实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 实验名称 | 实验二 动态规划算法设计 | | |
| 实验日期 | 2023年4月4日 |  |  |
| 学 号 | 2021213193 | 姓 名 | 李田 |
| 专业班级 | 2021级软件工程 2班 | | |
| 指导教师 | 谷志新 | | |

东北林业大学

软件工程专业

|  |
| --- |
| 1. 实验目的   掌握动态规划算法的基本思想及适用条件，掌握动态规划算法的设计步骤和具体实现。 |
| 1. 实验环境   Windows 7 以上操作系统，PC机，codeblocks环境 |
| 三、实验内容及结果  **1、 数字三角问题**  问题描述：给定一个由n行数字组成的数字三角形，如下图所示  7  3 8  8 1 0  2 7 4 4  4 5 2 6 5  试设计一个算法，计算出从三角形的顶至底的一条路径，使该路径经过的数字总和最大。  如上图最大值为30=7+3+8+7+5  #include<bits/stdc++.h>  using namespace std;  typedef long long LL;  #define PLL pair<LL,LL>  #define PII pair<int,int>  #define rep(i,a,b) for(int i=a;i<=b;i++)  #define dep(i,a,b) for(int i=a;i>=b;i--)  #define lowbit(x) (x&-x)  #define x first  #define y second  const double eps=1e-8;  clock\_t startTime;  #define l first  #define r second  double getCurrentTime() {      return (double)(clock() - startTime) / CLOCKS\_PER\_SEC;  }  const int N=110;  int f[N][N];  int Data[N][N];  vector<char>ans2;  void dfs(int cx,int cy)  {    if(cx==1&&cy==1) return ;    if(f[cx-1][cy]>f[cx-1][cy-1]&&cx>=2)    {      ans2.push\_back('L');      dfs(cx-1,cy);    }else ans2.push\_back('R'),dfs(cx-1,cy-1);  }  void Triangle\_sum(int n)  {      for(int i=1;i<=n;i++)    {      for(int j=1;j<=i;j++)      {        f[i][j]=max(f[i-1][j],f[i-1][j-1])+Data[i][j];      }      }    int mx=0,my=0,Max=0;    for(int i=1;i<=n;i++)    {      if(f[n][i]>Max)      {        Max=f[n][i];        mx=n,my=i;      }    }    cout<<"MAX=="<<Max<<endl;    dfs(mx,my);    }  // 7  // 3 8  // 8 1 0  // 2 7 4 4  // 4 5 2 6 5  void solve2()  {     int n;     cin>>n;     rep(i,1,n)rep(j,1,i) cin>>Data[i][j];     Triangle\_sum(n);     for(int i=ans2.size()-1;i>=0;i--)     {      cout<<ans2[i]<<" \n"[i==0];     }  }  int main()  {      int tt;      tt=1;      startTime = clock();      while(tt--)      {          solve2();          //printf("%.2lf",getCurrentTime());      }  }    // 运行结果  7  3 8  8 1 0  2 7 4 4  4 5 2 6 5      **2、最长公共子序列问题**  问题描述：给定两个序列X={x1,x2,...,xm}和Y={y1,y2,...,yn},找出X和Y的最长公共子序列。  输入：  第1行：两个子序列的长度，m n  第2行：第1个子序列的各个元素（序列下标从1开始）  第3行：第2个子序列的各个元素（序列下标从1开始）  输出：  最长公共子序列  实例：  输入：  第1行：  4 5 //m和n的值  第2行  abad //输入4个字符，下标从1开始  第3行  baade //输入5个字符，下标从1开始  输出：  Aad  #include<bits/stdc++.h>  using namespace std;  typedef long long LL;  #define PLL pair<LL,LL>  #define PII pair<int,int>  #define rep(i,a,b) for(int i=a;i<=b;i++)  #define dep(i,a,b) for(int i=a;i>=b;i--)  #define lowbit(x) (x&-x)  #define x first  #define y second  const double eps=1e-8;  clock\_t startTime;  #define l first  #define r second  double getCurrentTime() {      return (double)(clock() - startTime) / CLOCKS\_PER\_SEC;  }  const int N=110;  int f[N][N];  int n,m;  string a,b;  // 表示a中以i结尾，b中以j结尾的最长公共子序列  void  maxCommonSubstring()  {    for(int i=1;i<=n;i++)    {      for(int j=1;j<=m;j++)      {        f[i][j]=max(f[i-1][j],f[i][j-1]);        if(a[i]==b[j])f[i][j]=max(f[i-1][j-1]+1,f[i][j]);      }    }    // f[i-1][j-1],f[i][j-1],f[i-1][j];    vector<char>ans;    while(f[n][m]&&n&&m)    {        if(f[n][m]==f[n-1][m])      {        n--;      }else if(f[n][m]==f[n][m-1])      {        m--;      }else      {         ans.push\_back(a[n]);        n--,m--;      }    }    for(int i=ans.size()-1;i>=0;i--)    {      cout<<ans[i];    }  }  void solve3()  {    cin>>n>>m;    cin>>a>>b;    a=' '+a;    b=' '+b;    maxCommonSubstring();  }  int main()  {      int tt;      tt=1;      startTime = clock();      while(tt--)      {          solve3();          //printf("%.2lf",getCurrentTime());      }  }  //运行结果    **3、日常购物**  问题描述：小明今天很开心，因为在家买的新房子即将拿到钥匙。新房里面有一间他自己专用的、非常宽敞的房间。让他更高兴的是，他的母亲昨天对他说：“你的房间需要购买什么物品？怎么布置，你说了算，只要他们的价格总和不超过N元钱”。小明今天早上开始预算，但他想买太多的东西，肯定会超过母亲的N元限额。因此，他把对每件物品的渴望程度，分为5等级：用整数1->5表示，第5等表示最想要。他还从互联网上找到了每件商品（所有整数）的价格。他希望在不超过N元（可能等于N元）的情况下，将每件商品的价格与效益度的乘积的总和最大化.  设第j件物品的价格为p[j]，重要度为w[j]，其选中的k件商品，编号依次为j1，j2，……，jk，则所求的总和为：  p[j1]×w[j1]+p[j2]×w[j2]+ …+p[jk]×w[jk]。  请帮小明设计一个符合要求的购物清单。  其中N=2000,K=6  p[1]=200 w[1]=2  p[2]=300 w[2]=2  p[3]=600 w[3]=1  p[4]=400 w[4]=3  p[5]=1000 w[5]=4  p[6]=800 w[6]=5  //代码实现  #include <bits/stdc++.h>  using namespace std;  typedef long long LL;  #define PLL pair<LL, LL>  #define PII pair<int, int>  #define rep(i, a, b) for (int i = a; i <= b; i++)  #define dep(i, a, b) for (int i = a; i >= b; i--)  #define lowbit(x) (x & -x)  #define x first  #define y second  const double eps = 1e-8;  clock\_t startTime;  #define l first  #define r second  double getCurrentTime()  {    return (double)(clock() - startTime) / CLOCKS\_PER\_SEC;  }  const int N = 2020;  int p[N], w[N];  int f[N][N];  void solve()  {    int n, m;    cin >> m >> n;    rep(i, 1, n) cin >> p[i] >> w[i];    rep(i, 1, n)    {      rep(j, p[i], m)      {        f[i][j] = max(f[i][j], f[i - 1][j - p[i]] + p[i] \* w[i]);      }    }    cout << f[n][m] << endl;    vector<int> ans;    int i = n, j = m;    while (i > 1 && j >= 0)    {      if (f[i][j] - p[i] \* w[i] == f[i - 1][j - p[i]])      {        ans.push\_back(i);        j -= p[i];        i--;      }      else        i--;    }    cout << "ans.size()==" << ans.size() << endl;    cout << "商品信息：" << endl;    for (int i = ans.size() - 1; i >= 0; i--)    {      cout << ans[i] << " " << p[ans[i]] << " " << w[ans[i]] << endl;    }  }  int main()  {    int tt;    tt = 1;    startTime = clock();    while (tt--)    {      solve();      // printf("%.2lf",getCurrentTime());    }  }  // 2000 6  // 200 2  // 300 2  // 600 1  // 400 3  // 1000 4  // 800 5  //运行结果： |

|  |
| --- |
| 四、实验过程分析与讨论  在动态规划求路径是遇到了并解决了以下问题：   * + - 1. 第一题里的最优路径的终点一定是最后一行的某一个元素，而不是第n行第n列       2. 第二题里的最优路径只有当f[n][m]==f[n-1][m-1]时才能进行递归处理。       3. 第三题里的最优路径的递归路径不要忘记不合法的情况，那个时候直接i—即可。 |
| 五、指导教师意见  指导教师签字：  年 月 日 |