**★★★★☆**

**題組：Problem Set Archive** **with Online Judge**

**題號：10149 : Yahtzee**

**解題者：黃嘉彥**

**解題日期：2023年5月18日**

**題意：**

一次擲5個骰子，共值13次，每次依計分方法算分，計分方法也有13種，每種方法只能使用一次，若前六種方法的分數總和超過63分，則可以獲得額外35的獎勵分數。輸入會給分別13次的值骰點數，要輸出能取得的最高分，以及各個方法所獲得的分數。

方法與計分方式：

• 一 – 數字1的點數的總和

• 二 – 數字2的點數的總和

…

• 六 – 數字6的點數的總和

• 機會 – 所有骰子點數的總和

• 三條 – 所有骰子點數的總和，必須至少有3個骰子點數相同

• 鐵支 – 所有骰子點數的總和，必須至少有4個骰子點數相同

• 五條 - 50 分，5個骰子點數相同

• 短順子 – 25 分, 必須至少4個骰子排成一列

( 1,2,3,4 or 2,3,4,5 or 3,4,5,6 )

• 長順子 – 35 分，必須所有骰子排成一列 (1,2,3,4,5 or 2,3,4,5,6)

• 葫蘆 – 40 分, 必須有3個骰子點數一樣 + 另外兩個點數也一樣

**題意範例：**

|  |
| --- |
| 1 2 3 4 5  1 2 3 4 5  1 2 3 4 5… \*13 ➢ 1 2 3 4 5 0 15 0 0 0 25 35 0 0 90 |
| 1 1 1 1 1  6 6 6 6 6  6 6 6 1 1  1 1 1 2 2  1 1 1 2 3  1 2 3 4 5  1 2 3 4 6  6 1 2 6 6  1 4 5 5 5  5 5 5 5 6  4 4 4 5 6  3 1 3 6 3  2 2 2 4 6 ➢ 3 6 9 12 15 30 21 20 26 50 25 32 40 35 327 |

**解法：**

要計算所有可能性，就要計算13!次，時間過長，這題要使用動態規劃。另外還要結合位元壓縮，因為每組骰子都只能使用1次，若未使用為1，不使用為0，可將其轉為用數字表示，一共有13種方法，若所有骰子完全未使用，其數字就為1111111111111，而全部方法用完的情況為0000000000000，只有1 or 0，可用2進制表示，轉成10進制的後就是0~8191，此題解題方法就是宣告一個13(種方法數)\* 8192(種組合)，的dp陣列，表示其在剩餘n種方法時，目前已消耗掉骰子組的最佳分配分數，可以表示成：

|  |
| --- |
| dp[n][8191] =  max( dp[n-1][8191 – 第i個bit] + 計分函式(n, 骰子組[第i個bit]) ) |

另外題目有規定額外bonus，使用這種方法，只需在n = 6 時計算分數是否>=63即可。最後，題目有要求印出過程，只需在宣告一個額外的記憶陣列mem[13][8192][14]來記錄過程即可。

**解法範例：**

無。

**討論：**

DP新手，寫到崩潰。memory和bouns最難!

**程式：**

|  |
| --- |
| #include<iostream>  #include<algorithm>  #define n\_dices 5  #define n\_method 13  using namespace std;  int memory[13][8192][14] = {0};  int score\_count(int input[n\_dices],int method){  int ret = 0;  if(method<6){  for(int i=0;i<n\_dices;++i){  if(input[i]==method+1){  ret += method+1;  }  }  }  else{  bool check = false;  int tempcount[6] = {0};  switch(method){  case 6:  for(int i=0;i<n\_dices;++i){  ret += input[i];  }  break;  case 7:  for(int i=0;i<n\_dices;++i){  ret += input[i];  tempcount[input[i]-1]++;  }  for(int j=0;j<6;j++){  if(tempcount[j] >=3){  check = true;  break;  }  }  if(!check) ret = 0;  break;  case 8:  for(int i=0;i<n\_dices;++i){  ret += input[i];  tempcount[input[i]-1]++;  }    for(int j=0;j<6;j++){  if(tempcount[j] >=4){  check = true;  break;  }  }  if(!check) ret = 0;  break;  case 9:  if(input[0] == input[1] &&  input[2] == input[3]&&  input[3] == input[4]&&  input[0] == input[2]) ret = 50;  break;  case 10:  if(input[0]+1 == input[1] &&  input[1]+1 == input[2]&&  input[2]+1 == input[3]) ret = 25;  else if(input[3]+1 == input[4] &&  input[1]+1 == input[2]&&  input[2]+1 == input[3]) ret = 25;  break;  case 11:  if(input[0]+1 == input[1] &&  input[1]+1 == input[2]&&  input[2]+1 == input[3]&&  input[3]+1 == input[4]) ret = 35;  break;  case 12:  for(int i=0;i<n\_dices;++i){  tempcount[input[i]-1]++;  }  bool have3 = false;  bool have2 = false;  for(int i=0;i<6;++i){  if(tempcount[i] == 2) have2 = true;  if(tempcount[i] == 3) have3 = true;  }  if(have2 && have3) ret = 40;  break;  }  }  return ret;  }  //dp[13][b'1111111111111] == 13全部沒用過, dp[n][m]: 第n個步驟用地m排骰子  void dp\_init(int x[13][8192]){  for(int i=0;i<13;++i){  for(int j=0;j<8192;++j){  x[i][j] = -1;  }  }  }  int dp\_run(int dp[13][8192],int dice[13][5],int id\_method, int id\_round){  if(id\_round <= 0 || id\_method<0) return 0;    if(dp[id\_method][id\_round] != -1)return dp[id\_method][id\_round];    int dp\_max\_now = -999;  int maxi;  for(int i=0;i<13;++i){  if((id\_round&(1<<i)) != 0){  int temp = dp\_run(dp,dice,id\_method-1,id\_round - (1<<i)) + score\_count(dice[i],id\_method);  if(temp > dp\_max\_now){  dp\_max\_now = temp;  maxi = i;  }    }  }    if(id\_method >0){  for(int i=0;i<14;i++){  memory[id\_method][id\_round][i] = memory[id\_method-1][id\_round - (1<<maxi)][i];  }  }  memory[id\_method][id\_round][id\_method] = score\_count(dice[maxi],id\_method);    if(id\_method == 5){  if(dp\_max\_now >= 63){  dp\_max\_now += 35;  memory[id\_method][id\_round][13] = 35;  }  else{  memory[id\_method][id\_round][13] = 0;  }  }    dp[id\_method][id\_round] = dp\_max\_now;  return dp\_max\_now;  }  int dp\_run\_btup(int dp[13][8192],int dice[13][5],int id\_method, int id\_round){  ;  }  int run(){  int input[13][5];  int dp[13][8192];  while(cin>>input[0][0]>>input[0][1]>>input[0][2]>>input[0][3]>>input[0][4]){  sort(input[0],input[0]+5);  for(int i=1;i<13;++i){  cin>>input[i][0]>>input[i][1]>>input[i][2]>>input[i][3]>>input[i][4];  sort(input[i],input[i]+5);  }      dp\_init(dp);  dp\_run(dp,input,12,8191);  for(int i=0;i<14;i++){  cout<<memory[12][8191][i]<<" ";  }  cout<<dp[12][8191]<<endl;    // int sum = 0;  // for(int i=0;i<13;i++){  // int temp = score\_count(input[i],memory[12][8191][i]);  // sum+=temp;  // cout<<temp<<" ";  // if(i==5){  // cout<<" (bonus: "<<sum<<" )";  // }  // }  // cout<<"s: "<<sum<<endl;    }  return 0;  };  int main(){  run();  // int input[13][5];  // int dp[13][8192];  // while(cin>>input[0][0]>>input[0][1]>>input[0][2]>>input[0][3]>>input[0][4]){  // sort(input[0],input[0]+5);  // for(int i=1;i<13;++i){  // cin>>input[i][0]>>input[i][1]>>input[i][2]>>input[i][3]>>input[i][4];  // sort(input[i],input[i]+5);  // }  // dp\_init(dp);  // dp\_run(dp,input,12,8191);  // for(int i=0;i<14;i++){  // cout<<memory[12][8191][i]<<" ";  // }  // cout<<dp[12][8191]<<endl;  // }  return 0;  } |