# III. GIẢI THUẬT BÀI TOÁN

**Đề bài:**

Hãy tìm ra cách vận chuyển giúp người nông dân có thể mang được cả sói, dê và bắp cải qua sông.

Một nông dân muốn vượt qua sông với một con sói, một con dê và một cái bắp cải.

Người nông dân chỉ có một chiếc thuyền nhỏ và nó chỉ có thể chở theo anh ta cùng một trong ba thứ: con sói, con dê hoặc cái bắp cải. Vấn đề nằm ở chỗ nếu để sói và dê ở lại bờ, sói sẽ ăn thịt dê. Nếu để dê với bắp cải ở lại bờ, dê sẽ ăn bắp cải.



**Mục Tiêu:**

Chuyển toàn bộ các đối tượng từ một bờ của sông sang bờ kia theo một số quy tắc nhất định.

**Mô Tả:**

1. **Bờ Sông**

Sông được mô tả bằng một dòng chữ: "WFSC"

Các ký tự có thể bao gồm:

- W: Con sói.

- F: Người nông dân.

- S: Con cừu.

- C: Bắp cải.

**2. Quy Tắc Di Chuyển:**

- Chỉ có thể di chuyển một hoặc hai đối tượng (người và/hoặc động vật) mỗi lần.

- Chỉ có người nông dân (F) có thể lái thuyền.

1. **Ràng Buộc:**

- Không thể để con cừu (S) ở một bờ khi người nông dân không ở đó.

- Không thể để quả bắp cải (C) ở một bờ khi không có người chăn cừu.

**Mục Tiêu:**

Chuyển toàn bộ các đối tượng từ bờ ban đầu sang bờ đích sao cho tất cả đều tuân theo các quy tắc đã đề cập.

**Yêu Cầu:**

Xây dựng một thuật toán để tìm tất cả các bước di chuyển cần thiết để đạt được mục tiêu.

**Xây dụng giải thuật Breadth First Search**

*// Giải thuật Breadth First Search:*

public void startBreadthFirstSearch() {

TreeSet<String> left = new TreeSet<>(Set.of("W", "S", "C", "F"));

root = new Node(new State("left", left, new TreeSet<>()));

queue.add(root);

while (!queue.isEmpty()) {

Node node = queue.remove(0);

System.out.println("\t\* Tiến trình Level " + node.level + ": " + node.data);

for (String m : moves) {

State s = node.data.transits(m);

if (s != null && s.isAllow()) {

Node child = new Node(s);

child.parent = node;x

child.level = node.level + 1;

child.move = m + " moves " + child.data.bank();

if (!child.isAncestor()) {

node.adjList.add(child);

if (!child.data.isSolution()) {

queue.add(child);

System.out.println("\t+ Thêm trạng thái: " + child.data);

} else {

solutions.add(child);

System.out.println("\t-> Tìm giải pháp: " + child.data);

}

}

}

}

}

}

*// Hàm kiểm tra xem trạng thái di chuyển cho thoả mãn không*

public State transits(String move) {

String nbank;

TreeSet<String> nleft = new TreeSet<>();

TreeSet<String> nright = new TreeSet<>();

if (bank.equalsIgnoreCase("left"))

nbank = "right";

else

nbank = "left";

copylist(right, nright);

copylist(left, nleft);

for (int i = 0; i < move.length(); i++) {

String item = move.substring(i, i + 1);

if (bank.equalsIgnoreCase("left")) {

if (nleft.remove(item))

nright.add(item);

else

return null; // trả về null nếu di chuyển chứa

} else {

if (nright.remove(item))

nleft.add(item);

else

return null; // trả về null nếu di chuyển chứa

}

}

return new State(nbank, nleft, nright);

}

*// Hàm kiểm tra xem thứ tự di chuyển có thoả mãn yêu cầu không:*

private boolean checkAllowBank(TreeSet<String> b) {

if (b.contains("W") && b.contains("S") && !b.contains("F"))

return false;

return !b.contains("S") || !b.contains("C") || b.contains("F");

}

public boolean isAllow() {

return checkAllowBank(left) && checkAllowBank(right);

}