**算法分析与设计实验报告**

**第 2 次实验**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 杨杰 | 学号 | 201908010705 | | 班级 | 计科1907 |
| 时间 | 4.3 | 地点 | 软件大楼 | | | |
| 实验名称 | 半数集问题 | | | | | |
| 实验目的 | 通过上机实验，要求掌握半数集问题的问题描述、算法设计思想、程序设计。 | | | | | |
| 实验原理 | 运用动态规划，对于给定的自然数n，计算半数集set(n)中的元素个数。根据不同数据规模的测试用例，能准确输出半数集set(n)中的元素个数，并计算出程序运行所需要的时间。 | | | | | |
| 实验步骤 | * + - 1. 进行数学推导，构造递推公式，设计需要哪些函数来实现相应功能。  1. 构造set函数，用来求set(n)的值,这里用到动态规划，将重叠子问题的结果保存下来，减少计算量。 2. 进行测试验证，发现当n过大时结果会溢出，考虑大整数加法，用数组存储一个长整数，构造add函数。 3. 依据add函数，构造big\_num\_set函数，计算当n很大时半数集中的元素个数。 | | | | | |
| 关键代码 | ll set(int n)*//当set(n)较小时，用此函数计算set(n)的值*  {      if (a[n] != 0)          return a[n];      else if (n == 1)          a[n] = 1;      else      {          for (int i = 1; i <= n / 2; i++)              a[n] += set(i);          a[n]++;      }      return a[n];  }  string sstr(int n)*//整型数转字符串*  {      string s = "";      stringstream ss;      ss << n;      s = ss.str();      return s;  }  string add(string a, string b)*//将两个字符串对应的十进制整数相加，并返回一个字符串*  {      int l = a.length();      int t = b.length();      int x = l > t ? (l + 1) : (t + 1);      int aa[x - 1] = {0}, bb[x - 1] = {0};      int c[x] = {0};      int i, m;      for (i = 0, m = l - 1; m >= 0; i++, m--)          aa[i] = a[m] - '0';      for (i = 0, m = t - 1; m >= 0; i++, m--)          bb[i] = b[m] - '0';      for (i = 0; i < x - 1; i++)          c[i] = aa[i] + bb[i];      for (i = 0; i < x; i++)*//这里是把超过10的进位*      {          if (c[i] >= 10)          {              c[i + 1] = c[i + 1] + c[i] / 10;              c[i] %= 10;          }      }      string s = "";      int k = 0;      for (i = x - 1; i >= 0; i--)*//把前导0去掉*      {          if (c[i] == 0)              continue;          else          {              k = i;              break;          }      }      if (i == -1)          s = "0";      for (i = k; i >= 0; i--)          s = s + sstr(c[i]);      return s;  }  string big\_num\_set(int n)*//当set(n)超过unsigned long long范围时，用此函数计算set(n)的值*  {      if (A[n] != "")          return A[n];      else if (n == 1)          A[n] = "1";      else      {          for (int i = 1; i <= n / 2; i++)              A[n] = add(big\_num\_set(i), A[n]);          A[n] = add(A[n], "1");      }      return A[n];  } | | | | | |
| 算法复杂度分析 | 先逐个函数进行分析，set函数因为调用了重叠子问题的解，所以时间复杂度与打表时间复杂度相同，为 O(n2);sstr函数时间复杂度为O(1);add函数一层循环，时间复杂度为O(n); big\_num\_set函数二层循环，时间复杂度为O(n2)。 | | | | | |
| 测试结果  （含运行时间） |  | | | | | |
| 实验心得 | 本次实验要用到动态规划算法，因为有很多重叠子问题，将重叠子问题的答案存起来，用到的时候直接查找调用相比重复计算可以节省许多时间。  还有一个问题是本实验的数据量过大，存在溢出风险，所以本实验要用到大整数加法，输入两个字符串，将字符串对应的整型数按位相加并考虑进位，将结果按位存在数组中，遍历数组即可得到长整数的值。    其实当n超过15725时unsigned long long就会溢出。  另外，由于大整数加法相比基本数据类型加法耗时长得多，所以可以对输入的数据规模进行判断，当数据规模较小时使用基本数据类型的set函数，当数据规模过大时使用大整数加法的big\_num\_set函数。    上图为大整数加法，运行时间确实较长，但不存在溢出问题。  另外，针对数据溢出问题，考虑到python语言不会溢出，所以用python写了一份半数集代码，详见附录。  经过这次实验，我对于动态规划的算法思想已基本熟悉，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**

*//半数集问题*

#include <iostream>

#include <fstream>

#include <sstream>

#include <windows.h>

#include <string.h>

#define ll unsigned long long

using namespace std;

string A[25000];

ll a[25000];

ll set(int n)*//当set(n)较小时，用此函数计算set(n)的值*

{

    if (a[n] != 0)

        return a[n];

    else if (n == 1)

        a[n] = 1;

    else

    {

        for (int i = 1; i <= n / 2; i++)

            a[n] += set(i);

        a[n]++;

    }

    return a[n];

}

string sstr(int n)*//整型数转字符串*

{

    string s = "";

    stringstream ss;

    ss << n;

    s = ss.str();

    return s;

}

string add(string a, string b)*//将两个字符串对应的十进制整数相加，并返回一个字符串*

{

    int l = a.length();

    int t = b.length();

    int x = l > t ? (l + 1) : (t + 1);

    int aa[x - 1] = {0}, bb[x - 1] = {0};

    int c[x] = {0};

    int i, m;

    for (i = 0, m = l - 1; m >= 0; i++, m--)

        aa[i] = a[m] - '0';

    for (i = 0, m = t - 1; m >= 0; i++, m--)

        bb[i] = b[m] - '0';

    for (i = 0; i < x - 1; i++)

        c[i] = aa[i] + bb[i];

    for (i = 0; i < x; i++)*//这里是把超过10的进位*

    {

        if (c[i] >= 10)

        {

            c[i + 1] = c[i + 1] + c[i] / 10;

            c[i] %= 10;

        }

    }

    string s = "";

    int k = 0;

    for (i = x - 1; i >= 0; i--)*//把前导0去掉*

    {

        if (c[i] == 0)

            continue;

        else

        {

            k = i;

            break;

        }

    }

    if (i == -1)

        s = "0";

    for (i = k; i >= 0; i--)

        s = s + sstr(c[i]);

    return s;

}

string big\_num\_set(int n)*//当set(n)超过unsigned long long范围时，用此函数计算set(n)的值*

{

    if (A[n] != "")

        return A[n];

    else if (n == 1)

        A[n] = "1";

    else

    {

        for (int i = 1; i <= n / 2; i++)

            A[n] = add(big\_num\_set(i), A[n]);

        A[n] = add(A[n], "1");

    }

    return A[n];

}

int main()

{

    memset(a, 0, sizeof(a));

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

    ofstream outfile("banshuji\_out1.txt", ios::out);

    double time = 0;

    LARGE\_INTEGER nFreq, nBeginTime, nEndTime;

    int x;

    ll r = 0;

    string big\_num\_r = "0";

    bool flag = true;

    while (infile >> x)

    {

        QueryPerformanceFrequency(&nFreq);

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

        if (x <= 15725)

        {

            r = set(x);

            flag = true;

        }

        else

        {

            big\_num\_r = big\_num\_set(x);

            flag = false;

        }

        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*

        if (flag)

            outfile << "半数集set(" << x << ")中的元素个数为" << r << ",程序耗时" << time \* 1000 << "ms" << endl;

        else

            outfile << "半数集set(" << x << ")中的元素个数为" << big\_num\_r << ",程序耗时" << time \* 1000 << "ms" << endl;

    }

    infile.close();

    outfile.close();

    return 0;

}

*//半数集数据生成器*

#include <iostream>

#include <fstream>

#include <time.h>

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

using namespace std;

int main()

{

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

    srand((int)time(NULL));

    int t=0;

    cin>>t;

    while(t--)

        outfile<<random(1,2000)<<endl;

    return 0;

}

#半数集.py

import time as t  
  
a = [0 for i in range(25000)]  
  
  
def set(n): *# 计算set(n)的值* if a[n] != 0:  
 return a[n]  
 elif n == 1:  
 a[n] = 1  
 else:  
 for i in range(1, n // 2 + 1):  
 a[n] += set(i)  
 a[n] += 1  
 return a[n]  
  
  
def main():  
 with open(**"banshuji\_in.txt"**, **'r'**, encoding=**"GBK"**) as f:  
 ls = f.readlines()  
 with open(**"banshuji\_out.txt"**, **'w'**, encoding=**"GBK"**) as f:  
 for x in ls:  
 start = t.perf\_counter()  
 r = set(eval(x))  
 dur = t.perf\_counter() - start  
 f.write(**"半数集set({})中的元素个数为{},程序耗时{}ms**\n**"**.format(eval(x), r, dur \* 1000))  
  
  
main()