

        int i, o;

            int lChild, rChild, mChild, root, temp;

        root = (arr\_ubound-1)/2;

          for(i=root;i>=0;i--){

            lChild = (2\*i)+1;

                    rChild = (2\*i)+2;

            if((lChild <= arr\_ubound) && (rChild <= arr\_ubound)){

              if(array[rChild] >= array[lChild])

                mChild = rChild;

              else

                mChild = lChild;

            }

                    else{

              if(rChild > arr\_ubound)

                mChild = lChild;

              else

                mChild = rChild;

            }

            if(array[i] < array[mChild]){

              temp = array[i];

              array[i] = array[mChild];

                        array[mChild] = temp;

                                }

          }

        }

        temp = array[0];

        array[0] = array[arr\_ubound];

        array[arr\_ubound] = temp;

        return;

      }

}

**Измеримые характеристики программ**

**Java**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Операторы | | | Операнды | | |
| N | Оператор | Число вхождений | N | Операнд | Число вхождений |
| 1 | = | 21 | 1 | arr | 7 |
| 2 |  |  | 2 | 1 | 5 |
| 3 | + | 4 | 3 | 2 | 5 |
| 4 | System.out.println | 3 | 4 | 3 | 1 |
| 5 | for | 4 | 5 | 4 | 1 |
| 6 | < | 3 | 6 | 5 | 1 |
| 7 | ++ | 2 | 7 | 0 | 5 |
| 8 | <= | 2 | 8 |  |  |
| 9 | >= | 1 | 9 | i | 14 |
| 10 | [ ] | 9 | 10 | lChild | 6 |
| 11 | -- | 2 | 11 | rChild | 7 |
| 12 | - | 2 | 12 | mChild | 8 |
| 13 | > | 2 | 13 | root | 3 |
| 14 | if | 4 | 14 | temp | 5 |
| 15 | Else | 3 | 15 | array | 13 |
| 16 | / | 1 | 16 | arr\_ubound | 7 |
|  |  |  |  |  |  |

Расчет метрик

Число уникальных операторов η1 = 15

Общее число всех операторов N1 = 63

Число уникальных операндов  η2 = 15

Общее число всех операндов  N2 = 63

Словарь программы η = 30

Длина программы N =126

Теоретическая оценка длины  N = 15log215 + 15log215=  117

Расчетные характеристики программы

Реальный объем программы

hh

V=12630=618

Оценка ее реализации

L\*=(2h)/(hN) = 30/945=0.03

Трудность понимания

E = V\L\* = 618/0.03 = 20600

Трудоемкость  кодирования:

= 1/0.03=33

Уровень языка  выражения

= 618/0.0009=186.6

Информационное  содержание

l=V/D = 618/33=18.8

Оптимальная модульность

M=15/6=2.5

**Программа на C++**

using namespace std;

void iswap(int &n1, int &n2)

{

    int temp = n1;

    n1 = n2;

    n2 = temp;

}

int main()

{

    int const n = 100;

    int a[n];

    for ( int i = 0; i < n; ++i ) { a[i] = n - i; cout << a[i] << " "; }

    int sh = 0;

    bool b = false;

    for(;;)

    {

b = false;

for ( int i = 0; i < n; i++ )

{

    if( i \* 2 + 2 + < a[i \* 2 + 2 + sh] ) )

{

    if ( a[i \* 2 + 1 + sh] \*/ a[i \* 2 + 2 + sh] )

    {

iswap( a[i + sh], a[i \* 2 + 1 + sh] );

b = true;

    }

    else if ( a[i \* 2 + 2 + sh] \*/ a[ i \* 2 + 1 + sh])

         {

             iswap( a[ i + sh], a[i \* 2 + 2 + sh]);

             b = true;

         }

}

    }

    else if( i \* 2 + 1 + sh  < a[ i \* 2 + 1 + sh] )

             {

                 iswap( a[i + sh], a[i \* 2 + 1 + sh] );

                 b = true;

             }

         }

}

if (!b) sh++;

if ( sh + 2 == n ) break;

    }  //конец сортировки

    cout << endl << endl;

    for ( int i = 0; i < n; ++i ) cout << a[i] << " ";

    \_getch();

    return 0;

}

С

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Операторы | | | Операнды | | |
| N | Оператор | Число вхождений | N | Операнд | Число вхождений |
| 1 | const | 1 | 1 | n1 | 3 |
| 2 | = | 14 | 2 | n2 | 3 |
| 3 | printf | 4 | 3 | temp | 2 |
| 4 | for | 4 | 4 | n | 7 |
| 5 | < | 7 | 5 | a | 16 |
| 6 | ++ | 4 | 6 | i | 26 |
| 7 | if | 6 | 7 | sh | 16 |
| 8 | + | 25 | 8 | 0 | 5 |
| 9 | getch | 1 | 9 | b | 6 |
| 10 | [ ] | 9 | 10 | 2 | 17 |
| 11 | - | 1 |  |  |  |
| 12 | false | 2 |  |  |  |
| 13 | \* | 11 |  |  |  |
| 14 | else | 1 |  |  |  |
| 15 | true | 3 |  |  |  |

Число уникальных операторов η1 = 15

Общее число всех операторов N1 = 93

Число уникальных операндов  η2 = 10

Общее число всех операндов  N2 = 101

Словарь программы η = 25

Длина программы N = 194

Теоретическая оценка длины  N = 15log215 + 10log210 =

Расчетные характеристики программы

Реальный объем программы

hh

V=19425=901

Оценка ее реализации

L\*=(2h)/(hN) = 25/1940=0.012

Трудность понимания

E = V\L\* = 901/0.012 = 75083

Трудоемкость  кодирования:

= 1/0.012=83

Уровень языка  выражения

= 901/0.00014=625944

Информационное  содержание

l=V/D = 901/83=10.9

Оптимальная модульность

M=10/6=1.5

**Выводы:** В ходе выполнения работы были разработаны, модифицированы программы реализации алгоритма  на языках программирования Java, Си.  Для созданных программы были оценены метрические характеристики  по Холстеду.