UVa753

题意：有nn个插座，mm个设备和kk种转换器，每种转换器都有无限多。已知每个插座的类型，每个设备的插头类型，以及每种转换器的插座类型和插头类型。要求插的设备尽量多。问最少剩几个不匹配的设备。 （dinic算法）

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  #define clr(a,x) memset(a, x, sizeof(a))  #define mp(x,y) make\_pair(x,y)  #define pb(x) push\_back(x)  #define X first  #define Y second  #define fastin ios\_base::sync\_with\_stdio(0);cin.tie(0);  typedef long long ll;  typedef long double ld;  typedef pair<int, int> PII;  typedef vector<int> VI;  const int INF = 0x3f3f3f3f;  const int mod = 1e9 + 7;  const double eps = 1e-6;  const int maxn = 405;  struct Edge  {  int from, to, cap, flow;  Edge(int u, int v, int c, int f): from(u), to(v), cap(c), flow(f) {}  };  struct Dinic  {  int n, m, s, t; //结点数，边数（包括反向弧），源点编号和汇点编号  vector<Edge> edges; //边表。edge[e]和edge[e^1]互为反向弧  vector<int> G[maxn]; //邻接表，G[i][j]表示节点i的第j条边在e数组中的序号  bool vis[maxn]; //BFS使用  int d[maxn]; //从起点到i的距离  int cur[maxn]; //当前弧下标  void init(int n)  {  this->n = n;  for (int i = 0; i < n; i++) G[i].clear();  edges.clear();  }  void AddEdge(int from, int to, int cap)  {  edges.pb(Edge(from, to, cap, 0));  edges.pb(Edge(to, from, 0, 0));  m = edges.size();  G[from].pb(m - 2);  G[to].pb(m - 1);  }  bool BFS()  {  clr(vis, 0);  clr(d, 0);  queue<int> q;  q.push(s);  d[s] = 0;  vis[s] = 1;  while (!q.empty())  {  int x = q.front();  q.pop();  for (int i = 0; i < G[x].size(); i++)  {  Edge& e = edges[G[x][i]];  if (!vis[e.to] && e.cap > e.flow)  {  vis[e.to] = 1;  d[e.to] = d[x] + 1;  q.push(e.to);  }  }  }  return vis[t];  }  int DFS(int x, int a)  {  if (x == t || a == 0) return a;  int flow = 0, f;  for (int& i = cur[x]; i < G[x].size(); i++)  {  //从上次考虑的弧  Edge& e = edges[G[x][i]];  if (d[x] + 1 == d[e.to] && (f = DFS(e.to, min(a, e.cap - e.flow))) > 0)  {  e.flow += f;  edges[G[x][i] ^ 1].flow -= f;  flow += f;  a -= f;  if (a == 0) break;  }  }  return flow;  }  int Maxflow(int s, int t)  {  this->s = s;  this->t = t;  int flow = 0;  while (BFS())  {  clr(cur, 0);  flow += DFS(s, INF);  }  return flow;  }  } ans;  map <string, int> M;  int main()  {  #ifndef ONLINE\_JUDGE  freopen("1.in", "r", stdin);  freopen("1.out", "w", stdout);  #endif  fastin  int T, n, m, k;  cin >> T;  while (T--)  {  M.clear();  cin >> n;  int s = 0, t = 1;  ans.init(maxn);  int tot = 2;  string tmp, dev;  while (n--)  {  //插座->超级汇点  cin >> tmp;  if (!M[tmp]) M[tmp] = tot++;  ans.AddEdge(M[tmp], t, 1);  }  cin >> m;  for (int i = 0; i < m; i++)  {  //超级源点->电器  cin >> dev >> tmp;  if (!M[tmp]) M[tmp] = tot++;  ans.AddEdge(s, M[tmp], 1);  }  cin >> k;  string u, v;  while (k--)  {  //转换器  cin >> u >> v;  if (!M[u]) M[u] = tot++;  if (!M[v]) M[v] = tot++;  ans.AddEdge(M[u], M[v], INF);  }  cout << m - ans.Maxflow(s, t) << endl;  if (T) cout << endl;  }  return 0;  } |

UVa11802

题意：定义一个R\*C的正整数矩阵（1<=R,C<=20),设Ai为前i行所有元素之和，Bi为前i列所有元素之和。   
题目已知R,C和数组A，B。要找一个满足条件的矩阵。矩阵中的元素要满足(1<=X[i][j]<=20)。

|  |
| --- |
| #include<bits/stdc++.h>  #define maxn 1000  #define INF (1<<31)-1  using namespace std;  struct Edge  {  int from,to,cap,flow;  Edge(int u,int v,int c,int f):  from(u),to(v),cap(c),flow(f) {}  };  struct Dinic  {  int n,m,s,t;  vector<Edge> edges;  vector<int> G[maxn];  int d[maxn];  int cur[maxn];  bool vis[maxn];  void init(int n)  {  for (int i=0; i<n; i++)  G[i].clear();  edges.clear();  }  void Addedge(int from,int to,int cap)  {  edges.push\_back(Edge(from,to,cap,0));  edges.push\_back(Edge(to,from,0,0));  m = edges.size();  G[from].push\_back(m-2);  G[to].push\_back(m-1);  }  bool BFS()  {  memset(vis,false,sizeof(vis));  for (int i=0; i<n; i++) d[i] = INF;  d[s] = 0; vis[s] = true;  queue<int> Q;  Q.push(s);  while (!Q.empty())  {  int x = Q.front();  Q.pop();  for (int i=0; i<G[x].size(); i++)  {  Edge e = edges[G[x][i]];  if (!vis[e.to] && e.cap>e.flow)  {  vis[e.to] = true;  d[e.to] = d[x]+1;  Q.push(e.to);  }  }  }  return vis[t];  }  int DFS(int x,int a)  {  if (x == t || a == 0)  return a;  int flow = 0,f;  for (int i=cur[x]; i<G[x].size(); i++)  {  Edge& e = edges[G[x][i]];  if (d[e.to] == d[x]+1 && (f = DFS(e.to,min(a,e.cap-e.flow))) > 0)  {  e. flow += f;  edges[G[x][i]^1].flow -= f;  flow += f;  a -= f;  if (a == 0)  break;  }  }  return flow;  }  bool OKA()  {  for (int i=0; i<G[s].size(); i++)  {  Edge e = edges[G[s][i]];  if (e.cap!=e.flow)  return false;  }  return true;  }  bool OKB(int R,int C)  {  for (int j=R+1; j<=R+C; j++)  {  Edge& e = edges[G[j][0]];  if (e.cap!=e.flow)  return false;  }  return true;  }  void Maxflow(int t,int R,int C)  {  int flow = 0;  while (BFS())  {  memset(cur,0,sizeof(cur));  flow += DFS(s,INF);  }  cout<<"Matrix "<<t<<endl;  if (OKA() && OKB(R,C))  {  for (int i=1; i<=R; i++)  {  int j;  for (j=1; j<G[i].size()-1; j++)  cout<<edges[G[i][j]].flow+1<<' ';  cout<<edges[G[i][j]].flow+1<<endl;  }  }  cout<<endl;  }  };  int main()  {  Dinic aa;  int T,R,C,tmp;  int a[30],b[30],c[30],d[30];  #ifndef ONLINE\_JUDGE  freopen("1.in", "r", stdin);  freopen("1.out", "w", stdout);  #endif  cin>>T;  tmp = T;  while (T>0)  {  T--;  aa.init(maxn);  cin>>R>>C;  for (int i=1; i<=R; i++) cin>>a[i];  for (int i=1; i<=C; i++) cin>>b[i];  for (int i=1; i<=R; i++) c[i] = a[i]-a[i-1]-C;  for (int i=1; i<=C; i++) d[i] = b[i]-b[i-1]-R;  for (int i=1; i<=R; i++)  aa.Addedge(0,i,c[i]);  for (int i=1; i<=C; i++)  aa.Addedge(R+i,R+C+1,d[i]);  for (int i=1; i<=R; i++)  for (int j=1; j<=C; j++)  aa.Addedge(i,R+j,19);  aa.s = 0; aa.t = R+C+1;  aa.Maxflow(tmp-T,R,C);  }  return 0;  } |