精確覆蓋問題

在一個全集X中若干子集的集合為S，精確覆蓋是指，S的子集S\*，滿足X中的每一個元素在S\*中恰好出現一次。在計算機科學中，精確覆蓋問題指找出這樣的一種覆蓋，或證明其不存在。這是一個NP-完全問題，也是卡普的二十一個NP-完全問題之一。定義

滿足以下條件的集合為一個精確覆蓋：

定義:

S\*中任意兩個集合沒有交集，即X中的元素在S\*中出現最多一次

S\*中集合的全集為X，即X中的元素在S\*中出現最少一次

合二為一，即X中的元素在S\*中出現恰好一次。

舉例

令 S = {N, O, E, P} 是集合X = {1, 2, 3, 4}的一個子集的集合，並滿足：

N = { }

O = {1, 3}

E = {2, 4}

P = {2, 3}.

其中一個子集 {O, E} 是 X的一個精確覆蓋,因為 O = {1, 3} 而 E = {2, 4} 的並集恰好是 X = {1, 2, 3, 4}。同理， {N, O, E} 也是 X.的一個精確覆蓋。空集並不影響結論。

關係表示:

通常我們用S的每個子集與X的元素之間包含關係的[二元關係](https://zh.wikipedia.org/wiki/%E4%BA%8C%E5%85%83%E5%85%B3%E7%B3%BB" \o "二元關係)來表示精確覆蓋問題。

**矩陣表示法**

包含關係可以用一個關係矩陣表示。. 矩陣每行表示S的一個子集，每列表示X中的一個元素。矩陣行列交點元素為1表示對應的元素在對應的集合中，不在則為0.

通過這種矩陣表示法，求一個精確覆蓋轉化為求矩陣的若干個行的集合，使每列有且僅有一個1。同時，該問題也是精確覆蓋的典型例題之一。

下圖為其中一個例子：

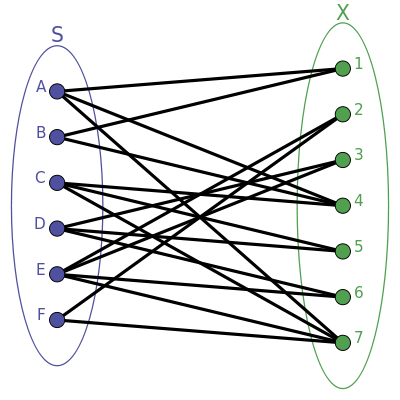
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| ***A*** | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| ***B*** | **1** | 0 | 0 | **1** | 0 | 0 | 0 |
| ***C*** | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| ***D*** | 0 | 0 | **1** | 0 | **1** | **1** | 0 |
| ***E*** | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| ***F*** | 0 | **1** | 0 | 0 | 0 | 0 | **1** |

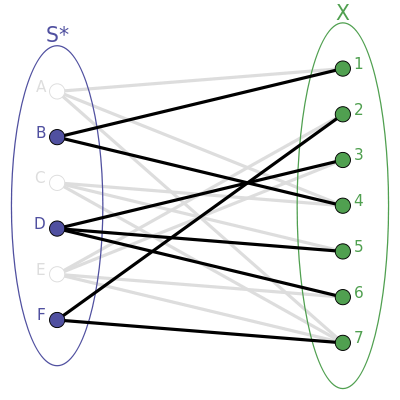
S\* = {*B*, *D*, *F*} 便是一個精確覆蓋。

**圖論表示法**

包含關係也可以用一個[二分圖](https://zh.wikipedia.org/wiki/%E4%BA%8C%E5%88%86%E5%9B%BE" \o "二分圖)表示。

二分圖左側每個節點表示S的每個集合，右側每個節點表示X的每個元素，而精確覆蓋便是一種匹配，滿足右側的每個點恰好有一條邊。

[](https://zh.wikipedia.org/wiki/File:Exact-cover-bigraph.svg)

[](https://zh.wikipedia.org/wiki/File:Exact-cover-bigraph-solved.svg)

程式碼:

#include<cstdio>

#include<vector>

#include<cstring>

#include<iostream>

using namespace std;

int \_;

int sudoku[10][10];

const int MAXN = 81\*4\*81+10,MAXR = 9\*9\*9+10,MAXC = 81\*4+10;

// 行編號從1開始，列編號為1~n，結點0是表頭結點; 結點1~n是各列頂部的虛擬結點

struct DLX {

//成員變數

int n, sz; // 列數，結點總數

int S[MAXC]; // 各列結點數

int row[MAXN], col[MAXN]; // 各結點行列編號

int L[MAXN], R[MAXN], U[MAXN], D[MAXN]; // 十字連結串列

vector<int>vec;

int ansd, ans[MAXR]; // 解

//成員函式

void init(int n);//n為列數

void remove(int c);

void restore(int c);

bool dfs(int d);//d為遞迴深度

bool solve();

void addRow(int r);

//數獨所需函式

void build();

void decode(int code,int &a,int &b,int &c);

inline int encode(int a,int b,int c);

void output();

inline int trans(int x,int y);

};

#define FOR(i,A,s) for(int i = A[s]; i != s; i = A[i])

void DLX::output() {

for(int i=0;i<ansd;i++){

int r,c,v;

decode(ans[i],r,c,v);

sudoku[r][c]=v;

}

for(int i=0;i<9;i++){

for(int j=0;j<9;j++){

printf("%d",sudoku[i][j]+1);

}

putchar('\n');

}

}

void DLX::addRow(int r) {

int first = sz;

for(int i = 0; i < vec.size(); i++) {

int c = vec[i];

// cout<<c<<" ";

L[sz] = sz - 1; R[sz] = sz + 1; D[sz] = c; U[sz] = U[c];

D[U[c]] = sz; U[c] = sz;

row[sz] = r; col[sz] = c;

S[c]++; sz++;

}

// cout<<endl;

R[sz - 1] = first; L[first] = sz - 1;

}

inline int DLX::encode(int a,int b,int c){

return 81\*a + b\*9+c+1;

}

inline int DLX::trans(int x,int y){

x = x/3;

y = y/3;

return x\*3+y;

}

void DLX::build() {

for(int i=0;i<9;i++) {

for(int j=0;j<9;j++) {

for(int k=0;k<9;k++){

if(sudoku[i][j]==-1||sudoku[i][j]==k){

//cout<<i<<" "<<j<<" "<<k<<endl;

vec.clear();

vec.push\_back(encode(0,i,j));

vec.push\_back(encode(1,i,k));

vec.push\_back(encode(2,j,k));

vec.push\_back(encode(3,trans(i,j),k));

addRow(encode(i,j,k));

}

}

}

}

return;

}

void DLX::init(int n) { // n是列數

this->n = n;

// 虛擬結點

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

U[i] = i; D[i] = i; L[i] = i-1, R[i] = i+1;

}

R[n] = 0; L[0] = n;

sz = n + 1;

memset(S, 0, sizeof(S));

}

void DLX::remove(int c) {

L[R[c]] = L[c];

R[L[c]] = R[c];

FOR(i,D,c)

FOR(j,R,i) { U[D[j]] = U[j]; D[U[j]] = D[j]; --S[col[j]]; }

}

void DLX::restore(int c) {

FOR(i,U,c)

FOR(j,L,i) { ++S[col[j]]; U[D[j]] = j; D[U[j]] = j; }

L[R[c]] = c;

R[L[c]] = c;

}

bool DLX::dfs(int d) {

if (R[0] == 0) { // 找到解

ansd = d; // 記錄解的長度

return true;

}

// 找S最小的列c

int c = R[0]; // 第一個未刪除的列

FOR(i,R,0) if(S[i] < S[c]) c = i;

remove(c); // 刪除第c列

FOR(i,D,c) { // 用結點i所在行覆蓋第c列

ans[d] = row[i];

FOR(j,R,i) remove(col[j]); // 刪除結點i所在行能覆蓋的所有其他列

if(dfs(d+1)) return true;

FOR(j,L,i) restore(col[j]); // 恢復結點i所在行能覆蓋的所有其他列

}

restore(c); // 恢復第c列

return false;

}

void DLX::decode(int code,int &a,int &b,int &c){

code--;

c = code%9;code/=9;

b = code%9;code/=9;

a = code;

}

bool DLX::solve() {

if(!dfs(0)) return false;

return true;

}

void input(){

for(int i=0;i<9;i++)

for(int j=0;j<9;j++)

scanf("%1d",&sudoku[i][j]),sudoku[i][j]--;

}

DLX dlx;

int main(){

scanf("%d",&\_);

while(\_--) {

input();

dlx.init(81\*4);

dlx.build();

bool ok = dlx.solve();

if(ok) {

dlx.output();

}

}

}

資料來源:

<https://www.itread01.com/content/1541166440.html>

<https://zh.wikipedia.org/wiki/%E7%B2%BE%E7%A1%AE%E8%A6%86%E7%9B%96%E9%97%AE%E9%A2%98>