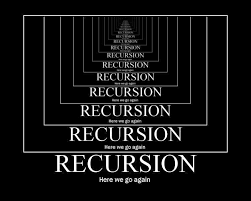
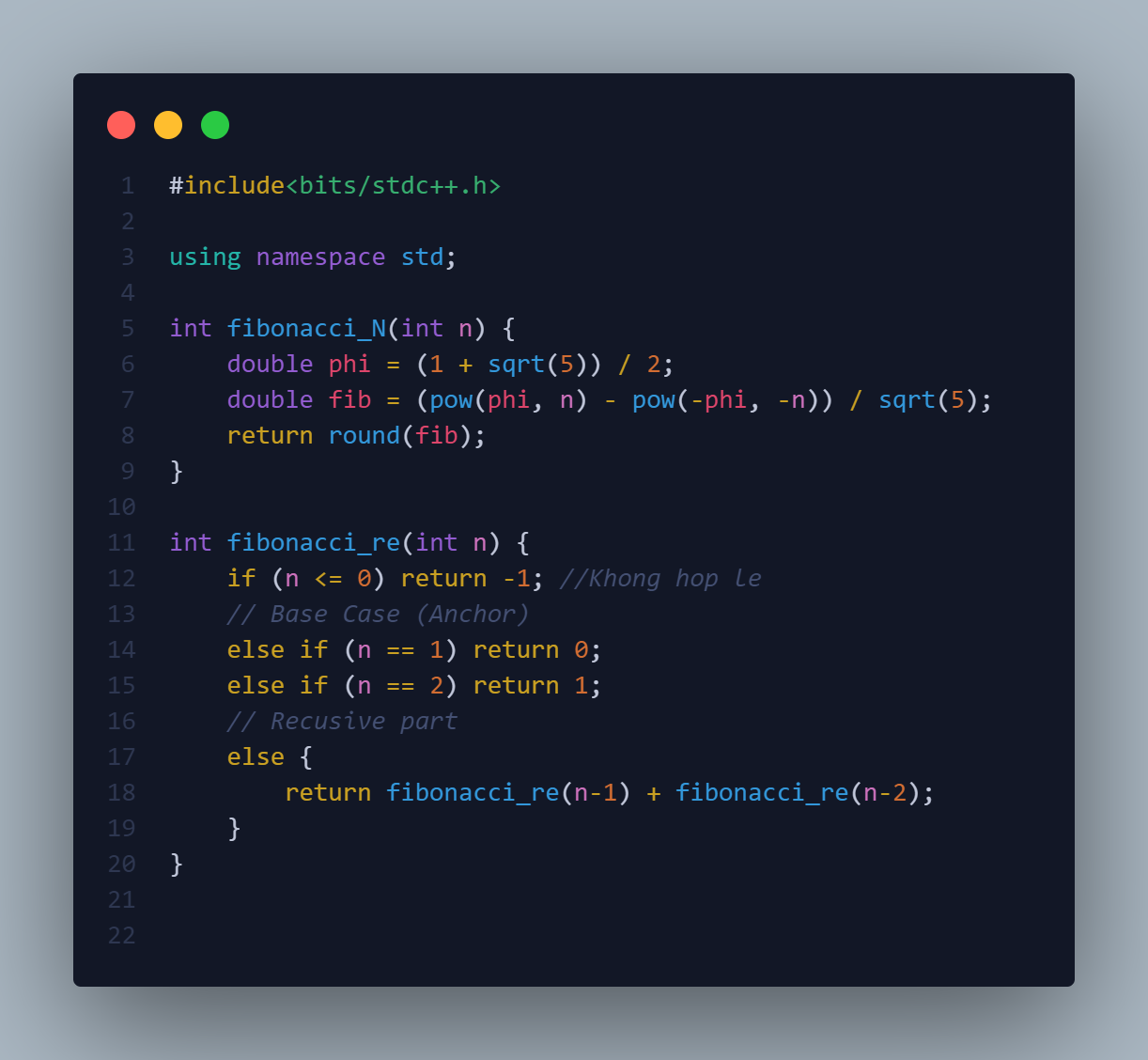
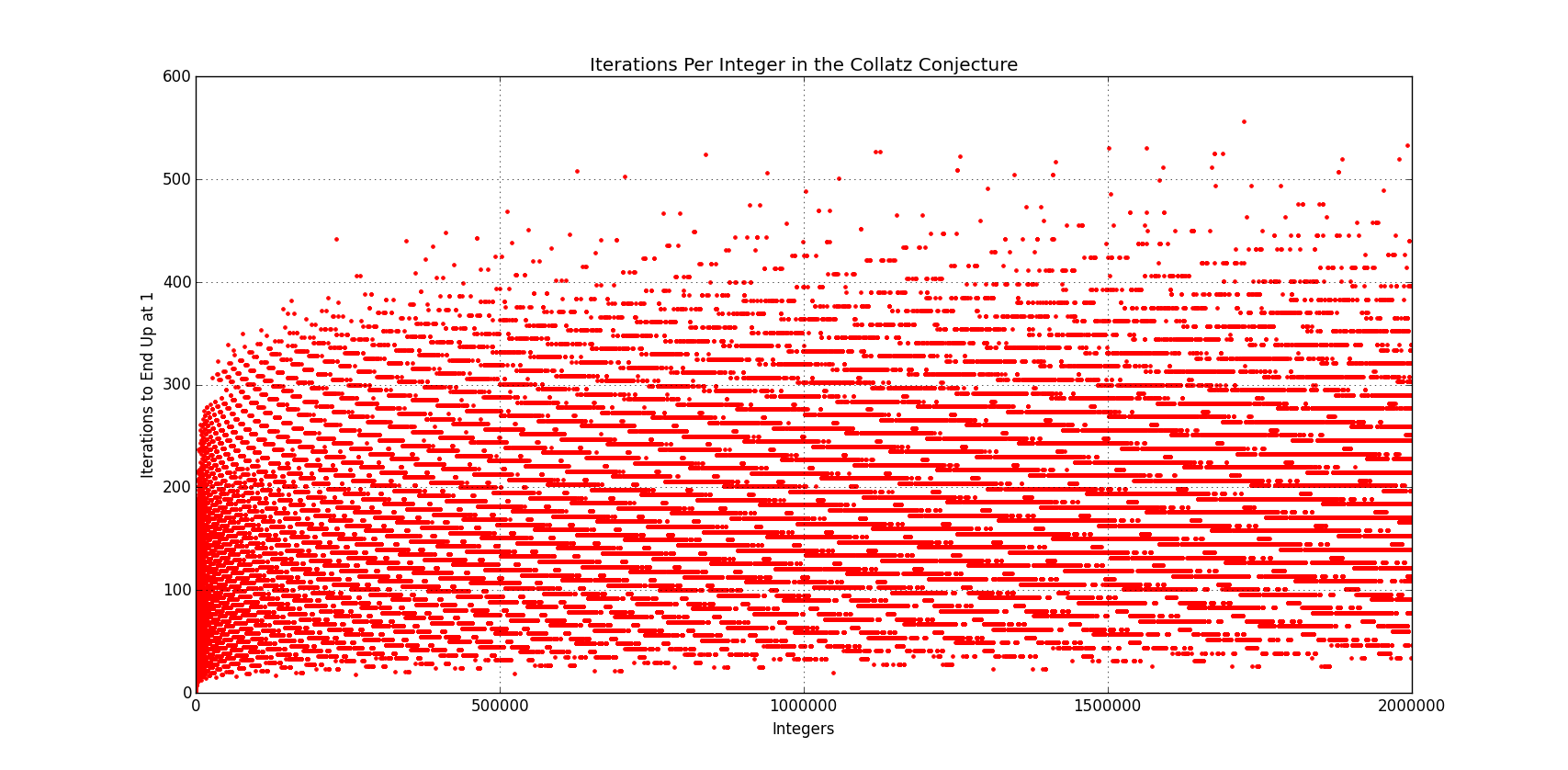
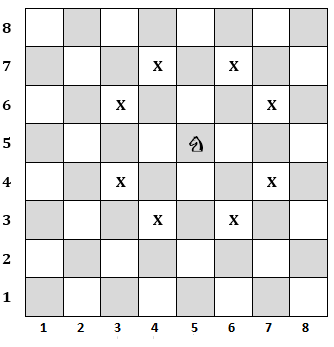
ĐỆ QUY (RECURSION)



1. Khái niệm về đệ quy, về giải thuật đệ quy
   1. Khái niệm đệ quy
      * Ta nói một đối tượng là đệ quy chính nó nếu nó được định nghĩa qua chính nó hoặc một đối tượng khác cùng dạng với chính nó bằng phép quy nạp.
      * Trong khoa học máy tính và lập trình, khi một hàm có thể gọi lại chính nó trong chính quá trình thực thi nó thì đó gọi là đệ quy. Việc hàm đó gọi đệ quy nhằm giải quyết các thao tác phức tạp mà có thể phân chia thành các bài toán “nhỏ hơn” có cùng cấu trúc.
   2. Giải thuật đệ quy
      * Nếu lời giải của một bài toán A nào đó được thực hiện bằng lời giải của bài toán A’ có dạng giống như A thì đó là một lời giải đệ quy. Giải thuật tương ứng với lời giải như vậy gọi là giải thuật đệ quy. Một điểm cần lưu ý là: A’ tuy có dạng giống như A, nhưng theo một nghĩa nào đó, nó phải “nhỏ” hơn A, dễ giải hơn A và việc giải nó không cần dùng đến A.
      * Định nghĩa một hàm đệ quy hay thủ tục đệ quy gồm 2 phần:
        + Phần neo (anchor) : Phần này được thực hiện khi bài toán rơi vào các trường hợp có thể coi là đơn giản nhất, có thể giải trực tiếp mà không cần đến bài toán con nào cả.
        + Phần đệ quy : Trong trường hợp bài toán chưa thể giải được bằng phần neo, ta xác định những bài toán con và gọi đệ quy giải những bài toán con đó, việc tiếp diễn giải các bài toán con sẽ dẫn đến phần neo. Kết hợp lời giải của những bài toán con để giải bài toán ta quan tâm.

* Phần đệ quy thể hiện tính quy nạp của lời giải, phần neo quyết định tới tính hữu hạn dừng của lời giải.

1. Một vài ví dụ đệ quy
   1. Dãy số Fibonacci
      * Đây là bài toán kinh điển được đặt ra rất lâu trong lịch sử. Dãy Fibonacci là dãy vô hạn các số tự nhiên bắt đầu bằng 2 phần tử 0 hoặc 1 và , các phần tử sau đó được thiết lập theo quy tắc: mỗi phần tử luôn bằng tổng hai phần tử trước đó.
        + Công thức truy hồi của dãy Fibonacci là :
        + Công thức tổng quát cho số hạng thứ n của dãy Fibonacci là:
      * Việc xác định số Fibonaccci thứ N có thể dựa vào công thức tổng quát hay có thể xây dựng dựa trên công thức tuy hồi của dãy bằng đệ quy.
   2. Giả thuyết của Collatz (Collatz conjecture)
      * Collatz đưa ra giả thuyết rằng: Với một số nguyên dương X, nếu X chẵn thì ta thực hiện gán , nếu X lẻ thì ta thực hiện gán: . Sau một số hữu hạn bước, ta sẽ thu được .
      * Nếu đặt giả thuyết trên là đúng đắn, ta có thể đặt ra một vấn đề: Cho trước số 1 cùng với 2 phép toán là (\* 2) và (), hãy sử dụng một cách hợp lí hai phép toán đó để biến số 1 thành một giá trị nguyên dương X cho trước ?
        + VD:
        + Dễ thấy lời giải bài toán gần như là thứ tự ngược lại của phép biến đổi Collatz:
2. Một số ứng dụng của đệ quy
   1. Giải thuật đệ quy thường được sử dụng để triển khai các giải thuật phức tạp như backtracking, binary search, sorting
   2. Một vài ứng dụng trong AI: Có thể xây dựng các kiến trúc mạng Neural phức tạp. Ví dụ, Mạng nơ-ron hồi quy (RNN) sử dụng đệ quy để truyền thông tin từ các nút trước đó tới các nút hiện tại.
   3. Sử dụng trong cây quyết định (Decision Trees): Một số thuật toán xây dựng cây quyết định sử dụng đệ quy như ID3, CART, sử dụng kỹ thuật đệ quy để chia dữ liệu thành các nhánh con dựa trên các thuộc tính của dữ liệu.
   4. Đồ thị: 2 thuật toán phổ biến trên đồ thị là DFS và BFS hoàn toàn có thể xây dựng dựa trên lời gọi đệ quy vì đệ quy rất hữu ích trong việc xử lí cây.
3. Bài toán giải quyết bằng đệ quy : Bài toán Mã đi tuần (knight's tour problem)
   1. Các định nghĩa, khái quát bài toán và ý tưởng
      * Mã đi tuần là bài toán về việc di chuyển một quân mã trên bàn cờ ô. Quân Mã được đặt ở một ô trên một bàn cờ trống và di chuyển để đi qua mỗi ô trên bàn cờ đúng một lần.
      * Nếu một quân Mã đi hết vị trí và tại vị trí cuối cùng có thể di chuyển đến vị trí bắt đầu thì gọi là một hành trình đóng. Nếu Mã đã đi hết 64 ô của bàn cờ và từ ô cuối của hành trình không thể về ô xuất phát trong 1 nước thì đó là hành trình mở.
      * Nước đi của quân mã giống như hình chữ L và nó có thể di chuyển tất cả các hướng. Ở một vị trí thích hợp thì quân mã có thể di chuyển đến được 8 vị trí.



* + - Gọi là độ dài bước đi trên các trục Oxy. Một bước đi hợp lệ thoả mãn : . Để đơn giản ta tạo mảng X[], Y[] để chứa các giá trị mà mỗi X[i],Y[i] sẽ là một cách di chuyển của quân mã
    - Ý tưởng để tìm đường đi cho quân Mã như sau: Sử dụng thuật toán quay lui (Backtracking) để thử tất cả các khả năng di chuyển của quân mã cho đến khi tìm được đường đi phù hợp hoặc không còn khả năng di chuyển nào khả dĩ.
  1. Thiết kế lời giải :
     + Khởi tạo bàn cờ NxN với tất cả các ô chưa được đi qua.
     + Đánh dấu ô xuất phát là đã đi qua.
     + Đặt biến đếm số nước đi hiện tại là 1.
     + Sử dụng một mảng chứa các bước di chuyển hợp lệ của quân mã (Ví dụ: [(2, 1), (1, 2), (-1, 2), (-2, 1), (-2, -1), (-1, -2), (1, -2), (2, -1)]).
     + Gọi hàm đệ quy solve\_knight\_tour(board, N, row, col, moves, move\_count) với các tham số:
       - board: Bàn cờ hiện tại.
       - N: Kích thước bàn cờ.
       - row, col: Vị trí hiện tại của quân mã.
       - moves: Các bước di chuyển hợp lệ của quân mã.
       - move\_count: Số nước đi hiện tại.
     + Trong hàm solve\_knight\_tour():
       - Nếu move\_count bằng , tức là đã đi qua tất cả các ô trên bàn cờ, lời giải đã được tìm thấy. Trả về True.
       - Lặp qua tất cả các bước di chuyển hợp lệ của quân mã:
         1. Tính toán vị trí tiếp theo (next\_row, next\_col) dựa trên bước di chuyển hiện tại.
         2. Kiểm tra nếu nước đi tiếp theo hợp lệ (không ra khỏi bàn cờ và chưa đi qua):

Đánh dấu ô tiếp theo là đã đi qua (với move\_count).

Gọi đệ quy solve\_knight\_tour() với vị trí tiếp theo và tăng move\_count lên 1.

Nếu đệ quy trả về True, tức là đã tìm thấy lời giải, trả về True.

Nếu không, đánh dấu ô tiếp theo là chưa đi qua (đặt lại giá trị ban đầu).

* + - * Nếu không có bước di chuyển nào khả dĩ, trả về False.
    - Nếu hàm solve\_knight\_tour() trả về False sau khi thử tất cả các khả năng, in ra thông báo "Không tìm thấy lời giải".
  1. Cài đặt:
  2. Đánh giá thuật toán :
     + Có thể có được nhiều lời giải khác nhau tuỳ thuộc vào bước khởi đầu của quân mã và quyết định di chuyển ưu tiên napf được thực hiện khi có nhiều lựa chọn. Do đó, thuật toán quay lui này không đảm bảo tìm ra tất cả lời giải và không đảm bảo tính chất bất biến của trường hợp này
     + Một số định lí liên quán đến bài toán mã đi tuần:
       - Định lí Warnsdorff: Trong bất kỳ bàn cờ (NxN), nếu mã không bị chặn lại và di chuyển vào một ô không đi qua trước đó, thì luôn có ít nhất một nước đi hợp lệ cho mã.
       - Định lí Euler: Euler khẳng định rằng trong một bàn cờ vuông (NxN) (với N >= 5), mã có một chu trình Hamilton, tức là có thể đi qua mỗi ô trên bàn cờ đúng một lần và quay về ô ban đầu.
     + Độ phức tạp thuật toán :
       - Trong trường hợp tốt nhất, thuật toán có thể tìm ra lời giải ngay từ bước đầu tiên tức là O(1).
       - Trong trường hợp xấu nhất, thuật toán phải thử tất cả các khả năng di chuyển cho tất cả các ô trên bàn cờ trước khi tìm ra lời giải hoặc không có lời giải. Vì mỗi ô trên bàn cờ có tối đa 8 khả năng di chuyển và bàn cờ kích thước (N \* N). Trong trường hợp này, độ phức tạp sẽ là : .
  3. Cải tiến thuật toán:
     + Ta có thể cải tiến thuật toán bằng phương pháp quy hoạch động: Sử dụng một bảng điểm để lưu trữ số lần thử di chuyển từ mỗi ô. Bằng cách này, ta có thể ưu tiên các ô có ít khả năng di chuyển hợp lệ hơn và tránh các đường đi không khả thi



1. Bài tập sử dụng đệ quy
   1. Cho 1 xâu kí tự, in ra toàn bộ các xâu con đối xứng có thể có trong xâu đã cho
   2. Thiết kế thuật toán để giải bài toán Tháp Hà Nội (Tower of Hanoi)