《算法分析与设计》

实 验 报 告

学 号

姓 名

年 级 2020

专 业 软件工程

院 系 计算机与人工智能学院

二O二二年三月

# 实验2.4分析算法时间复杂度

1. 实验目的

|  |
| --- |
| （1）正确分析算法时间复杂度 |

1. 实验任务

|  |
| --- |
| 1. 分析给定算法的时间复杂度。 2. 上机实验，绘制不同输入情况下的运行时间曲线，并与(1)中的分析结果进行比较。 3. 撰写相应的实验报告，实验报告内容包括：实验目的、实验任务、 实验环境、实验步骤、实验结果及其分析以及实验总结等部分内容。 |

1. 实验环境

|  |
| --- |
| * 1. 硬件环境  1. 计算机：拯救者R7000P 2020H 2. CPU: AMD Ryzen 7 4800H with Radeon Graphics 2.90 GHz 3. RAM：16GB    1. 软件环境 4. 操作系统：Windows11家庭中文版 5. 开发工具：Visual Studio Code |

1. 实验步骤及结果

|  |
| --- |
| * 1. 实验预习      1. 时间复杂度   算法外层循环为while（m），分析可知内层算法将会运行m次；  内层第一个for循环，从1到n1，运行n1次；  内嵌for循环，从1到n2，运行n2次，因是嵌套结构，所以内层整体将运行n1\*n2次；  所以整体分析可知，该程序时间复杂度为O(m\*n1\*n2)。   * + 1. 程序代码   #include <stdio.h>  #include <string.h>  #include <stdlib.h>  #include <math.h>  #define Max 110  int n1, n2, m, t;  int match[Max][Max], f[Max][Max], g[Max][Max];  int s1[Max], s2[Max];  int max(int *a*,int *b*)  {      return *a* > *b* ? *a* : *b*;  }  int main()  {      int i, j;      int ii, jj;      scanf("%d", &m);      while (m)      {          m--;          scanf("%d%d", &n1, &n2);          for (i = 1; i <= n1; i++)              scanf("%d", &s1[i]);          for (i = 1; i <= n2; i++)              scanf("%d", &s2[i]);          memset(f, 0, sizeof(f));          memset(g, 0, sizeof(g));          for (i = 1; i <= n1; i++)              for (j = 1; j <= n2; j++)              {                  if (s1[i]==s2[j-1])                      f[i][j] = j - 1;                  else                      f[i][j] = f[i][j - 1];              }          for (i = 1; i <= n2; i++)              for (j = 1; j <= n1; j++)              {                  if (s2[i] == s1[j - 1])                      g[i][j] = j - 1;                  else                      g[i][j] = g[i][j - 1];              }          memset(match, 0, sizeof(match));          for (i = 1; i <= n1; i++)              for (j = 1; j <= n2; j++)              {                  match[i][j] = max(match[i - 1][j], match[i][j - 1]);                  if (s1[i]==s2[j])                      continue;                  ii = f[i][j];                  jj = g[j][i];                  if (ii==0||jj==0)                      continue;                  match[i][j] = max(match[i][j], match[jj - 1][ii - 1] + 2);              }          printf("%d\n", match[n1][n2]);      }      system("pause");      return 0;  }  4.2上机实验  4.2.1算法测试   1. 测试数据   M=1 n1=5000 n2递增  M=1 n2=5000 n1递增   1. 测试数据   M=2 n1=5000 n2递增  M=2 n2=5000 n1递增  4.2.2测试结果及其分析  (1)对应测试数据(1)，其程序执行完成后的散点图如图1和图2所示。    图1 散点图1    图2 散点图2  结果分析：  由散点图拟合曲线看，时间与n1，n2均为线性关系。所以时间复杂度表达式应含有n1\*n2  (2)对应测试数据(2)，其程序执行完成后的散点图如图3和图4所示。    图3 散点图3    图4 散点图4  结果分析：  由散点图拟合曲线看，时间与n1，n2均为线性关系。所以时间复杂度表达式应含有n1\*n2  与m=1时相比较，各对应情况所运行出的时间为2倍关系，即时间复杂度也与m成正相关。  结合两组测试数据可得，理论分析得出的时间复杂度O(m\*n1\*n2)正确。 |
| 5.实验总结  通过本次实验，我可以比较不同数据输入规模情况下算法的执行时间，不同算法对问题求解效率的影响以及影响算法运行时间的一些因素，学会了如何分析算法的时间效率等。 |