**20230525第四次作业**

**作业1：Bone collector (hdu 2602)**

问题指达：骨头收集者带着体积为C的背包去检骨头，已知每块骨头的体积和价值，求能装进背包的最大价値。

输入：第1行输人测试数量；后面每3行为一个测试，其中第1行輪入骨头数量N和背包体积C。第2行输入每块骨头的价值w，第3行输入每块骨头的体积。

输出：最大价値。

数据范围：N≤1000,C≤1000。

（1）递推代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | #include<bits/stdc++.h>  using namespace std;  const int N = 1011;  int w[N], c[N]; // 物品的价值和体积  int dp[N][N];  int solve(int n, int C){  for(int i=1; i<=n; i++)  for(int j=0; j<=C; j++){  if(c[i] > j) dp[i][j] = dp[i-1][j]; //第i个物品比背包还大，装不了  else dp[i][j] = max(dp[i-1][j],dp[i-1][j-c[i]]+w[i]); //第i个物品能装  }  return dp[n][C];  }  int main(){  int T; cin>>T;  while(T--){  int n,C; cin >> n >> C;  for(int i=1;i<=n;i++) cin>>w[i];  for(int i=1;i<=n;i++) cin>>c[i];  memset(dp,0,sizeof(dp)); //清0，置初值为0  cout << solve(n, C) << endl;  }  return 0;  } |

（2）记忆化代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | int solve(int i, int j){ //前i个物品，放进容量j的背包  if (dp[i][j] != 0) return dp[i][j]; //记忆化  if(i == 0) return 0;  int res;  if(c[i] > j) res = solve(i-1,j); //第i个物品比背包还大，装不了  else res = max(solve(i-1,j), solve(i-1,j-c[i])+w[i]); //第i个物品可以装  return dp[i][j] = res;  } |

作业2：

ACboy needs your help (hdu 1712)

问题描述：ACboy这学期可以选N门课，他只想学 M 天。每门课的学分不同，问这N天如何安排N 门课，才能得到最多学分？

输入：有多个测试。每个测试的第1行输入N 和M。后面有N 行，每行輪入M个数字，表示一个短阵 A[i][j],1≤i≤N≤100,1≤j＜M<100,表示第i门课学j天能得到A[i][j]学分。若N=M=0，表示测试结束。

输出：对每个测试,输出最多学分。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | #include<bits/stdc++.h>  using namespace std;  const int N=105;  int w[N][N],c[N][N]; //物品的价值、体积  int dp[N];  int n,m;  int main(){  while(scanf("%d%d",&n,&m) && n && m){ //输入：n门课=n组，m天=容量m  for(int i=1;i<=n;i++) //第i门课，即第i组  for(int k=1;k<=m;k++){ //m天=容量m；在这题中，m也是第i组的物品个数  scanf("%d",&w[i][k]); //第i组第k个物品的价值  c[i][k] = k; //第i组第k个物品的体积。学k天才能得分，体积就是k  }  memset(dp,0,sizeof(dp));  for(int i=1;i<=n;i++) //n门课=n组。遍历每门课，即遍历每个组  for(int j=m;j>=0;j--) //容量=m  for(int k=1;k<=m;k++) //用k遍历第i组的所有物品  if(j >= c[i][k]) //第k个物品能装进容量j的背包  dp[j] = max(dp[j], dp[j-c[i][k]] + w[i][k]);//第i组第k个  printf("%d\n",dp[m]);  }  } |

作业3：

手机号码(洛谷 P4124)

问题描述：选手机号码，号码必须同时包含两个特征：手机号码中至少要出现3个相邻的相同数字；号码中不能同时出现8和4。例如，满足系件的号码有13000988721、23333333333、14444101000；而不滿足条件的号码有：1015400080、10010012022。手机号一定是11位数，不含前前导0。给出两个效a和b，统计出[a,b]区同内所有满足条件的号码数量。a和b也是11 位的手机号码。

输入：两个整数a和b,1010≤a≤b≤1011

输出：输出一个整数表示答案。

输入样例：

12121284000 12121285550

输出样例：

5

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | //改写自：www.luogu.com.cn/blog/yushuotong-std/solution--4124  #include<bits/stdc++.h>  using namespace std;  typedef long long ll;  ll dp[15][11][11][2][2][2];  int num[15];  ll dfs(int pos,int u,int v,bool state,bool n8,bool n4,bool limit){  ll ans=0;  if(n8 && n4) return 0; //8和4不能同时出现  if(!pos) return state;  if(!limit && dp[pos][u][v][state][n8][n4]!=-1) return dp[pos][u][v][state][n8][n4];  int up = (limit?num[pos]:9);  for(int i=0;i<=up;i++)  ans += dfs(pos-1,i,u,state||(i==u&&i==v),n8||(i==8),n4||(i==4),limit&&(i==up));  if(!limit) dp[pos][u][v][state][n8][n4]=ans;  return ans;  }  ll solve(ll x){  int len = 0;  while(x){num[++len]=x%10; x/=10;}  if(len!=11) return 0;  memset(&dp,-1,sizeof(dp));  ll ans = 0;  for(int i=1;i<=num[len];i++) //最高位1~9，避开前导0问题  ans+= dfs(len-1,i,0,0,i==8,i==4,i==num[len]);  return ans;  }  int main(){  ll a,b; cin >> a>>b;  cout <<solve(b)-solve(a-1);  } |

作业4：

刚5.16 没有上司的舞会（洛谷 P1352）

-问题描述：某单位有n个职员。编号为1~n。他们之间有从属关系。也就是说他们的关系就像一棵以老板为根的树。父结点就是子节点的直校上司。现在有一个周年庆宴会，宴会每邀请来一个职员都会增加一定的快乐指数ri。但是如果某个职员的直接上司来参加舞会，那么这个职员就无论如何也不肯来参加舞会了。编程计算邀请哪些职员可以使快乐指数最大，求最大的快乐指数。

输入：第1行输入一个整数n。第2～n+1行中，每行输入一个整数，第i+1行的整数标识职员i的快乐指教ri。第n十2~2n行中，每行输入两个整数l和k，代表k是l的直接上司。

输出：输出一行一个整数代表最大的快乐指数。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | #include<bits/stdc++.h>  using namespace std;  const int N = 6005;  int val[N], dp[N][2], father[N];  vector <int> G[N];  void addedge(int from,int to){  G[from].push\_back(to); //用邻接表建树  father[to] = from; //父子关系  }  void dfs(int u){  dp[u][0] = 0; //赋初值：不参加宴会  dp[u][1] = val[u]; //赋初值：参加宴会  //for(int i=0;i<G[u].size();i++){ //遍历u的邻居v。逐一处理这个父结点的每个子结点  // int v = G[u][i];  for(int v : G[u]){ //这一行和上面两行的作用一样  dfs(v); //深搜子结点  dp[u][1] += dp[v][0]; //父结点选择，子结点不选  dp[u][0] += max(dp[v][0], dp[v][1]); //父结点不选，子结点可选可不选  }  }  int main(){  int n; scanf("%d",&n);  for(int i=1;i<=n;i++) scanf("%d",&val[i]); //输入快乐指数  for(int i=1;i<n;i++){  int u,v; scanf("%d%d",&u,&v);  addedge(v,u);  }  int t = 1;  while(father[t]) t = father[t]; //查找树的根结点  dfs(t); //从根结点开始，用dfs遍历整棵树  printf("%d\n", max(dp[t][0], dp[t][1]));  return 0;  } |