# Shengchengshu

1639

/\*

最小度限制生成树

\*/

#include <iostream>

#include <cstdio>

#include <cstring>

using namespace std;

const int INF = 0x7fffffff;

const int N = 30;

int n,S,k; //节点总数 有度数限制的点v0 度数限制为k

int mst; //最终结果：最小k限制度生成树

int mp[N][N]; //图

int father[N]; //节点n的父节点

bool edge[N][N]; //判断边(i,j)是否加入到生成树中

int best[N]; //从v0到v路径上与v0无关的最大权边的点序号

char str[N][12];

int dis[N];

bool mark[N];

bool vis[N];

int pre[N];

void dfs(int now)

{

for(int i = 0; i < n; i++)

{

if(edge[i][now] && mark[i])

{

father[i] = now;

mark[i] = false;

dfs(i);

}

}

}

int prim(int s) //从点s开始的最小生成树

{

int i,Min,key;

int sum = 0;

memset(pre,0,sizeof(pre));

memset(mark,0,sizeof(mark));

mark[s] = true;

vis[s] = true;

for(i = 0; i < n; i++)

{

dis[i] = mp[s][i];

pre[i] = s;

}

while(true)

{

Min = INF;

for(i = 0; i < n; i++)

{

if(!vis[i] && !mark[i] && dis[i] < Min)

{

Min = dis[i];

key = i;

}

}

if(Min == INF) break;

mark[key] = true;

vis[key] = true;

edge[pre[key]][key] = edge[key][pre[key]] = true;

sum += Min;

for(i = 0; i < n; i++)

{

if(!vis[i] && !mark[i] && dis[i] > mp[key][i])

{

dis[i] = mp[key][i];

pre[i] = key;

}

}

}

Min = INF;

int root = -1; //找到与v0相关联的点的最小边

for(i = 0; i < n; i++)

{

if(mark[i] && mp[i][S] < Min)

{

Min = mp[i][S];

root = i;

}

}

mark[root] = false;

dfs(root); // 将树拉成有根树

father[root] = S;

return sum + Min;

}

int Best(int x) //记忆化搜索s到x的最大权值的边

{

int tmpt;

if(father[x] == S)

return -1;

if(best[x] != -1)

return best[x];

tmpt = Best(father[x]);

if(tmpt != -1 && mp[tmpt][father[tmpt]] > mp[father[x]][x])

best[x] = tmpt;

else

best[x] = x;

return best[x];

}

int find(char \*c)

{

for(int i = 0; i < n; i++)

{

if(strcmp(str[i],c) == 0)

return i;

}

return -1;

}

void input()

{

int i,j;

int m;

int w;

char s1[N],s2[N];

for(i = 0; i <= N-2; i++)

for(j = 0; j <= N-2; j++)

mp[i][j] = INF;

scanf("%d",&m);

n = 0;

strcpy(str[n++],"Park");

S = 0;

for(i = 0; i < m; i++)

{

scanf("%s %s %d",s1,s2,&w);

int x = find(s1);

if(x == -1)

{

x = n;

strcpy(str[n++],s1);

}

int y = find(s2);

if(y == -1)

{

y = n;

strcpy(str[n++],s2);

}

if(w < mp[x][y]) //可能有重边

mp[x][y] = mp[y][x] = w;

}

scanf("%d",&k);

}

void solve()

{

int i,j;

memset(vis,0,sizeof(vis));

memset(edge,0,sizeof(edge));

memset(father,-1,sizeof(father));

vis[S] = true;

int m = 0;

mst = 0;

//先求m度限制最小生成树

for(i = 0; i < n; i++)

{

if(!vis[i])

{

m++;

mst += prim(i);

}

}

int change; // 回路上权值最大的边，用于交换

int ax,bx,tmp;

for(i = m+1; i <= k && i < n; i++)

{

memset(best,-1,sizeof(best));

for(j = 0; j < n; j++)

{

if(best[j] == -1 && father[j] != S)

Best(j);

}

int minadd = INF; // 交换边的最小差值

for(j = 0; j < n; j++)

{

if(mp[S][j]!= INF && father[j] != S)

{

ax = best[j];

bx = father[ax];

tmp = mp[S][j] - mp[ax][bx];

if(tmp < minadd)

{

minadd = tmp;

change = j;

}

}

}

if (minadd >= 0) break; //用于度数不大于k的限制，如果k限制，就不用break了

mst += minadd;

ax = best[change];

bx = father[ax];

mp[ax][bx] = mp[bx][ax] = INF;

father[ax] = bx = S; // 改变生成树，将点ax直接指向源点S

mp[ax][S] = mp[S][ax] = mp[change][S];

mp[S][change] = mp[change][S] = INF;

}

}

int main()

{

input();

solve();

printf("Total miles driven: %d\n", mst);

return 0;

}

5629

#include<cstdio>

#include<iostream>

using namespace std;

typedef long long ll;

#define mod 1000000007ll

#define maxn 55

int T,n,a[maxn];

ll C[maxn][maxn],dp[maxn][maxn][maxn];

void init()

{

memset(C,0,sizeof(C));

C[0][0]=1;

for(int i=1;i<=50;i++)

{

C[i][0]=C[i][i]=1;

for(int j=1;j<i;j++)C[i][j]=(C[i][j-1]+C[i-1][j-1])%mod;

}

}

int main(){

init();

scanf("%d",&T);

while(T--)

{

scanf("%d",&n);

for(int i=1;i<=n;i++)scanf("%d",&a[i]);

memset(dp,0,sizeof(dp));

dp[0][0][0]=1;

for(int i=1;i<=n;i++)

for(int j=0;j<i;j++)

for(int k=0;k<=n-2;k++)

{

dp[i][j][k]=(dp[i][j][k]+dp[i-1][j][k])%mod;

for(int d=0;d<a[i]&&k+d<=n-2;d++)

dp[i][j+1][k+d]=(dp[i][j+1][k+d]+C[k+d][d]\*dp[i-1][j][k]%mod)%mod;

}

printf("%d",n);

for(int i=2;i<=n;i++)printf(" %I64d",dp[n][i][i-2]);

printf("\n");

}

return 0;

}