CHI TIẾT BÀI TOÁN 8 QUÂN HẬU

I. Phân tích yêu cầu bài toán

Hướng được chọn để giải quyết bài toán trên là dùng giải thuật quay lui (Backtracking) để bảo đảm tìm đáp án chính xác và đầy đủ.

Như đã đề cập trước đó, mục tiêu bài toán là chương trình sẽ tự động tìm lời giải hợp lệ cho bài toán 8 (nhiều hơn hoặc ít hơn) quân hậu.

Bài toán tám quân hậu là bài toán đặt tám quân hậu trên bàn cờ vua kích thước 8×8 sao cho không có quân hậu nào có thể "ăn" được quân hậu khác, hay nói khác đi không quân hậu nào có để di chuyển theo quy tắc cờ vua. Màu của các quân hậu không có ý nghĩa trong bài toán này.

Như vậy, lời giải của bài toán là một cách xếp tám quân hậu trên bàn cờ sao cho không có hai quân nào đứng trên cùng hàng, hoặc cùng cột hoặc cùng đường chéo. Bài toán tám quân hậu có thể tổng quát hóa thành bài toán đặt n quân hậu trên bàn cờ n×n.

Mỗi quân hậu được đặt vào bàn cờ sẽ được kiểm tra tính hợp lệ (xem có hợp lệ để đứng ở vị trí đó hay không), nếu hợp lệ thì sẽ đặt quân hậu tiếp theo, còn nếu không hợp lệ sẽ đặt quân hậu hiện tại ở vị trí khác.

Để áp dụng phương pháp trên, bài toán sẽ sử dụng một giải thuật rất phổ biến trong lập trình, gọi là giải thuật quay lui (Backtracking). Hiện tại có nhiều cách để giải quyết bài toán này, tuy nhiên trong phạm vi của niên luận này, em sẽ chỉ áp dụng một kỹ thuật để giải quyết bài toán, và ứng dụng nó vào phần mềm của mình.

II.Thiết kế giải thuật

1.Mô tả sơ lược về kỹ thuật quay lui:

Quay lui là một kĩ thuật thiết kế giải thuật dựa trên đệ quy. Ý tưởng của quay lui là tìm lời giải từng bước, mỗi bước chọn một trong số các lựa chọn khả dĩ và đệ quy. Người đầu tiên đề ra thuật ngữ này (backtrack) là nhà toán học người Mỹ D. H. Lehmer vào những năm 1950.

Các bài toán thỏa mãn ràng buộc là các bài toán có một lời giải đầy đủ, trong đó thứ tự của các phần tử không quan trọng. Các bài toán này bao gồm một tập các biến mà mỗi biến cần được gán một giá trị tùy theo các ràng buộc cụ thể của bài toán. Việc quay lui là để thử tất cả các tổ hợp để tìm được một lời giải. Thế mạnh của phương pháp này là nhiều cài đặt tránh được việc phải thử nhiều tổ hợp chưa hoàn chỉnh, và nhờ đó giảm thời gian chạy.

Ví dụ về thuật toán quay lui

Bài toán đặt ra là: Ta muốn tìm tất cả các cách có thể để xếp 2 nam và 1 nữ trên 3 chiếc ghế dài.

Ràng buộc: Bạn nữ không thể ngồi trên ghế giữa.

Bài giải: Có tổng là 3! = 6 khả năng. Chúng ta sẽ thử tất cả các khả năng và có được các giải pháp khả thi. Chúng ta thử kỹ thuật đệ quy cho tất cả các khả năng.

Tất cả các khả năng được minh họa trong hình bên dưới bao gồm như sau:

Table 1 Minh họa bài toán

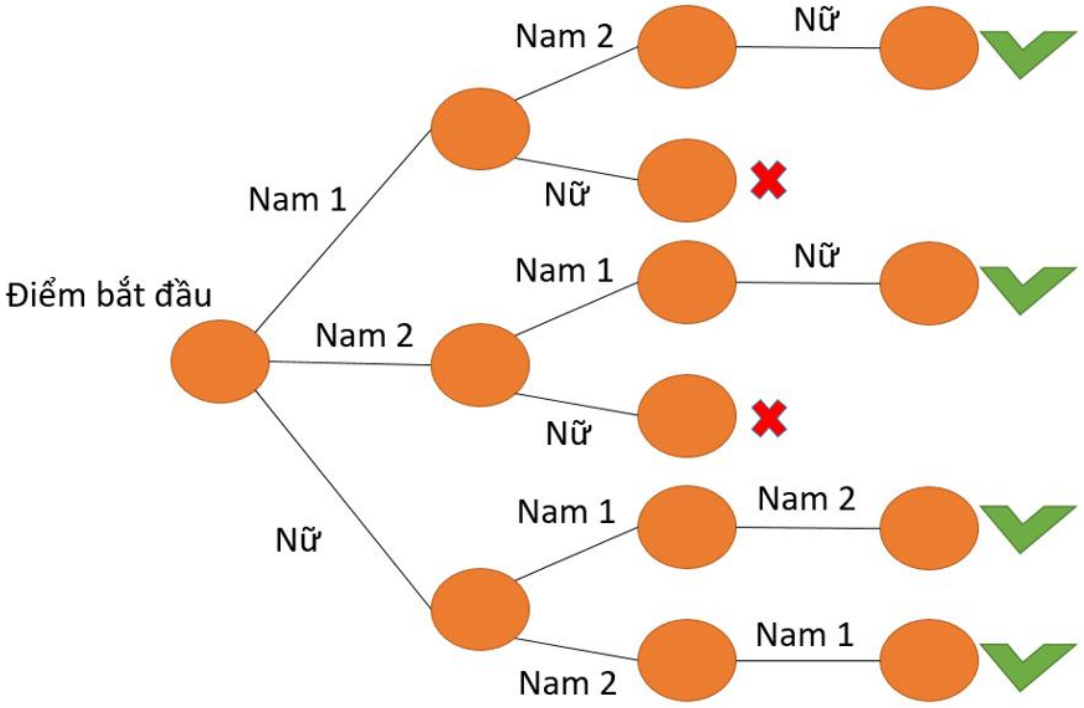

Table 2 Lời giải được sắp xếp theo cây

Vì với ràng buộc là bạn nữ không được ngồi ở giữa, do đó, các giải pháp để nữ có thể ngồi ở giữa sẽ không phải là lời giải cho bài toán này. Do vậy, ta sẽ loại bỏ đi các giải pháp mà trong đó nữ ngồi ở giữa, trong hình trên, ta sẽ đánh dấu X màu đỏ cho tất cả các giải pháp như vậy (chỉ có 2 giải pháp là nữ ngồi ở giữa), và thực hiện đánh dấu tích xanh cho tất cả các giải pháp còn lại cho bài toán.

Ý tưởng chính của bài toán đó là là chúng ta có thể xây dựng một giải pháp theo từng bước bằng cách sử dụng kỹ thuật đệ quy. Nếu trong suốt quá trình thực hiện, chúng ta nhận thấy đó không phải là một giải pháp hợp lệ, thì chúng ta ngừng việc tính toán cho giải pháp đó và chúng ta sẽ quay lại bước trước đó (quay lui lại). Trong trường hợp bài toán về sắp xếp chỗ ngồi như trên, khi chúng ta tính toán rằng cho phép nữ ngồi ở giữa (điều kiện không được phép), chúng ta buộc phải quay lui lại (không tính tiếp nữa), nhưng cũng có một số trường hợp khác mà chúng ta có thể nhận ra rằng chúng ta đang hướng đến một giải pháp không hợp lệ (hoặc không tốt) trước khi đạt được nó.

Nhận xét :

+ **Ưu điểm**: Việc quay lui là thử tất cả các tổ hợp để tìm được một lời giải. Thế mạnh của phương pháp này là nhiều cài đặt tránh được việc phải thử nhiều trường hợp chưa hoàn chỉnh, nhờ đó giảm thời gian chạy.

+ Nhược điểm: Trong trường hợp xấu nhất độ phức tạp của quay lui vẫn là cấp số mũ , vì nó mắc phải các nhược điểm sau:

Rơi vào tình trạng "thrashing": qúa trình tìm kiếm cứ gặp phải bế tắc với cùng một nguyên nhân.

Thực hiện các công việc dư thừa: Mỗi lần chúng ta quay lui, chúng ta cần phải đánh giá lại lời giải trong khi đôi lúc điều đó không cần thiết.
Không sớm phát hiện được các khả năng bị bế tắc trong tương lai. Quay lui chuẩn, không có cơ chế nhìn về tương lai để nhận biết đc nhánh tìm kiếm sẽ đi vào bế tắc.

Lưu đồ tổng quát của thuật toán quay lui :

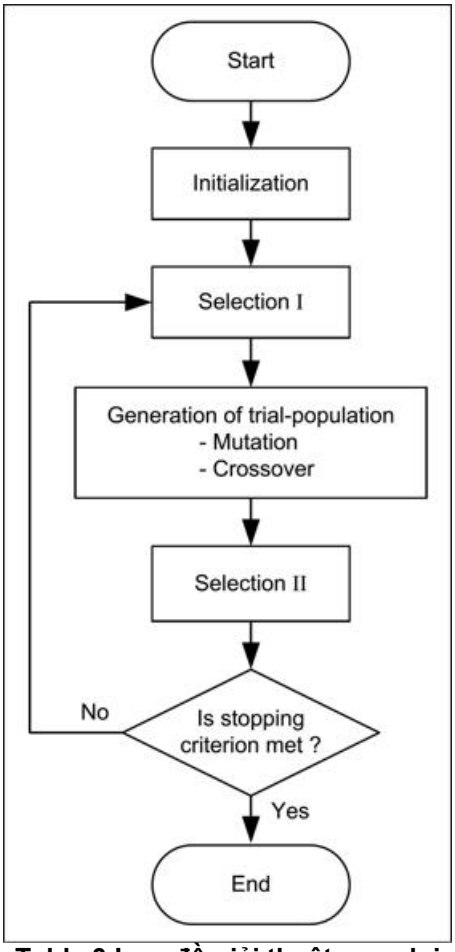

Table 3 Lưu đồ giải thuật quay lui

III. Mô tả chi tiết bài toán 8 quân hậu

Một quân hầu trên bàn cờ có thể di chuyển theo hàng ngang, cột dọc và 2 đường chéo.

Bài toán được đặt ra như sau: Cho một bàn cờ có kích thước NxN ($N \geq 1$), Bạn có thể đặt đúng N quân hậu lên bàn cờ (mỗi ô chỉ chứa tối đa một quân hậu), hãy đưa ra cách đặt N quân hậu sao cho không có 2 quân hậu nào ăn được nhau, nói cách khác là trên mỗi hàng, một cột, mỗi đường chéo của bàn cờ chỉ chứa tối đa một quân hậu.

Nhận xét bài toán: Chúng ta cần đặt N quân hậu sau cho trên mỗi hàng, một cột, mỗi đường chéo của bàn cờ chỉ chứa tối đa một quân hậu, như vậy trên mỗi hàng sẽ có đúng 1 quân hậu được đặt, ta sẽ đánh số quân hậu đặt trên hàng i là quân thậu thứ i.

Như vậy chúng ta có thể làm như sau:
Cách kiểm tra một ô vuông có nằm trong tầm ngắm của các quân hậu trước đó hay không:
- Sử dụng mảng boolean c để đánh dấu các cột của bàn cờ (c[i] = true nếu trên cột i chưa đặt quân hậu nào)

- Sử dụng màng bool c1 để đánh dấu các đường chéo song song với đường chéo chính của bạn cờ (c[i - j + N -1] = true, nghĩa là đường chéo đi qua ô(i, j) và song song với đường chéo chính chưa được đặt quân hậu nào.
- Sử dụng màng bool c2 để đánh dấu các đường chéo song song với đường chéo phụ của bạn cờ (c[i + j - 2] = true, nghĩa là đường chéo đi qua ô(i, j) và song song với đường chéo phụ chưa được đặt quân hậu nào.

Sau tìm xong vị trí của quân hậu thứ N thì ta lưu output đó lại.
Hàm đệ quy được viết như sau:

```
bool check(int i, int j) {
	if (c[j] == false || c1[i - j + N - 1] == false || c2[i + j - 2] ==    false)
		return false;
	return true;
}

void NQueen(int i) {
	for (int j = 1; j <= N; j++)
		if (check(i, j)) {
			x[i] = j;
			c[j] = c1[i - j + N - 1] = c2[i + j - 2] = false;
			if (i == N)
				a.push_back(x);
			else
				NQueen(i + 1);
			c[j] = c1[i - j + N - 1] = c2[i + j - 2] = true;
		}
}
```