## 2-SAT算法

题意：给出N个开始时间和结束时间和持续时间三元组，持续时间可以在开始后或者结束前，问如何分配可以没有冲突。

|  |
| --- |
| #include <iostream>  #include <map>  #include <algorithm>  #include <cstdio>  #include <cstring>  #include <cstdlib>  #include <vector>  #include <queue>  #include <stack>  #include <functional>  #include <set>  #include <cmath>  **using** **namespace** std;  #define IOS std::ios::sync\_with\_stdio (false);std::cin.tie(0)  #define pb push\_back  #define PB pop\_back  #define bk back()  #define fs first  #define se second  #define sq(x) (x)\*(x)  #define eps (3e-7)  #define IINF (1<<29)  #define LINF (1ll<<59)  #define INF (1000000000)  #define FINF (1e3)  #define clr(x) memset((x),0,sizeof (x));  **typedef** **long** **long** ll;  **typedef** **unsigned** **long** **long** ull;  **typedef** pair<**int**,**int**> pii;  **typedef** pair<**int**,**int**> P;  **const** **int** maxn=2005;  **int** n;  **int** a[maxn][3];  **char** r[300];  vector<**int**> G[maxn],rG[maxn],od;  **bool** vis[maxn];  **int** sccid[maxn];  **int** get(**char** a,**char** b){  **return** (a-'0')\*10+b-'0';  }  **bool** inter(**int** a,**int** b,**int** c,**int** d){  **return** !(a>=d||b<=c);  }  **void** addedge(**int** a,**int** b){  G[a].pb(b);  rG[b].pb(a);  }  **void** dfs1(**int** v){  vis[v]=1;  **for**(**int** i=0;i<G[v].size();i++){  **int** u=G[v][i];  **if**(!vis[u]) dfs1(u);  }  od.pb(v);  }  **void** dfs2(**int** v,**int** k){  vis[v]=1;  sccid[v]=k;  **for**(**int** i=0;i<rG[v].size();i++){  **int** u=rG[v][i];  **if**(!vis[u]) dfs2(u,k);  }  }  **int** V;  **void** scc(){  clr(vis);od.clear();  **for**(**int** i=1;i<=V;i++){  **if**(!vis[i]) dfs1(i);  }  clr(vis);  **int** id=1;  **for**(**int** i=od.size()-1;i>=0;i--){  **int** v=od[i];  **if**(!vis[v]) dfs2(v,id++);  }  }  **void** build(){  **for**(**int** i=1;i<=n;i++){  **for**(**int** j=i+1;j<=n;j++){  **if**(inter(a[i][0],a[i][0]+a[i][2],a[j][0],a[j][0]+a[j][2])){  addedge(i,j+n);  addedge(j,i+n);  }  **if**(inter(a[i][0],a[i][0]+a[i][2],a[j][1]-a[j][2],a[j][1])){  addedge(i,j);  addedge(j+n,i+n);  }  **if**(inter(a[i][1]-a[i][2],a[i][1],a[j][0],a[j][0]+a[j][2])){  addedge(i+n,j+n);  addedge(j,i);  }  **if**(inter(a[i][1]-a[i][2],a[i][1],a[j][1]-a[j][2],a[j][1])){  addedge(i+n,j);  addedge(j+n,i);  }  }  }  }  **bool** ans[maxn];  **int** main(){  freopen("/home/slyfc/CppFiles/in","r",stdin);  *//freopen("defense.in","r",stdin);*  *//freopen("defense.out","w",stdout);*  cin>>n;  V=n\*2;  **for**(**int** i=1;i<=n;i++){  scanf("%s",r);  a[i][0]=get(r[0],r[1])\*60+get(r[3],r[4]);  scanf("%s",r);  a[i][1]=get(r[0],r[1])\*60+get(r[3],r[4]);  scanf("%d",&a[i][2]);  }  build();  scc();  **for**(**int** i=1;i<=n;i++){  **if**(sccid[i]==sccid[i+n]){  puts("NO");  **return** 0;  }**else**{  **if**(sccid[i]>sccid[i+n]){  ans[i]=1;  }**else**{  ans[i]=0;  }  }  }  puts("YES");  **for**(**int** i=1;i<=n;i++){  **if**(ans[i]){  **int** s=a[i][0],t=a[i][0]+a[i][2];  printf("%02d:%02d %02d:%02d**\n**",s/60,s%60,t/60,t%60);  }**else**{  **int** s=a[i][1]-a[i][2],t=a[i][1];  printf("%02d:%02d %02d:%02d**\n**",s/60,s%60,t/60,t%60);  }  }  **return** 0;  } |