**Đề số 11:**

Câu 1:Nêu ý tưởng Trộn tự nhiên và minh họa :1,2,9,8,7,6,5,3,4

1. Sắp xếp trộn tự nhiên

Ý tưởng :

· Quét qua tập dữ liệu một lần từ đầu đến cuối.

· Tách dữ liệu thành các chuỗi con (run) liên tiếp đã được sắp xếp (tăng dần).

· Lợi dụng các chuỗi con tự nhiên có sẵn mà không cần phải chia nhỏ từng phần như trong sắp xếp trộn thông thường.

· Dùng thuật toán trộn hai dãy (merge two sorted arrays) để trộn hai chuỗi con tăng dần liên tiếp thành một chuỗi con lớn hơn, theo thứ tự tăng dần.

· Tiếp tục quá trình trộn cho đến khi toàn bộ dữ liệu được hợp thành một chuỗi duy nhất đã sắp xếp.

Minh họa

Dữ liệu đầu vào: **1 2 9 8 7 6 5 3 4**.

Dãy đầu vào là 1 2 9 8 7 6 5 3 4. Chúng ta sẽ áp dụng thuật toán **Sắp xếp ngoài trộn tự nhiên** như sau:

### ****Dữ liệu ban đầu (****f0****):****

1 2 9 8 7 6 5 3 4

### ****Bước 1: Chia các run tự nhiên từ**** f0 ****vào**** f1 ****và**** f2

Duyệt qua dãy f0 để xác định các đoạn tăng dần (run tự nhiên):

* **Run 1**: 1 2 9 (tăng dần)
* **Run 2**: 8
* **Run 3**: 7
* **Run 4**: 6
* **Run 5**: 5
* **Run 6**: 3 4 (tăng dần)

Chia luân phiên các run vào f1 và f2:

**F1 : 1 2 9 , 7 , 5**

**F2: 8, 6 , 3 4**

### ****Bước 2: Trộn run từ**** f1 ****và**** f2 ****vào lại**** f0

**F0**1 2 8 9 6 7 3 4 5

### ****Bước 3: Chia lại**** f0 ****thành các run tự nhiên vào**** f1 ****và**** f2

Chia các run tự nhiên trong f0:

* **Run 1:** 1 2 8 9
* **Run 2:** 6 7
* **Run 3:** 3 4 5

Chia luân phiên vào f1 và f2:

F1: 1 2 8 9 , 3 4 5

F2: 6 7

### ****Bước 4: Trộn run từ**** f1 ****và**** f2 ****vào lại**** f0

**F0:**1 2 6 7 8 9 3 4 5

### ****Bước 5: Chia lại**** f0 ****thành các run tự nhiên vào**** f1 ****và**** f2

Chia các run tự nhiên trong f0:

* **Run 1:** 1 2 6 7 8 9
* **Run 2:** 3 4 5

Chia luân phiên vào f1 và f2:

**F1: 1 2 6 7 8 9**

**F2: 3 4 5**

### ****Bước 6: Trộn run từ**** f1 ****và**** f2 ****vào lại**** f0

**F0:**1 2 3 4 5 6 7 8 9

### ****Bước 7: Dừng****

* Dừng vì f0 chỉ còn một run duy nhất.

### ****Kết quả cuối cùng:****

Dãy đã sắp xếp: 1 2 3 4 5 6 7 8 9.

Câu2:Cam hỏng

\* Bài quả cam

#include <iostream>

#include <vector>

#include <queue>

using namespace std;

// Hàm xác định thời gian tối thiểu để làm hỏng toàn bộ cam

int rotOranges(vector<vector<int>>& matrix) {

int rows = matrix.size();

int cols = matrix[0].size();

queue<pair<int, int>> q; // Hàng đợi để BFS

int fresh\_count = 0; // Đếm số lượng quả cam tươi

// Khởi tạo hàng đợi và đếm số quả cam tươi

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

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

if (matrix[i][j] == 2) {

q.push({i, j}); // Đưa quả cam bị hỏng vào hàng đợi

} else if (matrix[i][j] == 1) {

fresh\_count++; // Đếm quả cam tươi

}

}

}

// Nếu không có quả cam tươi, trả về 0

if (fresh\_count == 0) return 0;

// Các hướng di chuyển (trái, phải, lên, xuống)

vector<pair<int, int>> directions = {{-1, 0}, {1, 0}, {0, -1}, {0, 1}};

int minutes = 0; // Thời gian tối thiểu

// BFS để lan tỏa cam bị hỏng

while (!q.empty()) {

int size = q.size();

bool rotten = false; // Kiểm tra xem có cam tươi nào bị hỏng trong vòng này không

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

auto [x, y] = q.front();

q.pop();

for (auto dir : directions) {

int nx = x + dir.first;

int ny = y + dir.second;

// Kiểm tra nếu vị trí hợp lệ và chứa cam tươi

if (nx >= 0 && nx < rows && ny >= 0 && ny < cols && matrix[nx][ny] == 1) {

matrix[nx][ny] = 2; // Cam tươi trở thành cam bị hỏng

q.push({nx, ny}); // Thêm vào hàng đợi

fresh\_count--; // Giảm số lượng cam tươi

rotten = true;

}

}

}

// Nếu có quả cam mới bị hỏng, tăng thời gian

if (rotten) minutes++;

}

// Nếu còn quả cam tươi, trả về -1. Ngược lại trả về thời gian.

return fresh\_count > 0 ? -1 : minutes;

}

int main() {

// Test case 1

vector<vector<int>> matrix1 = {

{2, 1, 0, 2, 1},

{1, 0, 1, 2, 1},

{1, 0, 0, 2, 1}

};

// Test case 2

vector<vector<int>> matrix2 = {

{2, 1, 0, 2, 1},

{1, 0, 1, 2, 1},

{1, 0, 0, 2, 0}

};

cout << "Thời gian tối thiểu để làm hỏng toàn bộ cam (matrix1): " << rotOranges(matrix1) << endl;

cout << "Thời gian tối thiểu để làm hỏng toàn bộ cam (matrix2): " << rotOranges(matrix2) << endl;

return 0;

}

Minh họa

Ma trận ban đầu:

matrix1 =

[ {2, 1, 0, 2, 1}, {1, 0, 1, 2, 1}, {1, 0, 0, 2, 1}]

**Ký hiệu:**

* 2: Cam bị hỏng.
* 1: Cam tươi.
* 0: Ô trống.

Bước 1: Khởi tạo

**Hàng đợi (queue):** Chứa tất cả vị trí cam bị hỏng (2)

queue = [(0, 0), (0, 3), (1, 3), (2, 3)]

**Số cam tươi (**fresh\_count**):** 6 (đếm tất cả các ô chứa 1).

Bước 2: BFS - Phút 1

Lấy cam bị hỏng trong hàng đợi (lan tỏa):

queue = [(0, 0), (0, 3), (1, 3), (2, 3)]

**Lan tỏa cam bị hỏng đến các ô lân cận (trái, phải, trên, dưới):**

* Cam tươi ở vị trí (0, 1) bị hỏng.
* Cam tươi ở vị trí (1, 0) bị hỏng.
* Cam tươi ở vị trí (1, 4) bị hỏng.
* Cam tươi ở vị trí (2, 4) bị hỏng.

Cập nhật ma trận:

[ {2, 2, 0, 2, 1}, {2, 0, 1, 2, 2}, {1, 0, 0, 2, 2}]

Hàng đợi mới:

queue = [(0, 1), (1, 0), (1, 4), (2, 4)]

**Cam tươi còn lại (**fresh\_count**):** 2.

Bước 3: BFS - Phút 2

Lấy cam bị hỏng trong hàng đợi (lan tỏa):

queue = [(0, 1), (1, 0), (1, 4), (2, 4)]

· **Lan tỏa cam bị hỏng đến các ô lân cận:**

* · Cam tươi ở vị trí (2, 0) bị hỏng.
* Cam tươi ở vị trí (0, 4) bị hỏng.

· **Cập nhật ma trận:**

· [ {2, 2, 0, 2, 2}, {2, 0, 1, 2, 2}, {2, 0, 0, 2, 2}]

Hàng đợi mới: queue = [(2, 0), (0, 4)]

**Cam tươi còn lại (**fresh\_count**):** 0.

### ****ước 4: Kết thúc****

* **Hàng đợi rỗng (**queue.empty()**): queue = []**
* Không còn cam tươi (fresh\_count == 0).

**Ma trận sau khi xử lý:**

Sao chép mã

[ {2, 2, 0, 2, 2}, {2, 0, 1, 2, 2}, {2, 0, 0, 2, 2}]

**Thời gian tối thiểu:** 2 phút.

Câu 3: Viết chương trình chèn dãy giá trị {4371,1323,6173,4199,4344,9679,1989} vào bảng băm thăm dò bậc 2 với hàm băm hash(x)=x%10 và vẽ hình minh họa chèn từng giá trị trên vào bảng băm thăm dò bậc 2

Viết chương trình chèn dãy giá trị {4371, 1323, 6173, 4199, 4344, 9679, 1989} vào bảng băm thăm dò bậc hai với hàm băm hash(x) = x % 10 và vẽ hình minh họa quá trình chèn từng giá trị trên vào bảng băm thăm dò bậc hai.(python)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 4371 | 1323 | 6173 | 4199 | 4344 | 9679 | 1989 |
|  |  |  |  |  |  |  |  |  |
| 0 |  |  |  |  |  |  | 9679 | 9679 |
| 1 |  | 4371 | 4371 | 4371 | 4371 | 4371 | 4371 | 4371 |
| 2 |  |  |  |  |  |  |  |  |
| 3 |  |  | 1323 | 1323 | 1323 | 1323 | 1323 | 1323 |
| 4 |  |  |  | 6173 | 6173 | 6173 | 6173 | 6173 |
| 5 |  |  |  |  |  | 4344 | 4344 | 4344 |
| 6 |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  | 1989 |
| 9 |  |  |  |  | 4199 | 4199 | 4199 | 4199 |

hash\_table\_size = 10 # kích cỡ bảng băm  
hash\_table = [-1] \* hash\_table\_size # bảng băm  
  
  
# hàm băm  
def hash\_function(x):  
 return x % 10  
  
  
# thêm số mới  
def insert(x):  
 hash\_value = hash\_function(x)  
 # thêm vào bảng băm bằng phương pháp thăm dò bậc 2  
 for i in range(len(hash\_table)):  
 if hash\_table[(hash\_value + i \* i) % len(hash\_table)] == -1:  
 hash\_table[(hash\_value + i \* i) % len(hash\_table)] = x  
 break  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 lst = [4371, 1323, 6173, 4199, 4344, 9679, 1989]  
 for i in lst:  
 insert(i)  
  
 # in ra toàn bộ số  
 for i in range(0, len(hash\_table)):  
 if hash\_table[i] == -1:  
 print(str(i) + ". ")  
 else:  
 print(str(i) + ". " + str(hash\_table[i]))