**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 | 集合划分问题 | | | | | |
| 实验目的 | 通过上机实验，要求掌握集合划分问题的问题描述、算法设计思想、程序设计。 | | | | | |
| 实验原理 | 运用动态规划，对于给定的正整数n，计算出n个元素的集合可以划分为多少个不同的非空子集。根据不同数据规模的测试用例，能准确计算可以划分的不同的非空子集个数，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fun函数，用来求fun(n,m)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，考虑大整数加法和大整数乘法，用数组存储一个长整数，构造add函数和mul函数。 3. 依据add函数和mul函数，构造big\_num\_fun函数，计算当n很大时可以划分的不同的非空子集个数。 | | | | | |
| 关键代码 | ll fun(int n, int m)*//当fun(n,m)较小时，用此函数计算fun(n,m)的值*  {      if (a[n][m] != -1)          return a[n][m];      else if (n < m)          a[n][m] = 0;      else if (m == 0)          a[n][m] = 0;      else if (m == n)          a[n][m] = 1;      else          a[n][m] = fun(n - 1, m - 1) + m \* fun(n - 1, m);      return a[n][m];  }  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 mul(string a, string b)*//将两个字符串对应的十进制整数相乘，并返回一个字符串*  {      int l = a.length();      int t = b.length();      int aa[l] = {0}, bb[t] = {0};      int c[l + t] = {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 < l; i++)          for (m = 0; m < t; m++)              c[i + m] += aa[i] \* bb[m];      for (i = 0; i < l + t; 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 = l + t - 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\_fun(int n, int m)*//当fun(n,m)超过long long范围时，用此函数计算fun(n,m)的值*  {      if (A[n][m] != "")          return A[n][m];      else if (n < m)          A[n][m] = "0";      else if (m == 0)          A[n][m] = "0";      else if (m == n)          A[n][m] = "1";      else          A[n][m] = yunsuan(big\_num\_fun(n - 1, m - 1), sstr(m), big\_num\_fun(n - 1, m));      return A[n][m];  } | | | | | |
| 算法复杂度分析 | 如果**fun函数不调用重叠子问题的解而单纯递归的话**，设fun函数时间代价函数为T(n,m),T(n,m)存在一个下界Ω(min{2^(n-m),2^(m-1)})，存在一个上界O(2^(n-1)).即分治递归法解集合划分问题的时间复杂度为指数级。  分治递归法的时间复杂度证明：  由计算式：s[n][m]=s[n-1][m-1]+m\*s[n-1][m]，得到：T(n,m)=T(n-1，m-1)+T(n-1,m)  **下界**  ​其解可视作一棵二叉树，每个状态都有两种不同的子状态，考虑其下界，即一直走左子节点或右子节点，即有：  A.一直走左子节点  T(n,m)≈T(n−1,m−1)+T(n−1,m−1)=2T(n−1,m−1)  ​由题有n>=m，利用乘法消项： T(n,m)=2T(n-1,m-1)  2T(n-1,m-1)=2^2T(n-2,m-2) ​由之前列出的退化结果，其出口为s[i][1]=1于是可以得到： T(n,m)=2^(m-1)T(n-m+1,1)= 2^(m-1)  B.一直走右子节点  T(n,m)≈T(n−1,m)+T(n−1,m)=2T(n−1,m)  ​同理有n>m，利用乘法消项： T(n,m)=2T(n-1,m)  2T(n-1,m)=2^2T(n-2,m) ​由之前的退化结果，可以得此时的出口为s[i][i]=1于是我们可以得到： T(n,m)=2^(n-m)T(m,m)=2^(n-m) ​因此我们得到此方法的下界为： T(n,m)=Ω(min{2^(n−m),2^(m−1)}).  **上界**  ​考虑解在二叉树左右子节点反复横跳的情况，不难得出极端情况为：一次左，一次右，因此我们可以将之前计算的单走某一支的情况相乘即表示所有可能的情况。  ​不难得出，其上界为： T(n,m)= 2^(n−m)\*2^(m−1)=O(2^(n-1))    反之，如果**fun函数调用重叠子问题的解**，那么时间复杂度将等于打表的时间复杂度O(mn).  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); mul函数二层循环，时间复杂度为O(n2);big\_num\_fun函数时间复杂度为O(mn)。 | | | | | |
| 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和大整数乘法，输入两个字符串，将字符串对应的整型数按位相加或相乘并考虑进位，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25时long long就会溢出。  另外，由于大整数加法和乘法相比基本数据类型加法、乘法耗时长得多，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fun函数，当数据规模过大时使用大整数加法和乘法的big\_num\_fun函数。    上图为通过大整数加法和乘法实现的集合划分，运行时间确实较长，但不存在溢出问题。  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**

*//集合划分问题*

#include <iostream>

#include <string.h>

#include <sstream>

#include <fstream>

#include <windows.h>

#define yunsuan(a, b, c) add(a, mul(b, c))

#define ll long long

using namespace std;

string A[1000][1000];

ll a[1000][1000];

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

{

    if (a[n][m] != -1)

        return a[n][m];

    else if (n < m)

        a[n][m] = 0;

    else if (m == 0)

        a[n][m] = 0;

    else if (m == n)

        a[n][m] = 1;

    else

        a[n][m] = fun(n - 1, m - 1) + m \* fun(n - 1, m);

    return a[n][m];

}

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

{

    int l = a.length();

    int t = b.length();

    int aa[l] = {0}, bb[t] = {0};

    int c[l + t] = {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 < l; i++)

        for (m = 0; m < t; m++)

            c[i + m] += aa[i] \* bb[m];

    for (i = 0; i < l + t; 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 = l + t - 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\_fun(int n, int m)*//当fun(n,m)超过long long范围时，用此函数计算fun(n,m)的值*

{

    if (A[n][m] != "")

        return A[n][m];

    else if (n < m)

        A[n][m] = "0";

    else if (m == 0)

        A[n][m] = "0";

    else if (m == n)

        A[n][m] = "1";

    else

        A[n][m] = yunsuan(big\_num\_fun(n - 1, m - 1), sstr(m), big\_num\_fun(n - 1, m));

    return A[n][m];

}

int main()

{

    memset(a, -1, sizeof(a));

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

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

    int x = 0;

    ll sum = 0;

    string big\_num\_sum = "0";

    bool flag = true;

    double time = 0;

    LARGE\_INTEGER nFreq, nBeginTime, nEndTime;

    while (infile >> x)

    {

        QueryPerformanceFrequency(&nFreq);

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

        if (x <= 25)

        {

            for (int i = 1; i <= x; i++)

                sum += fun(x, i);

            flag = true;

        }

        else

        {

            for (int i = 1; i <= x; i++)

                big\_num\_sum = add(big\_num\_sum, big\_num\_fun(x, i));

            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 << x << "个元素的集合可以划分为" << sum << "个不同的非空子集，程序耗时" << time \* 1000 << "ms" << endl;

        else

            outfile << x << "个元素的集合可以划分为" << big\_num\_sum << "个不同的非空子集，程序耗时" << time \* 1000 << "ms" << endl;

        sum = 0;

        big\_num\_sum = "0";

    }

    infile.close();

    outfile.close();

    return 0;

}

*//集合划分数据生成器*

#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;

    srand((int)time(NULL));*//time seed*

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

    while(t--)

        outfile<<random(1,1000)<<endl;

    outfile.close();

    return 0;

}