CSD201- Data Structure & Algorithms using Java

Giảng viên: **Tiến sĩ Bùi Thanh Hùng**
Trưởng Lab Khoa học Phân tích dữ liệu và Trí tuệ nhân tạo
Giám đốc chương trình Trí tuệ nhân tạo và Hệ thống thông tin
Đại học Thủ Dầu Một
Email: **hungbt3t@fe.edu.vn**
Website: **https://sites.google.com/site/hungthanhbui1980**

I- YÊU CẦU

1- Thời gian thực hiện đến slot số 27, nộp bài trước 24h slot số 27 trên LMS
2- Chấm bài slot số 29
3- Nội dung nộp:
Báo cáo theo mẫu gửi kèm theo, có tự chấm điểm ở cuối
Dữ liệu chương trình
Source code giải quyết bài toán bằng Java
Tất cả nén vào 1 file, đặt tên theo quy tắc:
Mã lớp – Tên nhóm – Họ tên SV 1- Họ tên SV 2
và nộp vào đúng thư mục cần nộp, 2 bạn 1 nhóm chỉ cần 1 người nộp

II- NỘI DUNG BÁO CÁO

Trình bày các nội dung sau
1. Giới thiệu Bài toán
Bài toán này là bài toán gì, có ý nghĩa sao
2. Giải quyết bài toán
2.1 Cấu trúc dữ liệu
Trình bày chi tiết cấu trúc dữ liệu sử dụng trong bài toán
2.2 Sơ đồ giải thuật
Vẽ sơ đồ giải thuật tổng quát và từng phần sử dụng trong bài toán
Nếu bài toán có nhiều giải thuật nhỏ, vẽ từng giải thuật
2.3 Hiện thực
Trình bày code hiện thực của các giải thuật
2.4 Kết quả và thảo luận
Đánh giá kết quả đạt được và thảo luận về kết quả đó
3. Kết luận
Rút ra những điều đã học được cho bản thân qua việc làm Project này
Hướng phát triển trong tương lai

TỰ ĐÁNH GIÁ – ĐỀ TÀI NHÓM

Nội dung	Điểm	Ghi chú
Giới thiệu về bài toán (0.25 đ)		
Mô tả cấu trúc dữ liệu (1.25 đ)		
Sơ đồ giải thuật (1.5 đ)		
Hiện thực (5 đ)		
Kết quả và thảo luận (0.5 đ)		
Điểm nhóm (0.75 đ)		
Các điều rút ra cho bản thân (0.25 đ)		
Báo cáo đúng theo mẫu (0.5 đ)		
Tổng điểm		

TỰ ĐÁNH GIÁ- ĐỀ TÀI 1 THÀNH VIÊN

Nội dung	Điểm	Ghi chú
Giới thiệu về bài toán (0.25 đ)		
Mô tả cấu trúc dữ liệu (1.25 đ)		
Sơ đồ giải thuật (1.5 đ)		
Hiện thực (5.75 đ)		
Kết quả và thảo luận (0.5 đ)		
Các điều rút ra cho bản thân (0.25 đ)		
Báo cáo đúng theo mẫu (0.5 đ)		
Tổng điểm		

Project 1

Xây dựng từ điển sử dụng cấu trúc dữ liệu Cây nhị phân tìm kiếm cân bằng với giải thuật Cây cân bằng đơn giản như trình bày ở dưới. Từ điển có các chức năng:

a- Load từ điển từ file theo cấu trúc:
File có nhiều dòng (tối thiểu là 50 dòng)
Mỗi dòng chứa từ tiếng English và nghĩa tiếng Việt
File example (sử dụng delimiters bất kỳ mà bạn muốn):
measuring | đo lường, tiêu chuẩn
electronics | Điện tử dân dụng
facility | nhà vệ sinh

b- Xây dựng cây cân bằng đơn giản (simple balanced tree)

c- Thêm vào 1 từ

d- Xóa 1 từ

e- Tìm kiếm 1 từ

f- In ra đường đi giữa 2 node bất kỳ trên cây

Giải thuật Simple balanced tree algorithm

B1- Sắp xếp dãy số

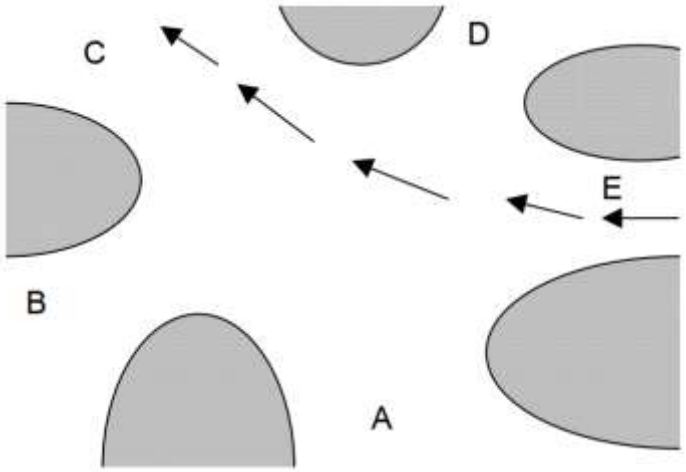
B2- Lấy phần tử ở giữa làm gốc, chia dãy thành 3 phần [Bên trái] Gốc [Bên phải]

B3- Lặp lại bước 2 với dãy ở bên trái và bên phải

B4- Hoàn thành cây nhị phân tìm kiếm cân bằng

Project 2

Bài toán Đèn giao thông
Hãy xây dựng các cột đèn sao cho việc lưu thông không bị giao nhau (số màu đèn là bao nhiêu). Lưu ý tuyến EC là một chiều.



Hãy xây dựng bài toán một cách tổng quát với số lượng điểm và số tuyến đường 1 chiều nhập vào từ 1 file

Với dữ liệu bài toán đã cho được mô tả trong file input.txt như sau:

Dòng đầu số điểm giao thông được đánh thứ tự theo bảng chữ cái

Các dòng tiếp theo là các tuyến đường 1 chiều

Input.txt

5

E C

Project 3

Tại một cửa hàng sách mới, mới nhập về 12 cuốn sách thuộc các loại sau:

- ✓ Truyện cười : A,C,D,G
- ✓ Âm nhạc: B,H,K
- ✓ Lịch sử : E,J,L
- ✓ Khoa học: F, I

Hãy sắp xếp các quyển sách này vào kệ sao cho số kệ sử dụng là ít nhất mà tuân theo các yêu cầu sau:

- Các quyển sách cùng loại không được để chung cùng kệ

- Quyển A không để chung với sách khoa học

- Quyển L không để chung với sách âm nhạc.

Hãy xây dựng bài toán một cách tổng quát với số loại, số sách của mỗi loại và các rule nhập vào từ 1 file (tức là có thể nhập số loại, số sách và rule tùy ý theo yêu cầu)

Với dữ liệu bài toán đã cho được mô tả trong file input.txt như sau:

Dòng đầu số thể loại và số cuốn sách

Các dòng tiếp theo là các thể loại được đánh số từ 1 trở đi và số loại sách của từng thể loại cách nhau bởi dấu ,

Dòng tiếp theo là số luật (Không tính Luật Các quyển sách cùng loại không được để chung cùng kệ- Đây là luật chung cho mọi bài)

Các luật tiếp theo loại nào không trong thể loại nào được nhập theo cú pháp như ví dụ ở dưới.

Input.txt

4 12

1: A,C,D,G

2: B,H,K

3: E,J,L

4: F, I

2

A not in 4

L not in 2

Project 4

Hãy cài đặt giải thuật Decision Tree ID3.

ID3 (J. R. Quinlan 1993) sử dụng phương pháp tham lam tìm kiếm từ trên xuống thông qua không gian của các nhánh có thể không có backtracking. ID3 sử dụng Entropy và Information Gain để xây dựng một cây quyết định.

Project 5

Viết chương trình mô phỏng bài toán người lái đò với giao diện đồ họa. Bài toán phát biểu như sau:

• Tại bến sông nọ có bắp cải, sói và dê muốn bác lái đò chở qua sông. Biết rằng tại một thời điểm thuyền của bác lái đò chỉ chở tối đa được 2 khách. Nếu sói và dê đứng riêng với nhau (không có mặt bác lái đò và bắp cải) thì sói sẽ ăn thịt dê. Nếu dê và bắp cải đứng riêng với nhau (không có mặt bác lái đò và sói) thì dê sẽ ăn bắp cải.

• Ký hiệu bờ sông mà sói, dê, bắp cải và bác lái đò đang đứng là 1, bờ sông bên kia là 2. Hãy viết chương trình giải quyết bài toán trên.

• Chương trình cho phép máy tự chơi hoặc người chơi, máy trợ giúp. Trong trường hợp người chơi, máy trợ giúp, nếu người tắc ở một tình thế nào đó giải được, máy

sẽ hướng dẫn cách đi đến trạng thái đích. Nếu không, đưa ra thông báo là "Bạn không thể hoàn thành công việc".

Project 6

Viết chương trình sắp xếp thời khoá biểu của một học kỳ cho một trường đại học hoặc một trường phổ thông. Các thông tin về thời khoá biểu bao gồm:

- Giáo viên và môn học: giáo viên có thể dạy những môn nào.

- Giáo viên và số tiết học tối thiểu, tối đa trong một tuần.

- Phòng học, ca học: có những phòng học nào; mỗi phòng học trong 1 ngày được chia

làm mấy ca, mỗi ca bao nhiêu tiết

- Trong học kỳ này, mỗi môn học cần được tổ chức bao nhiêu lớp.

Sinh viên tự đưa ra dữ liệu mẫu và thực hiện chạy chương trình trên dữ liệu đó.

Kết quả sắp xếp thời khoá biểu là kết quả gần tối ưu, không nhất thiết là kết quả thoả mãn

tất cả các ràng buộc.

Chương trình sẽ được đánh giá cao nếu có giao diện cho phép:

- Hiển thị kết quả thời khoá biểu.

- Thay đổi thông tin dữ liệu đầu vào.

Project 7

Cây DNA

We turn to the problem of searching for matching sequences in a sequence database. A key element in many bioinformatics problems is the biological sequence. A biological sequence is just a list of characters chosen from some alphabet. Two of the common biological sequences are DNA (composed of the four characters A, C, G, and T) and RNA (composed of the four

characters A, C, G, and U). In this asignment, you will be storing and searching for DNA sequences, defined to be strings on the alphabet A, C, G, and T.

Given a sequence, simply searching a list of sequence strings for one that matches would take a long time for a large collection of sequences. Instead, we will use a tree structure to store sequences in a way that allows for efficient search. Not only can we determine whether a specified sequence is in the database, but we can also find any sequence in the database that matches a search prefix.

DNA Trees:

We will define a new tree data structure to store DNA sequences, that we call a DNA tree. DNA trees store sequences in their leaf nodes. Internal nodes serve only as placeholders to help direct search, they store no data. A leaf node is either empty, or stores a single sequence. Whenever you attempt to insert a new sequence and the insert process reaches a leaf node containing a sequence, that leaf node must split (just as in PR quadtree insertion). Whenever you remove a sequence from a leaf node, if possible that node will merge with its siblings.

The DNA tree is a 5-way branching tree, with a branch for each possible letter. In addition to the letters A, C, G, and T, we must augment the alphabet for DNA sequences to contain $ to indicate the termination of a sequence. This permits us to store sequences that are prefixes of other sequences already stored (without the $ symbol, a prefix would end up at an internal node of the tree, which is forbidden). Thus, the five branches correspond to the five letters of the augmented DNA alphabet: A, C, G, T, and $.

When traversing through the tree structure to perform an operation, the first branch from an internal node corresponds to the letter A, the second branch to the letter C, and so on, with the fifth branch corresponding to $. Thus, all sequences stored that begin with A will be in the first subtree, all sequences stored that begin with C will be in the second subtree, and so on. If there are no sequences stored in the tree, the tree consists of a single empty leaf node. If there is one sequence stored in the tree, the tree consists of a single leaf node containing the sequence.

Input and Output:

The program will be invoked from the command-line as:

P2 <common-file>

The name of the program is P2. Parameter command-file is the name of the input file that holds the commands to be processed by the program.

The input for this project will consist of a series of commands (some with associated parameters, separated by spaces), one command for each line. A blank line may appear anywhere in the command file, and any number of spaces may separate parameters. You need not worry about checking for syntactic errors. That is, only the specified commands will appear in the file, and the specified parameters will always appear. However, you must check for logical errors.

The commands will be read from the command file, and the output from processing the commands will be written to standard output. The program should terminate after reading the EOF mark. The commands are as follows:

insert *sequence*

Insert sequence into the DNA tree. Print a message indicating if the insertion was successful, and if so, indicate the level of the leaf node inserted. It is an error to insert a duplicate sequence. Such an error should be reported in the output, and no changes to the tree structure should take place.

remove *sequence*

Remove sequence from the DNA tree if it exists. Print a suitable message if sequence is not in the tree. print Print out the DNA tree, including both the node structure and the sequences it contains. You should perform a preorder traversal of the tree, and print each node on a separate line in the order that it is visited by the traversal. If the node is internal, just print the letter I. If the node is an empty leaf node, just print the letter E. If the node contains a sequence, print the sequence. All nodes should be printed so that the line is indented by 2 spaces for each level in the tree. That is, the root node is printed with no indentation, immediate children of the root are indented 2 spaces, grandchildren of the root are indented 4 spaces, and so on.

print *lengths*

Output is identical to that of the print command, except that the length of the sequence is printed after the sequence for all sequences stored in the database.

print *stats*

Output is identical to that of the print command, except that the letter breakdown (by percentage) of the sequence is printed after the sequence for all sequences stored in the database. That is, for each letter A, C, G, and T, the percentage of A's in that sequence is printed, the percentage of C's, and so on. Percentages should be printed with two decimal places. For example, AAAAG should show A's as 80.00% and G's as 20.00%.

search *sequence*

Descriptor Find all sequences that match sequenceDescriptor. The sequenceDescriptor can come in two forms. The first form is simply as a sequence containing letters from the alphabet A, C, G, and T. If this form is given, then print all sequences stored in the tree for which sequenceDescriptor is a prefix (including exact matches). The second form is a sequence from the letters A, C, G, and T, followed by a $ symbol. If this form is given, then only an exact match of the sequence (without the $ symbol) is to be printed. Print the number of nodes visited in the tree during the search.

Implementation:

You must use class inheritance to design your DNA Tree nodes. You must have a DNA Tree node base class, with subclasses for the internal nodes and the leaf nodes.

Note that many leaf nodes of the DNA tree will contain no data. Storing many distinct "empty" leaf nodes is quite wasteful. One design option is to store a NULL pointer to an empty leaf node in its parent. However, this requires the parent node to understand this convention, and explicitly check the value of its child pointers before proceeding, with special action taken if the pointer is NULL. A better design is to use a "flyweight" object. A flyweight is a single empty leaf node that is created one time at the beginning of the program, and pointed to whenever an empty child node is needed. Your project should use a flyweight design to implement empty leaf nodes.

Internal nodes may not store data of any type (other than the pointers to children). No operation should look at more nodes than necessary (especially the search operation).

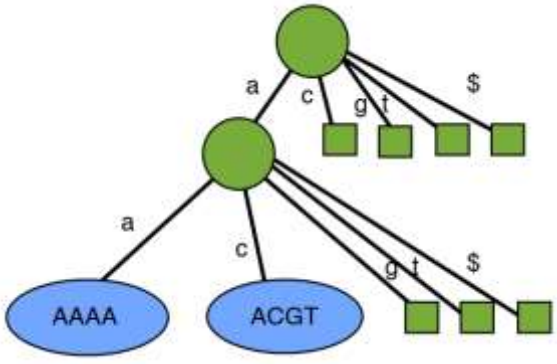
All DNA tree operations must be implemented recursively.

The DNA sequences stored in the leaf nodes may be implemented using a linked list, or you may store them in a character array of the appropriate size.

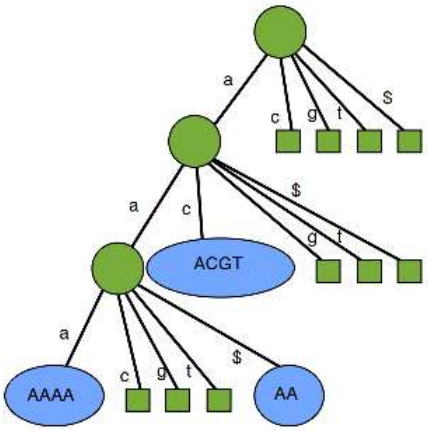
Example:

Insert ACGT

Insert AAAA



Insert AA



Project 8

FP Growth Algorithms

Cài đặt giải thuật FP Growth và ứng dụng giải thuật trong giải quyết bài toán tìm tập luật phổ biến của tập hóa đơn bán hàng

So sánh tập luật tìm được với Công cụ Weka

Project 9

Xây dựng từ điển sử dụng Hash với Chaining (Collision Resolution – Giải quyết đụng độ bằng phương pháp Chaining method) và division (Hash Function – Xây dựng hàm Hash Function bằng phương pháp Division). Từ điển có các chức năng:

a- Load từ điển từ file theo cấu trúc:
File có nhiều dòng (tối thiểu là 50 dòng)
Mỗi dòng chứa từ tiếng English và nghĩa tiếng Việt
File example (sử dụng delimiters bất kỳ mà bạn muốn):
measuring | đo lường, tiêu chuẩn
electronics | Điện tử dân dụng
facility | nhà vệ sinh

b- Xây dựng Hash theo yêu cầu trên
c- Thêm từ mới
d- Xóa 1 từ trong từ điển
e- Tra 1 từ
f- In ra thông tin về Hash:

- Kích cỡ của Hash Table
- Phần tử dài nhất trong Hash Table

Project 10

Viết chương trình mô phỏng bài toán con khỉ - nải chuối với giao diện đồ họa. Xuất phát từ 1 trạng thái bất kỳ, chương trình có thể đưa ra được cách giải quyết để con khỉ lấy được nải chuối. Chương trình cho phép máy tự chơi hoặc người chơi, máy trợ giúp.

Con khỉ và nải chuối, khỉ bị nhốt trong lồng, nải chuối bị treo trên trần nhà, con khỉ ko thể với tới nải chuối, nhưng có thể dùng 1 cái ghế và 1 cây gậy để khều nải chuối.

Biết con khỉ có thể:

1/ Di chuyển cái ghế

2/ Trèo lên 1 vật

3/ Cầm 1 vật

4/ Khều nải chuối

Tìm ra hành động tốt nhất để con khỉ lấy được nải chuối.

Project 11

Hãy cài đặt giải thuật cho trò chơi cryptarithmetic puzzle

A cryptarithmetic puzzle is a mathematical exercise where the digits of some numbers are represented by letters (or symbols). Each letter represents a unique digit. The goal is to find the digits such that a given mathematical equation is verified:

```
   CP
+  IS
+  FUN
--------
=  TRUE
```

One assignment of letters to digits yields the following equation:

```
   23
+  74
+  968
--------
=  1065
```

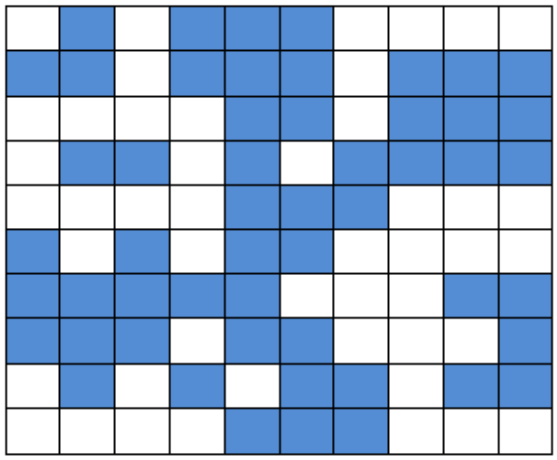
Project 12

Cài đặt giải thuật Apriori và ứng dụng giải thuật trong giải quyết bài toán tìm tập luật phổ biến của tập hóa đơn bán hàng. So sánh tập luật tìm được với Công cụ Weka

Project 13

Tìm số đảo của dữ liệu ở dưới theo 2 giải thuật:

- BFS (Breadth First Search)
- UCS (Uniform Cost Search)



1		2				3	3	3	3
		2				3			
2	2	2	2			3			
2			2		3				
2	2	2	2				5	5	5
	2		2			5	5	5	5
					5	5	5		
			4			5	5	5	
4		4		4			5		
4	4	4	4				5	5	5

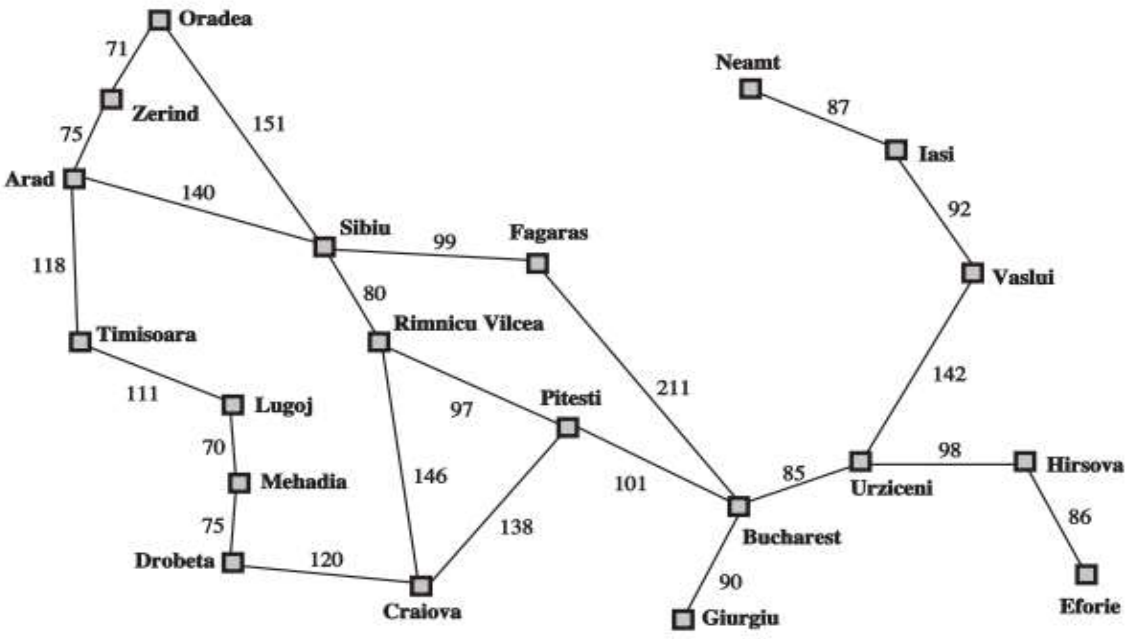
Input: Green color: sea , land: white color, Output: 5 islands

Project 14

Giải quyết bài toán du lịch Romania với dữ liệu cho ở dưới bằng 2 giải thuật:

- A* và Breadth First Traversal

Example: from **Arad to Bucharest**



Straight-line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Project 15

Hãy cài đặt giải thuật cho trò chơi Game of Life với số ô là 20x20 và tự động sinh ra các lưới ban đầu theo một giải thuật do bạn đề xuất.

Các tế bào chết được biểu thị bằng dấu gạch ngang (‘-‘), và các tế bào sống được biểu thị bằng dấu hoa thị (‘*’).

Trò chơi được thể hiện là một lưới ô vuông trực giao hai chiều, vô hạn, mỗi ô vuông ở một trong hai trạng thái có thể có, sống hoặc chết, (hoặc có dân cư và không có dân cư, tương ứng). Mỗi ô tương tác với tám ô lân cận của nó, đó là các ô nằm cạnh theo chiều ngang, chiều dọc hoặc đường chéo. Tại mỗi bước trong thời gian, các quá trình chuyển đổi sau đây xảy ra:

1. Bất kỳ ô sống nào có ít hơn hai người hàng xóm còn sống sẽ chết

2. Bất kỳ ô sống nào có hai hoặc ba người hàng xóm còn sống sẽ truyền sang thế hệ tiếp theo.

3. Bất kỳ ô sống nào có nhiều hơn ba người hàng xóm còn sống sẽ chết

4. Bất kỳ ô chết nào có đúng ba ô sống chung sẽ trở thành ô sống, như thể bằng cách sinh sản.

Những quy tắc này, so sánh với cuộc sống thực, có thể được cô đọng thành những điều sau:

1. Bất kỳ ô sống nào có hai hoặc ba người hàng xóm sống sót đều sống sót.

2. Bất kỳ ô chết nào có ba người hàng xóm sống trở thành một ô sống.