**《算法设计与分析》实验报告**

实验名称 ： 实验2 递归与分治 2

实验日期 ： 　 2025.3.11

姓 名 ： 　高心阳

学 号 ： 　 084623237

班 级 ： 　计算机232

成 绩 ：

**人工智能与信息技术学院**

**南京中医药大学**

|  |
| --- |
| **实验目的：** |
| 1. 熟练掌握递归与分治的使用 2. 将递归分治用到实际比赛题中 |
| **实验内容和要求** |
| 1. 蓝桥杯题目 1004: [递归]母牛的故事  题目描述  有一头母牛，它每年年初生一头小母牛。每头小母牛从第四个年头开始，每年年初也生一头小母牛。请编程实现在第n年的时候，共有多少头母牛？  输入  输入数据由多个测试实例组成，每个测试实例占一行，包括一个整数n(0<n<55)，n的含义如题目中描述。  n=0表示输入数据的结束，不做处理。  输出  对于每个测试实例，输出在第n年的时候母牛的数量。  每个输出占一行。  样例输入  2  4  5  0  样例输出  2  4  6  2、**菜鸟理财**  **问题描述：**小明同学过年回家收到不少压岁钱，听说炒股比较赚钱，决定自己学习股票交易。股票交易规则很简单，低买高卖，但是小明同学还是想先练练手，于是他找了某只股票近一段时间的股价信息，研究如何买卖才能收益最大化。  **问题抽象：**给定n个交易日的股票价格，选择一个买入和卖出日期，使收益最大。  **限制条件：**先买入再卖出，只能买卖一次。    **变量说明：**  整形数组A[n]：股票价格  结构体R(d1,d2,sum)：包含买入日期d1、卖出日期d2、收益sum  \*3. 题目描述  青蛙跳台阶问题  一只青蛙一次可以跳上1级台阶，也可以跳上2级。  （1）求该青蛙跳上一个n级的台阶总共有多少种跳法。  （2）如果一只青蛙一次可以跳上1级台阶，也可以跳上2级……它也可以跳上 n 级。  请问这时该青蛙跳上一个 n 级的台阶总共有多少种跳法。 |
| **运行结果（写清题号）** |
| |  | | --- | | Task1代码 | | #include <iostream>    using namespace std;    typedef long long ll;    void solve() {   ll v[55 + 5];   v[1] = 1;   v[2] = 2;   v[3] = 3;   for (int i = 4; i < 55 + 1; ++i)   v[i] = v[i - 3] + v[i - 1];   int n;   while (cin >> n) {   if (n == 0)break;   cout << v[n] << endl;   }  // for (int i = 1; i < 55 + 1; ++i)  // cout << i << " " << v[i] << endl;  }    int main() {   int T;  // cin>>T;   T = 1;   while (T--) {   solve();   }   return 0;  } |  |  | | --- | | Task2代码 | | #include <bits/stdc++.h>    using namespace std;    typedef long long ll;    std::vector<int> v;    class R {  private:   int d1;//买入日期   int d2;//卖出日期  public:   int sum;//收益   R(int date1, int date2, int sum) {d1 = date1; d2 = date2; this->sum = sum;}   R(const R &other) {d1 = other.d1; d2 = other.d2; sum = other.sum;}   ~R() {}  };    R find(int a, int b) {   if (a == b)   return R(a, b, 0);     int mid = (a + b) / 2;   R l = find(a, mid);   R r = find(mid+1, b);     int min = v[a], max = v[mid];   int min\_d = a, max\_d = mid;     for (int i = a + 1; i <= mid; i++) {   if (min > v[i])   {   min = v[i];   min\_d = i;   }   }   for (int i = mid + 1; i <= b; i++) {   if (max < v[i])   {   max = v[i];   max\_d = i;   }   }     R m(min\_d, max\_d, max - min);     if (l.sum > r.sum && l.sum > m.sum)   return l;   if (r.sum > l.sum && r.sum > m.sum)   return r;   return m;  }    void solve() {   int n;   cin >> n;   v.resize(n + 3);   for (int i = 0; i <= n; ++i)   cin >> v[i];   R ans = find(0, n);   cout << ans.sum << endl;     // 线性扫描   // int n;   // cin >> n;   // vector<int> prices(n+3);   // for (int i = 0; i <= n; ++i)   // cin >> prices[i];   // int minPrice = INT\_MAX;   // int maxProfit = 0;   // for (int i = 0; i < n; ++i) {   // if (prices[i] < minPrice) {   // minPrice = prices[i];   // } else if (prices[i] - minPrice > maxProfit) {   // maxProfit = prices[i] - minPrice;   // }   // }   // cout << maxProfit << endl;  }    int main() {   int T;  // cin>>T;   T = 1;   while (T--) {   solve();   }   return 0;  } |  |  | | --- | | Task3代码 | | #include <bits/stdc++.h>    using namespace std;    typedef long long ll;    const int N = 1e5;    ll steps[N];    void solve() {   // 1 or 2   steps[0] = 1;   steps[1] = 1;   steps[2] = 2;   steps[3] = steps[3 - 2] + steps[3 - 1];   int n;   cin >> n;   for (int i = 4; i <= n; i++) {   steps[i] = steps[i - 2] + steps[i - 1];   }   cout << steps[n] << endl;     // 1...n   // steps[0] = 0;   // int step\_n;//最高可跳n级   // cin >> step\_n;   // for (int i = 1; i <= step\_n; i++)   // steps[i] = steps[i - 1] + 1;   // int n;   // cin >> n;   // for (int i = step\_n + 1; i <= n; i++) {   // steps[i] = 0;   // for (int j = 1; j <= step\_n; j++)   // steps[i] += steps[i - j];   // }   // cout << steps[n] << endl;  }    int main() {   int T;  // cin>>T;   T = 1;   while (T--) {   solve();   }   return 0;  } | |
| **实验的体会与建议** |
| 学习了分治和递归算法，了解了斐波那契数列及其常见的提醒，收获颇丰。 |