### 斗地主

有人说这题是不用考虑什么的大搜索。而且由于数据是随机生成的，所以大部分可能不那么严谨的算法都可以过了。

#### 思路分析

不过大概还是有几个有用的点：

1. 花色几乎是没有用的，这里题目描述的不是很清楚，没有说两张大王不能当对子。

如果当可以的话代码中就可以只考虑码数了。

1. 因为“2”这张牌不能当做顺子，所以它没有什么作为“2”的必要，而“A”可以排在JQK后面作为顺子，我们可以在读入的时候将所有的牌的码数转换一下：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 王 | A | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | J | Q | K |
| 0 | 12 | 13 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |

这样在代码中可以大大简化顺子的部分以及减少很多特判。

1. 剪枝。常用的剪枝是一旦当前使用的步数大于等于ans而牌还没有打完就直接回退。
2. 代码顺序。这里采用一点点贪心的技巧，将可以一次打出较多牌的选法排在前面，这样在后面的运行中可以大量的剪枝。
3. 处理单牌和对子。单牌和对子在最后会剩下比较多，所以如果直接继续递归的话会浪费很多时间。我们可以发现只剩单牌和对子时最少的步骤就是依次将每种码数的牌全部打出，答案也就是有余牌的码数个数。

#### 具体代码

|  |
| --- |
| /\*  Landlords  \*/  #include<iostream>  #include<cstring>  #include<cstdio>  #include<cstdlib>  **using** **namespace** std;  inline void read(int& A)  {  /\*  char r;  for(r=getchar();r<48||r>57;r=getchar());  for(A=0;r>=48&&r<=57;r=getchar())A=A\*10+r-48;  \*/  scanf("%d",&A);  }  const int maxn=25;  const int inf=2147483647;  int T,n,ans;  int card[14];  inline int recode(int k)  {  **if**(k==1)**return** 12;  **if**(k>=3&&k<=13)  **return** k-2;  **if**(k==0)**return** 0;  **if**(k==2)**return** 13;  }  void input()  {  int num,color;  **for**(int i=1;i<=n;++i)  {  read(num),read(color);  ++card[recode(num)];  }  }  void dfs(int t,int rest)  {  //cout<<t<<' '<<rest<<endl;  **if**(t>ans)**return**;    **if**(rest==0)  {  **if**(t<ans)ans=t;  **return**;  }  int cnt[24];  memset(cnt,0,**sizeof**(cnt));  //init  **for**(int i=0;i<=13;++i)  {  **for**(int j=card[i];j>=1;--j)  {  ++cnt[j];  }  }  //if(cnt[1]+t>ans)return; it's wrong  //4 carry 2  **if**(rest>=6)  {  **for**(int i=0;i<=13&&cnt[4];++i)  {  **if**(card[i]>=4)  {  card[i]-=4;  **for**(int r=1;r<=2;++r)  {  //choose two different  **for**(int j=0;j<=13&&cnt[r];++j)  {  **if**(card[j]>=r)  {  card[j]-=r;  **for**(int k=0;k<=13;++k)  {  **if**(k!=j&&card[k]>=r)  {  card[k]-=r;  dfs(t+1,rest-4-2\*r);  card[k]+=r;  }  }  card[j]+=r;  }  }  //choose two same  **for**(int j=0;j<=13&&cnt[2\*r];++j)  {  **if**(card[j]>=2\*r)  {  card[j]-=2\*r;  dfs(t+1,rest-4-2\*r);  card[j]+=2\*r;  }  }  }  dfs(t+1,rest-4);  card[i]+=4;  }  }  // 3\*3+  **for**(int i=1;i<=11&&cnt[3]>=2;++i)  {  **if**(card[i]>=3)  {  int yan;  **for**(yan=i+1;yan<=12&&card[yan]>=3;++yan);  **for**(int j=yan-1;j>=i+1;--j)  {  **for**(int k=i;k<=j;++k)  {  card[k]-=3;  }  dfs(t+1,rest-(j-i+1)\*3);  **for**(int k=i;k<=j;++k)  {  card[k]+=3;  }  }  i=yan;  }  }    //3\*2+  **for**(int i=1;i<=10&&cnt[2]>=3;++i)  {  **if**(card[i]>=2)  {  int yan;  **for**(yan=i+1;yan<=12&&card[yan]>=2;++yan);  **for**(int j=yan-1;j>=i+2;--j)  {  **for**(int k=i;k<=j;++k)  {  card[k]-=2;  }  dfs(t+1,rest-(j-i+1)\*2);  **for**(int k=i;k<=j;++k)  {  card[k]+=2;  }  }  i=yan;  }  }  }  // 5\*1+  **if**(rest>=5)  {  **for**(int i=1;i<=8&&cnt[1]>=5;++i)  {  int yan;  **if**(card[i]>=1)  {  **for**(yan=i+1;yan<=12&&card[yan]>=1;++yan);  **for**(int j=yan-1;j>=i+4;--j)  {  **for**(int k=i;k<=j;++k)  {  --card[k];  }  dfs(t+1,rest-(j-i+1));  **for**(int k=i;k<=j;++k)  {  ++card[k];  }  }  }  }  // 3carry 2  **for**(int i=0;i<=13&&cnt[3]&&cnt[2];++i)  {  **if**(card[i]>=3)  {  card[i]-=3;  **for**(int j=0;j<=13&&cnt[2];++j)  {  **if**(card[j]>=2)  {  card[j]-=2;  dfs(t+1,rest-5);  card[j]+=2;  }  }  card[i]+=3;  }  }  }  //tnt 4  **if**(rest>=4)  {  //3carry 1  **for**(int i=0;i<=13&&cnt[3];++i)  {  **if**(card[i]>=3)  {  card[i]-=3;    **for**(int j=0;j<=13;++j)  {  **if**(j!=i&&card[j]>=1)  {  card[j]-=1;  dfs(t+1,rest-4);  card[j]+=1;  }  }  dfs(t+1,rest-3);  card[i]+=3;  }  }  }  // 2 and 1  **if**(cnt[3]==0)  {  dfs(t+cnt[1],0);  }  }  int main()  {  freopen("landlords.in","r",stdin);  //freopen("landlords.out","w",stdout);  read(T),read(n);  **for**(int e=1;e<=T;++e)  {  ans=inf;  memset(card,0,**sizeof**(card));  input();  dfs(0,n);  printf("%d\n",ans);  }  **return** 0;  } |