**算法分析与设计实验报告**

**第 3 次实验**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 杨杰 | 学号 | 201908010705 | | 班级 | 计科1907 |
| 时间 | 5.22 | 地点 | 软件大楼 | | | |
| 实验名称 | 单调递增最长子序列问题 | | | | | |
| 实验目的 | 设计指定时间复杂度的算法，通过上机实验，要求掌握单调递增最长子序列问题的问题描述、算法设计思想、程序设计。 | | | | | |
| 实验原理 | 以一个数组记录临时的最长序列，然后记录这个序列中的位置的变化。 | | | | | |
| 实验步骤 | 1.初始化B数组，令B[0]=array[0]  2.遍历array数组，检查第i个元素array[i]是否大于B[len-1](len为当前最长递增子序列的长度)，若大于，那么将array[i]追加到B数组末尾，len的值加1，否则将替换B数组某位置的元素。  3.回溯找到LIS的结果序列 | | | | | |
| 关键代码 | int BinSearch(int len, int w)*//修改的二分查找算法，返回数组元素需要插入的位置*  {      int left = 0, right = len - 1;      int mid;      while (left <= right)      {          mid = left + (right - left) / 2;          if (B[mid] > w)              right = mid - 1;          else if (B[mid] < w)              left = mid + 1;          else*//找到了该元素，则直接返回*              return mid;      }      return left;*//数组B中不存在该元素，则返回该元素应该插入的位置*  }  int LIS()*//计算最长递增子序列的长度,计算B数组的元素,array[]循环完一遍后,B的长度len即为所求*  {      int len = 1;*//用于标示B数组中的元素个数*      B[0] = array[0];      int i, pos = 0;      for (i = 1; i < n; ++i)      {          pos = BinSearch(len, array[i]);*//二分查找需要插入的位置*          B[pos] = array[i];          if (pos == len)              len++;      }      return len;  }      B[len] = MAX;      for (int i = 0; i < n; i++)*//回溯*      {          if (array[i] >= B[j] && array[i] <= B[j + 1])          {              outfile << array[i] << " ";              j++;              if (j == len)                  break;          }      } | | | | | |
| 算法复杂度分析 | 二分查找的时间复杂度为O(logn)，在LIS函数中，循环共执行n-1次，所以时间复杂度为O(nlogn).  因为开了多个一维数组，所以空间复杂度为O(n). | | | | | |
| 测试结果  （含运行时间） | **小规模数据**      **中规模数据**      **大规模数据** | | | | | |
| 实验心得 | 该算法在得到最长递增子序列长度的基础上又可以输出子序列的元素，使用了回溯的方法，将符合条件的元素放入数组。  经过这次实验，我对于单调递增最长子序列问题的相关代码已基本熟悉，算法知识得到了复习与巩固。在写代码与调试的过程中，在解决问题过程中，丰富了个人编程的经历和经验，提高了个人解决问题的能力。 | | | | | |
| 实验得分 |  | 助教签名 | |  | | |

**附录：完整代码**

*//LIS*

#include <iostream>

#include <fstream>

#include <windows.h>

#define MAX 9999

using namespace std;

int array[MAX];

int A[MAX];

int B[MAX];

int n;

ifstream infile("LIS\_in.txt", ios::in);

ofstream outfile("LIS\_out.txt", ios::out);

int BinSearch(int len, int w)*//修改的二分查找算法，返回数组元素需要插入的位置*

{

    int left = 0, right = len - 1;

    int mid;

    while (left <= right)

    {

        mid = left + (right - left) / 2;

        if (B[mid] > w)

            right = mid - 1;

        else if (B[mid] < w)

            left = mid + 1;

        else*//找到了该元素，则直接返回*

            return mid;

    }

    return left;*//数组B中不存在该元素，则返回该元素应该插入的位置*

}

int LIS()*//计算最长递增子序列的长度,计算B数组的元素,array[]循环完一遍后,B的长度len即为所求*

{

    int len = 1;*//用于标示B数组中的元素个数*

    B[0] = array[0];

    int i, pos = 0;

    for (i = 1; i < n; ++i)

    {

        pos = BinSearch(len, array[i]);*//二分查找需要插入的位置*

        B[pos] = array[i];

        if (pos == len)

            len++;

    }

    return len;

}

int main()

{

    infile >> n;

    for (int i = 0; i < n; i++)

    {

        infile >> array[i];

    }

    infile.close();

    double time = 0;

    LARGE\_INTEGER nFreq, nBeginTime, nEndTime;

    QueryPerformanceFrequency(&nFreq);

    QueryPerformanceCounter(&nBeginTime);*//开始计时*

    int len = LIS();

    QueryPerformanceCounter(&nEndTime);*//停止计时*

    time = (double)(nEndTime.QuadPart - nBeginTime.QuadPart) / (double)nFreq.QuadPart;*//计算程序执行时间单位为s*

    outfile << "LIS: " << len << "\n";

    int j = 0;

    B[len] = MAX;

    for (int i = 0; i < n; i++)

    {

        if (array[i] >= B[j] && array[i] <= B[j + 1])

        {

            outfile << array[i] << " ";

            j++;

            if (j == len)

                break;

        }

    }

    outfile << "\n程序耗时" << time \* 1000 << "ms" << endl;

    outfile.close();

    return 0;

}

*//LIS随机数据生成器*

#include <iostream>

#include <fstream>

#include <time.h>

#define random(a, b) (rand() % (b - a + 1) + a)

using namespace std;

int main()

{

    int t = 0;

    cin >> t;

    ofstream outfile("LIS\_in.txt", ios::out);

    outfile << t << endl;

    srand((int)time(NULL));

    for (int i = 1; i <= t; i++)

    {

        outfile << random(1, 100) << " ";

        if (i % 20 == 0)

            outfile << endl;

    }

    outfile.close();

    return 0;

}