**算法分析与设计实验报告**

**第 三 次实验**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 李平凡 | 学号 | 201907040102 | | 班级 | 计科1905 |
| 时间 | 5.20 | 地点 | 软件大楼 | | | |
| 实验名称 | 单调递增子序列 | | | | | |
| 实验目的 | 通过搜索资料，要求掌握一些基本问题和典型例题的处理方式 | | | | | |
| 实验原理 | 原来采用O(n^2)的求解方法来解决单调递增子序列，有两种方式来做，一种是上课讲的，采用动态递归利用打表的方法来将一个序列和它按递增顺序排列的序列进行比较，找这两个序列的最长公共子序列，另一种理解是每次找到一个元素的时候记录这个元素的最佳插入位置，并用一个数组记录这个值（这个序列的最末元素），最后只需要直接看看数组长度就可以知道答案，时间开销达到n^2是因为它需要遍历一遍前面所有元素。而现在采用nlogn开销来解决这个问题，其实也采用了动态规划的思想，因为里面存在着很多的重复计算，所以每次我并不需要更新那个记录数组的时候每个元素都遍历一遍，我只需要找到合适的位置把这个元素更改即可，所以就采用了二分搜索的方法，并利用反正法，证明出这个记录的数组一定是单增的。 | | | | | |
| 实验步骤 | ①首先输入一个随机序列，并创建一个数组用来记录每次更新一个元素在这个数组中的最佳插入位置（下标代表了最佳插入位置）以及这个单调递增子序列的最末元素值。  ②采用一层循环来遍历每一个元素，同时用二分搜索，如果这个元素比记录中的元素小，就用二分搜索返回这个元素最佳序列插入位置，否则就将数组元素加一，同时新增元素的值是该元素（也就是该元素结尾）  ③ 将最后的记录数组长度输出即可 | | | | | |
| 关键代码 | int getresult(int a[],int n)  {  int tempindex;  int count=0;  vector<int> b(n+1,0);  for(int i=0;i<n;i++)  {  if(a[i]<=b[i+1])  {  tempindex=Bsearch(i+1,b,a[i]);  b[tempindex]=a[i];  }  else  {  b[i+2]=a[i];  count++;  }  }  cout<<count<<endl;  file1<<count;  }  这个求解函数中采用一个for循环用来遍历所有的元素，如果当前元素和记录数组中的元素相比较小，那么就需要重新定位这个元素，将其记录在记录数组中的合适位置，如果大于那么就将记录数组加一个元素，并将这个元素的值改成该元素。  int Bsearch(int len,vector<int> b,int n)  {  int left = 1;  int right = len;  while (left < right)  {  int mid = (left + right) / 2;  if (b[mid] > n)  {  right = mid;  }  else  {  left = mid+1;  }  }  return right;  }  采用二分搜索来返回一个合适的下标值给这个元素，时间复杂度为nlogn | | | | | |
| 测试结果 |  | | | | | |
| 实验心得 | 对于这个算法采用的思想就是一个打表记录，它将n^2的时间开销中的重复计算减少到最小，从而达到n的水平，但是二分搜索的时间开销是logn，在一层循环中最坏的就是每次计算一次返回下标，所以时间复杂度也就达到了nlogn，所以最终时间开销是O(nlogn) | | | | | |
| 实验得分 |  | 助教签名 | |  | | |