# Squeen

## Task 1: Steepest-ascend Hill Climbing Search

**Ý tưởng thuật toán**

**-** Bắt đầu với một bàn cờ ngẫu nhiên.

**-** Mỗi bước, thử di chuyển một quân hậu trong cột của nó đến mọi hàng có thể.

**-** Tính số xung đột (conflicts) của từng cấu hình mới.

**-** Chọn cấu hình tốt nhất (ít xung đột nhất) để chuyển sang.

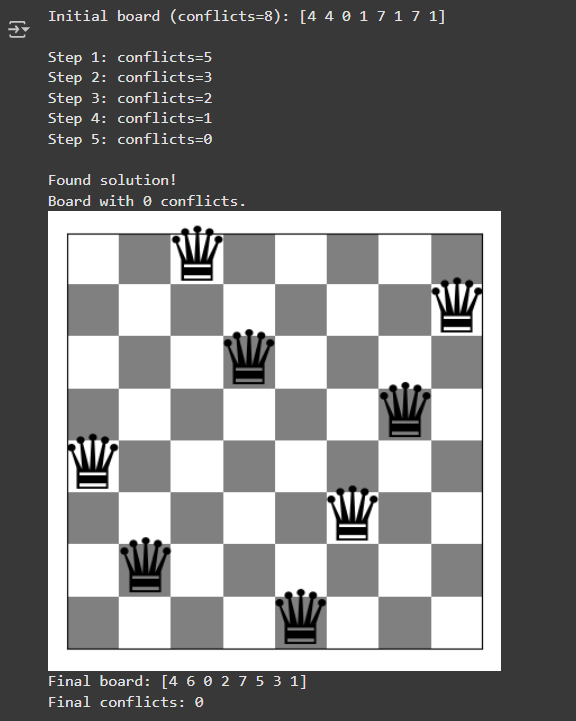
**-** Lặp lại cho đến khi:

**+** Không tìm thấy trạng thái tốt hơn (→ đạt cực tiểu cục bộ), hoặc

**+** Tìm được nghiệm (conflicts == 0), hoặc

**+** Hết số bước max\_steps.

**Code và Mô tả**



**Initial board (conflicts=8): [4 4 0 1 7 1 7 1]**

- Bàn cờ ban đầu được tạo ngẫu nhiên, có 8 quân hậu (cho bài toán 8-Queens).

- Mảng [4 4 0 1 7 1 7 1] nghĩa là:

+ Cột 0 → Hậu ở hàng 4

+ Cột 1 → Hậu ở hàng 4

+ Cột 2 → Hậu ở hàng 0

+ Cột 3 → Hậu ở hàng 1

+ Cột 4 → Hậu ở hàng 7

+ Cột 5 → Hậu ở hàng 1

+ Cột 6 → Hậu ở hàng 7

+ Cột 7 → Hậu ở hàng 1

Nhiều quân hậu đang trùng hàng và trùng chéo, nên ban đầu có 8 xung đột.\

**Step 1: conflicts=5**

**Step 2: conflicts=3**

**Step 3: conflicts=2**

**Step 4: conflicts=1**

**Step 5: conflicts=0**

- Ở mỗi bước, thuật toán thử di chuyển từng hậu trong cột của nó đến tất cả các hàng khác để tìm cấu hình có ít xung đột nhất

- Mỗi bước giảm dần số lượng conflicts:

+ Từ 8 → 5 → 3 → 2 → 1 → 0

- Khi đạt 0 xung đột, nghĩa là không còn quân hậu nào ăn nhau.

**Found solution!**

**Board with 0 conflicts.**

**Final board: [4 6 0 2 7 5 3 1]**

**Final conflicts: 0**

- Thuật toán tìm được một lời giải hợp lệ cho bài toán 8-Queens.

- Mảng [4 6 0 2 7 5 3 1] nghĩa là:

+ Cột 0: Hậu ở hàng 4

+ Cột 1: Hậu ở hàng 6

+ Cột 2: Hậu ở hàng 0

+ Cột 3: Hậu ở hàng 2

+ Cột 4: Hậu ở hàng 7

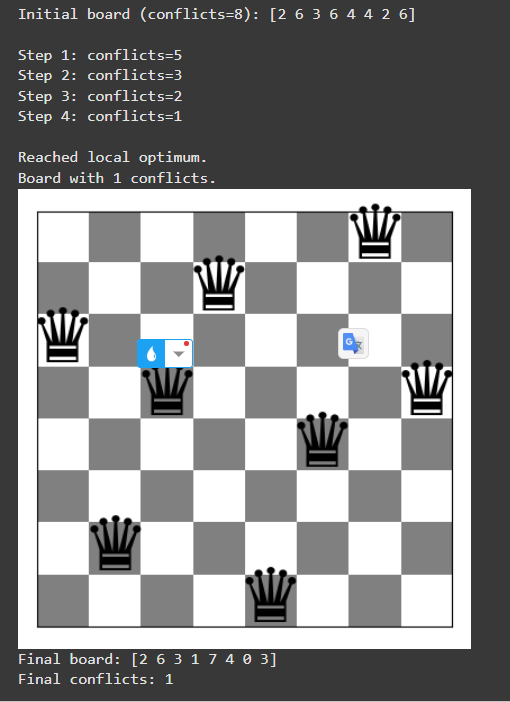
+ Cột 5: Hậu ở hàng 5

+ Cột 6: Hậu ở hàng 3

+ Cột 7: Hậu ở hàng 1

Không có 2 hậu nào cùng hàng, cùng cột hay cùng chéo → đây là một nghiệm hợp lệ

## Task 2: Stochastic Hill Climbing 1



**Initial board (conflicts=8): [2 6 3 6 4 4 2 6]**

- Bàn cờ 8×8 ban đầu sinh ngẫu nhiên.

- Có 8 xung đột, nghĩa là nhiều quân hậu đang ăn nhau (trùng hàng và chéo).

**Step 1: conflicts=5**

**Step 2: conflicts=3**

**Step 3: conflicts=2**

**Step 4: conflicts=1**

**Reached local optimum.**

- Sau mỗi bước, thuật toán chọn ngẫu nhiên một nước đi giúp giảm số xung đột.

- Không phải luôn chọn bước tốt nhất như thuật toán “steepest ascent”.

- Sau bước thứ 4, không còn nước đi nào giúp giảm thêm số xung đột,

→ thuật toán bị kẹt trong cực tiểu cục bộ.

- Vẫn còn 1 conflict, nên chưa đạt nghiệm hoàn hảo.

**Board with 1 conflicts.**

**Final board: [2 6 3 1 7 4 0 3]**

**Final conflicts: 1**

- Bàn cờ cuối cùng chỉ còn 1 cặp hậu ăn nhau.

- Mảng [2 6 3 1 7 4 0 3] nghĩa là:

+ Cột 0 → Hậu ở hàng 2

+ Cột 1 → Hậu ở hàng 6

+ Cột 2 → Hậu ở hàng 3

+ Cột 3 → Hậu ở hàng 1

+ Cột 4 → Hậu ở hàng 7

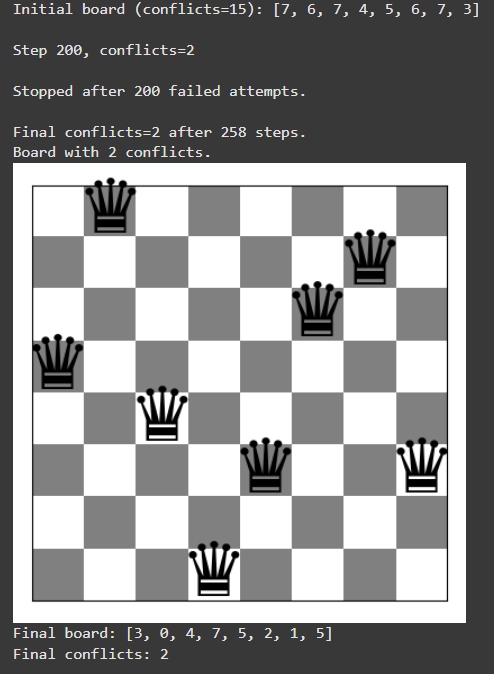
+ Cột 5 → Hậu ở hàng 4

+ Cột 6 → Hậu ở hàng 0

+ Cột 7 → Hậu ở hàng 3

→ Vẫn còn hai hậu trùng hàng 3, nên conflicts = 1.

## Task 3: Stochastic Hill Climbing 2



**Initial board (conflicts=15): [7, 6, 7, 4, 5, 6, 7, 3]**

- Đây là bàn cờ 8×8 được sinh ngẫu nhiên.

- Có tới 15 xung đột — rất nhiều quân hậu đang ăn nhau.

(Xung đột xảy ra nếu 2 hậu nằm cùng hàng hoặc cùng đường chéo.)

**Step 200, conflicts=2**

- Sau 200 bước di chuyển ngẫu nhiên, số xung đột giảm mạnh từ 15 → 2

- Nghĩa là thuật toán vẫn tìm được hướng cải thiện tốt.

- Tuy nhiên, sau đó nó không tìm được bước nào tốt hơn nữa.

**Stopped after 200 failed attempts.**

**Final conflicts=2 after 258 steps.**

- Sau 200 lần thử mà không cải thiện, thuật toán tự dừng lại.

- Nghĩa là không còn nước đi nào giúp giảm số xung đột, dù chưa đạt nghiệm hoàn hảo (conflicts=0).

**Final board: [5, 2, 6, 6, 3, 7, 4, 0]**

**Final conflicts: 2**

- Bàn cờ cuối cùng có 2 cặp hậu ăn nhau.

- Dù chưa hoàn hảo, nhưng khá gần nghiệm đúng (6/8 quân hậu an toàn).

## Task 4: Hill Climbing Search with Random Restarts

**Steepest-Ascent Hill Climbing**

=== Restart 1 ===

Initial board (conflicts=8): [7 5 0 0 1 5 0 1]

Step 1: conflicts=5

Step 2: conflicts=3

Step 3: conflicts=2

Step 4: conflicts=2

Reached local minimum.

=== Restart 2 ===

Initial board (conflicts=12): [7 1 5 4 7 5 4 5]

Step 1: conflicts=7

Step 2: conflicts=4

Step 3: conflicts=2

Step 4: conflicts=1

Step 5: conflicts=1

Reached local minimum.

=== Restart 3 ===

Initial board (conflicts=8): [0 2 0 2 3 3 6 7]

Step 1: conflicts=5

Step 2: conflicts=3

Step 3: conflicts=1

Step 4: conflicts=1

Reached local minimum.

=== Restart 4 ===

Initial board (conflicts=6): [6 2 1 7 2 4 5 0]

Step 1: conflicts=4

Step 2: conflicts=3

Step 3: conflicts=2

Step 4: conflicts=2

Reached local minimum.

=== Restart 5 ===

Initial board (conflicts=8): [6 3 2 7 1 2 2 3]

Step 1: conflicts=5

Step 2: conflicts=3

Step 3: conflicts=1

Step 4: conflicts=1

Reached local minimum.

=== Restart 6 ===

Initial board (conflicts=4): [5 0 1 6 5 3 4 7]

Step 1: conflicts=3

Step 2: conflicts=3

Reached local minimum.

=== Restart 7 ===

Initial board (conflicts=5): [1 7 2 2 7 3 6 4]

Step 1: conflicts=3

Step 2: conflicts=2

Step 3: conflicts=2

Reached local minimum.

=== Restart 8 ===

Initial board (conflicts=6): [3 1 7 7 2 1 6 2]

Step 1: conflicts=4

Step 2: conflicts=2

Step 3: conflicts=1

Step 4: conflicts=1

Reached local minimum.

=== Restart 9 ===

Initial board (conflicts=6): [2 3 3 7 0 1 0 4]

Step 1: conflicts=3

Step 2: conflicts=2

Step 3: conflicts=1

Step 4: conflicts=1

Reached local minimum.

=== Restart 10 ===

Initial board (conflicts=7): [2 3 6 7 7 5 4 6]

Step 1: conflicts=3

Step 2: conflicts=2

Step 3: conflicts=1

Step 4: conflicts=1

Reached local minimum.

=== Restart 11 ===

Initial board (conflicts=10): [1 1 4 7 4 4 7 1]

Step 1: conflicts=6

Step 2: conflicts=3

Step 3: conflicts=2

Step 4: conflicts=2

Reached local minimum.

=== Restart 12 ===

Initial board (conflicts=9): [3 3 1 3 6 2 3 5]

Step 1: conflicts=5

Step 2: conflicts=2

Step 3: conflicts=1

Step 4: conflicts=1

Reached local minimum.

=== Restart 13 ===

Initial board (conflicts=10): [3 1 1 6 6 7 3 6]

Step 1: conflicts=6

Step 2: conflicts=4

Step 3: conflicts=2

Step 4: conflicts=1

Step 5: conflicts=1

Reached local minimum.

=== Restart 14 ===

Initial board (conflicts=9): [4 3 4 0 3 7 0 7]

Step 1: conflicts=6

Step 2: conflicts=4

Step 3: conflicts=2

Step 4: conflicts=2

Reached local minimum.

=== Restart 15 ===

Initial board (conflicts=5): [3 4 7 5 3 2 6 4]

Step 1: conflicts=3

Step 2: conflicts=2

Step 3: conflicts=1

Step 4: conflicts=1

Reached local minimum.

=== Restart 16 ===

Initial board (conflicts=9): [7 2 0 4 5 3 2 3]

Step 1: conflicts=6

Step 2: conflicts=4

Step 3: conflicts=3

Step 4: conflicts=2

Step 5: conflicts=2

Reached local minimum.

=== Restart 17 ===

Initial board (conflicts=9): [0 1 4 4 4 6 7 5]

Step 1: conflicts=5

Step 2: conflicts=3

Step 3: conflicts=1

Step 4: conflicts=1

Reached local minimum.

=== Restart 18 ===

Initial board (conflicts=8): [7 3 5 4 2 6 1 3]

Step 1: conflicts=4

Step 2: conflicts=3

Step 3: conflicts=2

Step 4: conflicts=1

Step 5: conflicts=1

Reached local minimum.

=== Restart 19 ===

Initial board (conflicts=8): [5 7 6 3 7 7 6 3]

Step 1: conflicts=5

Step 2: conflicts=2

Step 3: conflicts=2

Reached local minimum.

=== Restart 20 ===

Initial board (conflicts=6): [5 5 2 0 5 1 3 6]

Step 1: conflicts=3

Step 2: conflicts=1

Step 3: conflicts=1

Reached local minimum.

=== Restart 21 ===

Initial board (conflicts=10): [5 4 7 5 1 6 7 4]

Step 1: conflicts=6

Step 2: conflicts=3

Step 3: conflicts=2

Step 4: conflicts=0

Found solution!

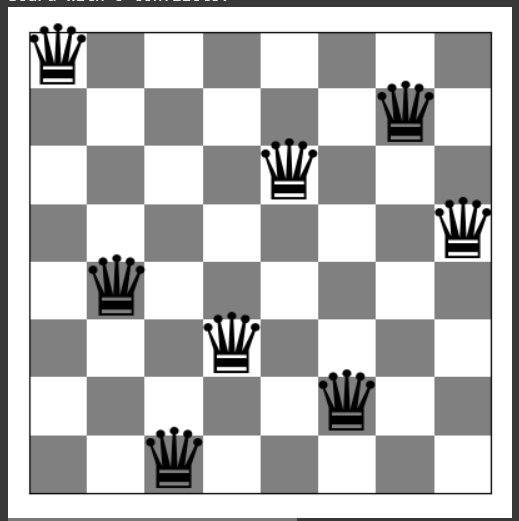
Found global solution!

Best conflicts after 21 restarts: 0

Board with 0 conflicts.

Best board: [0 4 7 5 2 6 1 3]

Best conflicts: 0



**Hiệu suất:**

- Thuật toán đã phải trải qua 21 lần khởi động lại mới tìm thấy lời giải toàn cục.

- Hầu hết các lần khởi động lại đều mắc kẹt tại tối ưu cục bộ (Local Minimum) với số xung đột thấp nhất là 1 hoặc 2 (ví dụ: Restart 1, 2, 3...).

- Lần khởi động lại thành công (Restart 21):

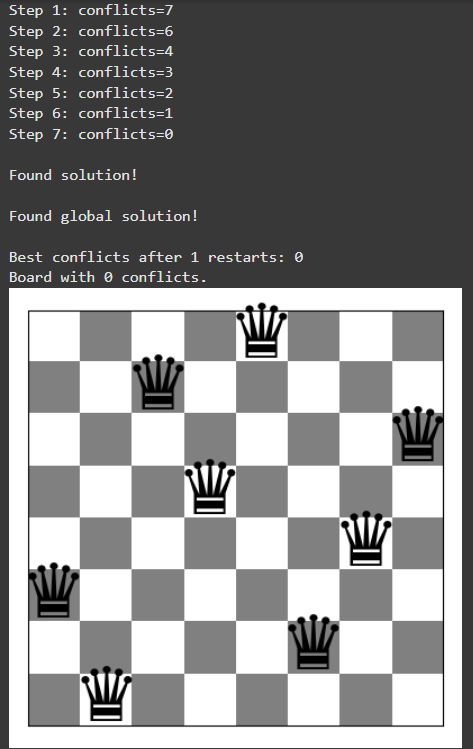
+ Bắt đầu với xung đột = 10.

+ Giảm dần: 10→6→3→2→0.

+ Tìm thấy lời giải tại Bước 4.

- Kết quả: best\_conflicts=0 sau 21 lần khởi động lại.

**Stochastic Hill Climbing**



**Hiệu suất:**

- Lần khởi động lại thành công (Restart 1):

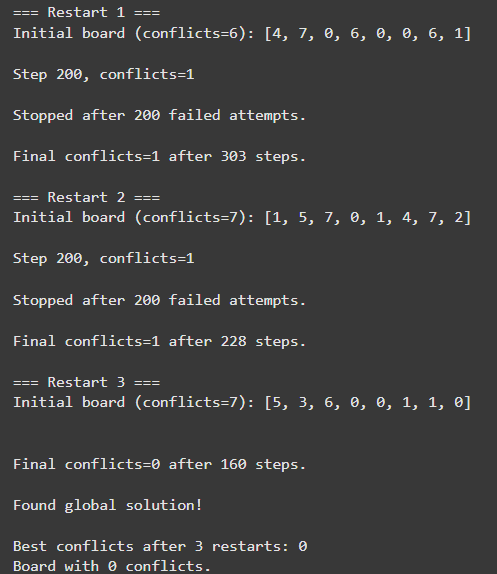
+ Bắt đầu với xung đột = 9.

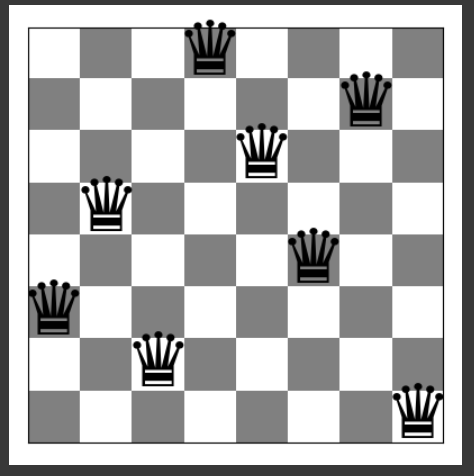
+ Giảm dần: 9→7→6→4→3→2→1→0.

+ Tìm thấy lời giải tại Bước 7.

- Kết quả: best\_conflicts=0 sau 1 lần khởi động lại.

**First-Choice Hill Climbing**





- Restart 1 & 2:

+ Bắt đầu với xung đột 6 và 7.

+ Cả hai lần đều Stopped after 200 failed attempts và kết thúc với xung đột = 1. Thuật toán đã thử 200 nước đi không cải thiện được trạng thái hoặc không tìm được nước đi cải thiện

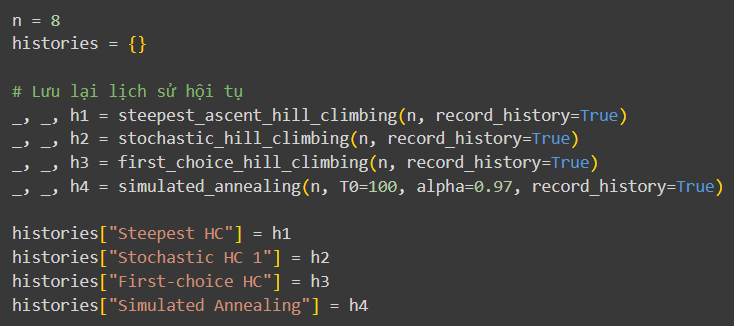
- Lần khởi động lại thành công (Restart 3):

+ Bắt đầu với xung đột = 7.

+ Final conflicts=0 after 160 steps.

+ Tìm thấy lời giải (xung đột = 0) trong lần khởi động lại thứ 3.

## Algorithm Convergence

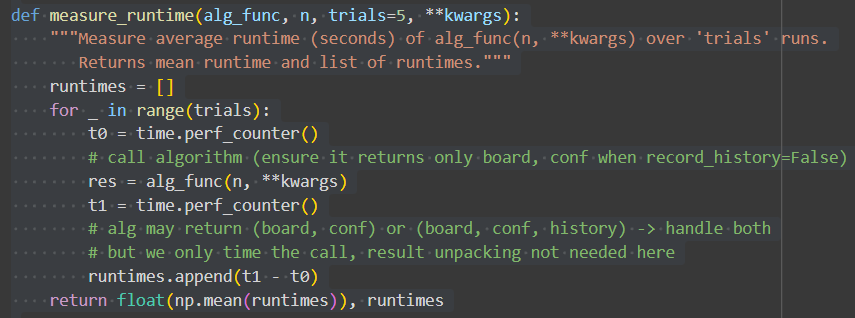


n = 8: Đặt kích thước bàn cờ là 8 (bài toán 8-Queens).

Gọi hàm với record\_history=True: Mỗi hàm thuật toán được gọi với tham số record\_history=True. Điều này bắt buộc hàm phải trả về giá trị thứ ba là lịch sử xung đột (một danh sách chứa số lượng xung đột tại mỗi bước).

histories = {}: Các danh sách lịch sử xung đột (h1, h2, h3, h4) được lưu trữ trong từ điển histories với tên gọi tương ứng.

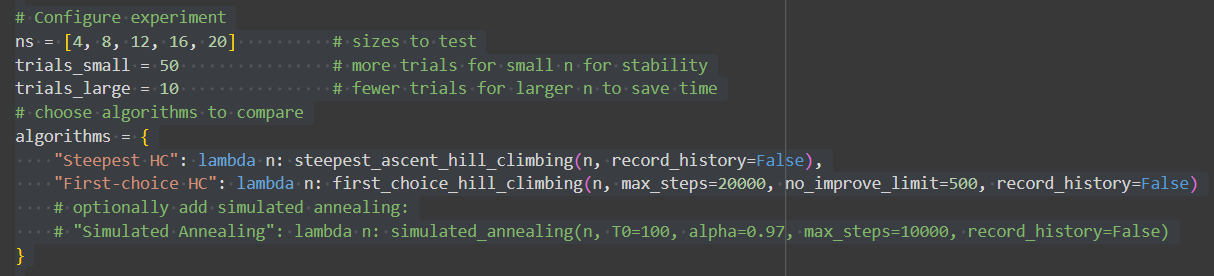
## Problem Size Scalability



Mục đích: Đo thời gian thực hiện của hàm thuật toán (alg\_func) trên bàn cờ kích thước n qua nhiều lần lặp (trials).

time.perf\_counter(): Sử dụng để đo thời gian với độ chính xác cao.

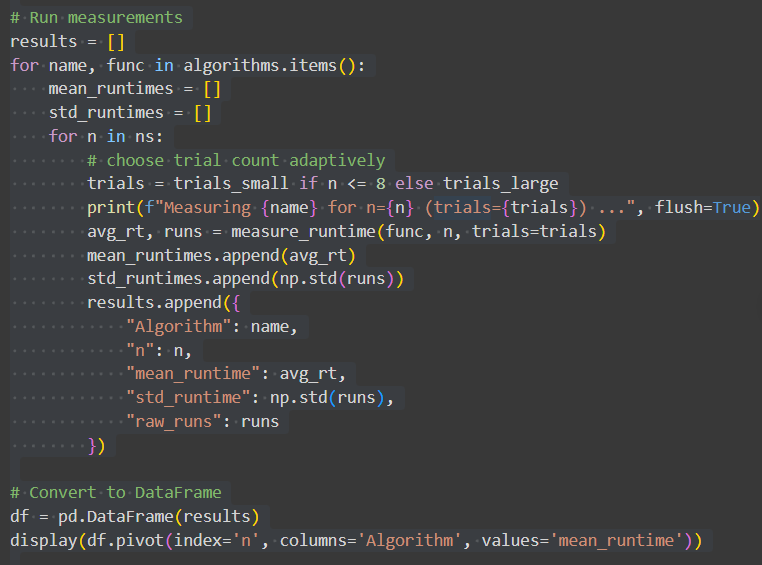
Trả về: Thời gian chạy trung bình (mean runtime) và danh sách các lần chạy (raw\_runs) để tính độ lệch chuẩn.



ns: Danh sách các kích thước bài toán (n) cần thử nghiệm.

trials\_small/trials\_large: Số lần thử nghiệm được điều chỉnh: nhiều hơn cho n nhỏ (để đạt độ ổn định thống kê) và ít hơn cho n lớn (để tiết kiệm thời gian, do thời gian chạy tăng nhanh).

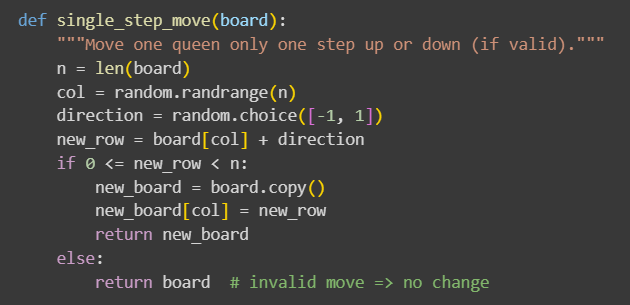
algorithms: Sử dụng hàm lambda để gói gọn việc gọi hàm thuật toán với các tham số mặc định (ví dụ: record\_history=False để tối ưu thời gian).



Vòng lặp chạy qua tất cả các cặp (thuật toán, n) và gọi measure\_runtime.

Kết quả được lưu vào DataFrame Pandas, sau đó sử dụng df.pivot() để tạo một bảng tóm tắt so sánh thời gian chạy trung bình của các thuật toán theo kích thước n.

## Exploring other Local Moves Operators



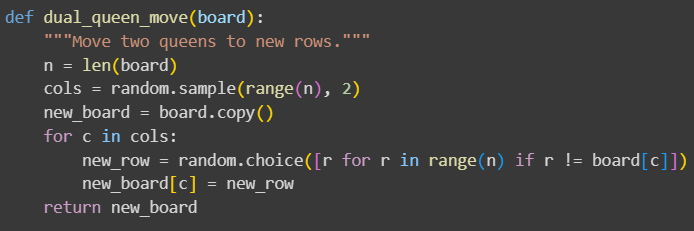
Mô tả: Đây là toán tử di chuyển có biên độ nhỏ nhất. Nó chọn ngẫu nhiên một quân Hậu và di chuyển nó chỉ một hàng lên hoặc xuống.

Không gian lân cận: Tạo ra một không gian lân cận nhỏ hơn nhiều so với toán tử mặc định (di chuyển một quân Hậu tới bất kỳ hàng nào).

## 

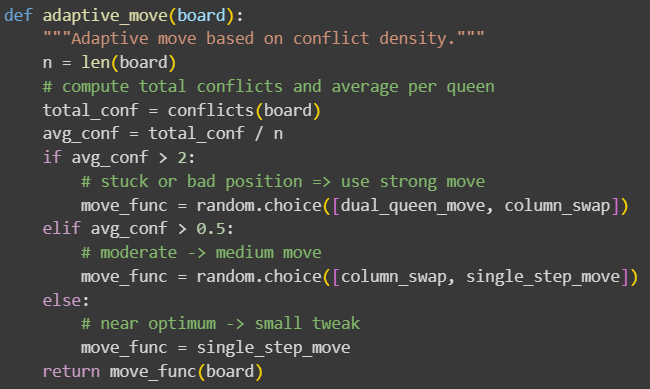
Mô tả: Chọn ngẫu nhiên hai cột và hoán đổi vị trí hàng của hai quân Hậu ở hai cột đó.

Không gian lân cận: Là một loại di chuyển có biên độ lớn (large-step move) vì nó thay đổi vị trí của hai quân Hậu đồng thời.



Mô tả: Chọn ngẫu nhiên hai cột và di chuyển mỗi quân Hậu ở hai cột đó đến một hàng mới hoàn toàn ngẫu nhiên.

Không gian lân cận: Đây là một di chuyển có biên độ lớn khác, tạo ra sự ngẫu nhiên cao hơn column\_swap vì vị trí mới của hai quân Hậu không liên quan đến nhau.

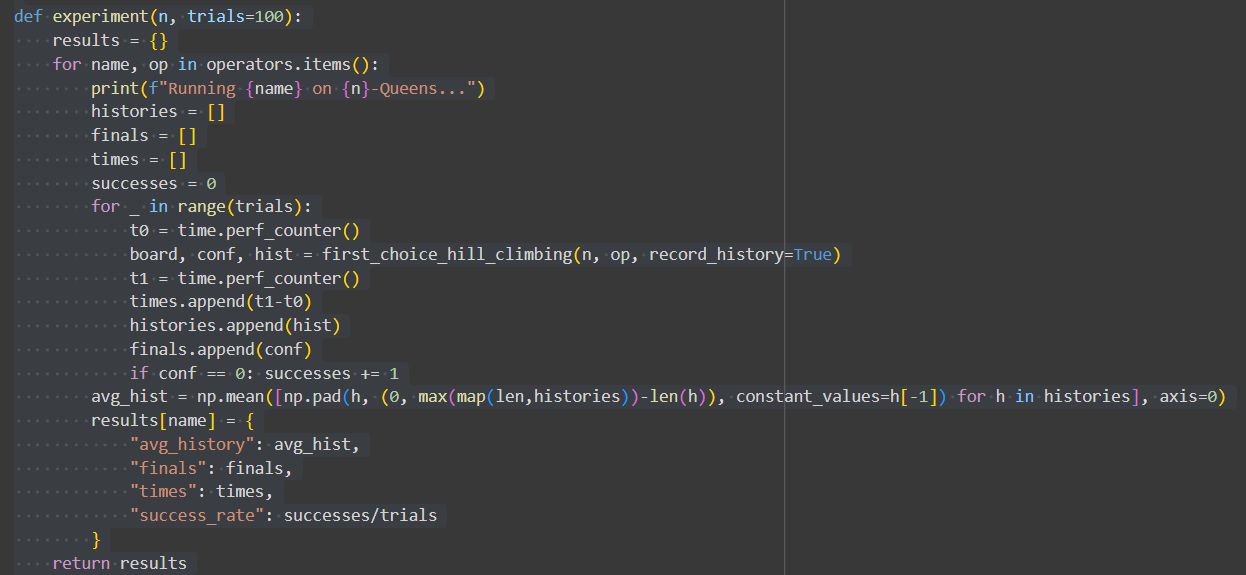


Mô tả: Toán tử này là một chiến lược meta-heuristic (siêu heuristic). Nó chọn toán tử di chuyển dựa trên mức độ xung đột trung bình hiện tại (avg\_conf).

Logic:

+ Xung đột cao: Chọn nước đi mạnh (dual\_queen\_move hoặc column\_swap) để khám phá (exploration).

+ Xung đột thấp: Chọn nước đi yếu (single\_step\_move) để khai thác (exploitation/tinh chỉnh) xung quanh cực tiểu cục bộ.



- Chức năng: Hàm experiment chạy thuật toán First-Choice Hill Climbing nhiều lần (trials) cho mỗi toán tử di chuyển.

- Sử dụng toán tử tùy chỉnh: Điểm cốt lõi là cách gọi hàm: first\_choice\_hill\_climbing(n, op, record\_history=True). Hàm first\_choice\_hill\_climbing (Task 3) phải được sửa đổi để sử dụng hàm op (toán tử di chuyển) thay vì tạo nước đi ngẫu nhiên mặc định (nếu op được cung cấp).

- Xử lý Lịch sử Trung bình:

+ Các lần chạy khác nhau sẽ có số bước lặp (độ dài lịch sử) khác nhau.

+ Đoạn code sử dụng np.pad() để đệm các lịch sử ngắn hơn bằng giá trị xung đột cuối cùng (h[-1]) của chúng.

+ Sau đó, nó tính giá trị trung bình (np.mean(..., axis=0)) cho mỗi bước lặp, tạo ra một đường cong hội tụ trung bình mượt mà, đại diện cho hiệu suất tổng thể.