PROGRESS SOLUSI PROBLEM ADACROW SPOJ

Raditya Yusuf Annaafi' (5025231033)

1. DESKRIPSI MASALAH

Pada masalah ini diminta untuk mencari jumlah minimum *scarecrows* untuk diletakkan pada *farmhouses* yang terhubung dengan ladang-ladang dan memastikan ladang-ladang tersebut terhindar dari serangan burung gagak.

ADACROW - Ada and Scarecrow

#np-hard #branch-and-bound #kernel

As you might already know, Ada the Ladybug is a farmer. She has multiple farmhouses and some fields which always connect two adjacent farmhouses. She plans to buy some scarecrows to scare crows!

A scarecrow placed on a farmhouse scares all crows from all adjacent fields. A crow on a field can be disastrous, so Ada has to arrange the scarecrows in such way that they cover all the fields. As scarecrows are pretty expensive she wants to minimize the number of them. Can you count it for her?

Note: Even thought it might look like that from description, the formed "graph" doesn't have to be planar! Also multi-fields and self-field are not allowed.

Input

The first line contains an integer $\mathbf{1 \le T \le 100}$

Each of the next **T** test-cases begins with two integers $\mathbf{0 \le M \le 150}$, $\mathbf{1 \le N \le 150}$, the number of farmhouses and the number of fields.

Each of next **M** lines contains two numbers $\mathbf{0 \le A_i, A_j < N}$, the farmhouses, which are connected by a field.

Output

For each test-case output the minimal number of scarecrows Ada has to buy, to cover all fields.

2. HIPOTESIS PENYELESAIAN

2.1. *Minimum Vertex Cover*

Masalah ini merupakan masalah NP-Hard. Oleh karena itu, langkah yang harus dilakukan adalah memetakan ADACROW ke suatu masalah NP-Hard. Setelah saya melihat beberapa masalah NP-Hard, saya merasa bahwa masalah ini mempunyai kemiripan dengan masalah NP-Hard, ***Minimum Vertex Cover***.

Dalam graf tak berarah, G = (V, E), dengan V adalah himpunan *vertex* dan E adalah himpunan *edge*. *Vertex Cover* (VC) adalah subhimpunan dari $S \subseteq V$ sedemikian rupa sehingga setiap *edge* di G memiliki setidaknya satu titik ujung di S (setiap *edge* terhubung ke minimal satu *vertex*). Inti permasalahan *Minimum Vertex Cover* adalah untuk menemukan jumlah minimum VC dalam suatu graf.

Masalah ADACROW meminta kita untuk menempatkan sesedikit mungkin *scarecrow* pada *farmhouses* dan memastikan tidak ada ladang yang diserang oleh burung gagak. Ladang-ladang yang terhubung dengan *farmhouse* yang memiliki *scarecrow* otomatis terhindar dari serangan burung gagak. Oleh karena itu,

ADACROW dapat dipetakan ke masalah *Minimum Vertex Cover*. Dengan memilih sesedikit mungkin *vertices* (*farmhouses*) dan memastikan semua *edge* (ladang-ladang) terhubung ke paling sedikit satu *vertex* (*farmhouse*).

2.2. *Exact Algorithm*

Exact algorithm adalah metode penyelesaian masalah yang menjamin dapat menemukan solusi yang optimal dalam waktu secepat mungkin, tetapi tidak dalam waktu polinomial, umumnya eksponensial dan terkadang subeksponensial.

2.3. *Approximation Algorithm*

Berbeda dengan *exact algorithm*, *approximation algorithm* tidak dapat memperoleh solusi yang optimal, melainkan hanya mendekati solusi optimal (*near-optimal*). Walaupun begitu, algoritma penyelesaian ini dapat berjalan dalam waktu polinomial.

2.4. *Exact* vs *Approximation*

Untuk masalah *Minimum Vertex Cover* solusi dapat dicari menggunakan kedua metode yang telah disebutkan sebelumnya. Jika menggunakan *exact*, solusi optimal dijamin dapat diperoleh, tetapi *runtime*-nya dapat "meledak" seiring bertambah besarnya *input*. Sebaliknya, jika menggunakan *approximation* solusi optimal mungkin tidak akan diperoleh, tetapi *runtime*-nya tidak akan "meledak" karena pertumbuhannya bersifat polinomial.

Dalam masalah ADACROW terdapat tag "***Branch and Bound* (B&B)**". B&B termasuk ke dalam kategori *exact algorithm* yang berarti solusi optimal dapat ditemukan dengan mengorbankan *runtime*. Berhubung ADACROW memiliki tag B&B, saya menyimpulkan bahwa dengan menggunakan *exact algorithm* (tepatnya B&B) *runtime* tetap tidak akan melebihi *time limit* pada masalah ADACROW. Oleh karena itu, saya akan memilih *exact algorithm* (tepatnya B&B) untuk menyelesaikan masalah ADACROW.

2.5. *Branch and Bound*

Branch and Bound (B&B) adalah metode pencarian dalam ruang solusi dengan membentuk ruang solusi menjadi bentuk *tree*. Setiap *node* pada *tree* berisi solusi parsial (kombinasi *vertex cover*) yang bercabang untuk menandai "*vertex* berikutnya" sebagai *vertex cover* atau bukan. Pemilihan "*vertex* berikutnya" mengacu pada urutan saat *edge* di-*input*.

Algoritma ini juga mempunyai batas-batas yang mencegah eksplorasi ke suatu *node* di dalam *tree* yang sekiranya tidak akan mengarah ke solusi optimal, di antaranya adalah berikut:

1. Jika semua tetangga suatu *vertex* merupakan *vertex cover*, *vertex* tersebut akan ditandai sebagai "bukan *vertex cover"* terlebih dahulu.
2. *Vertex* yang masih mempunyai tetangga yang bukan *vertex cover* tidak boleh ditandai sebagai "bukan *vertex cover*".
3. *Upper Bound*

   Jika ada sebuah graf $G = (V, E)$, jumlah *Minimum Vertex Cover* yang mungkin adalah bilangan bulat xN dengan $N = |V|$ dan $x \in [0, 1]$. Oleh karena itu, *upper bound* setiap pencarian bisa disebut dengan K, di mana K

adalah setiap kemungkinan xN yang ada. Jadi untuk setiap K yang ada akan dijalankan algoritma B&B untuk menemukan kombinasi *vertex cover* yang mungkin.

Untuk setiap *child node* dalam B&B *tree*, K akan dikurangi dengan 1 karena pada *parent* dari sebuah *node* sudah ditandai 1 *vertex cover* yang mengurangi jumlah *vertex* yang dapat ditandai agar tetap memenuhi K.

4. *Lower Bound*

   Untuk *lower bound*, pada setiap *node* akan dihitung jumlah paling sedikit *edges* yang belum ter-*cover* (M) untuk cabang berikutnya dengan mempertimbangkan derajat *vertices* yang belum ditandai sebagai *vertex cover* d(i), di mana derajat sebuah *vertex* adalah banyaknya jumlah tetangga *vertex* tersebut yang belum ditandai sebagai *vertex cover*.

   $$M = \max(0, E - \max(d(v_1) + d(v_2) + \cdots d(v_k)))$$

   Misal pada sebuah *node*, K yang tersisa adalah 3, akan dicari 3 derajat *vertices* terbesar. Kemudian jumlah *edges* yang belum ter-*cover* pada *node* tersebut (E) akan dikurangkan jumlah derajat tersebut. Dari kalkulasi tersebut akan dihasilkan M untuk memutuskan apakah percabangan masih boleh dilakukan atau tidak dengan membandingkan M dengan jumlah *edges* yang belum ter-*cover* pada solusi terbaik pada keseluruhan *tree*.

Dengan menerapkan batas-batas tersebut diharapkan dapat mengurangi percabangan yang tidak perlu dan dapat memangkas *runtime* agar pencarian solusi optimal dapat dilakukan dengan secepat mungkin.