ĐẠI HỌC QUỐC GIA TPHCM

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

PHÂN TÍCH VÀ THIẾT KẾ THUẬT TOÁN

Lý thuyết trò chơi

Nhóm 5 thực hiện

Môn học: CS112.P11.KHTN

Sinh viên thực hiện:

Nguyễn Thiên Bảo - 23520127

Trần Lê Minh Nhật - 23521098

Giáo viên hướng dẫn:

Nguyễn Thanh Sơn

ngày 4 tháng 12 năm 2024

Mục lục

1 Bài 1: Đối Kháng

1.1 Đề bài

Hằng năm ở thành phố X, sẽ tổ chức một cuộc thi đối kháng hai người. Ban đầu người ta sẽ giao cho hai bạn một số nguyên dương p. Hai người thi đấu theo lượt, A đi trước B. Nếu ai làm cho p bằng 0 thì người đó thắng. Trong một lượt chơi, người chơi thực hiện thao tác sau:

- Nếu p lẻ, người chơi được chọn tăng p hoặc giảm p 1 đơn vị.
- Nếu p chẵn, thì người chơi bắt buộc giảm p xuống 1 nửa ($p := \frac{p}{2}$).

Cho trước số nguyên dương p. Bạn A luôn đi trước B, nếu cả hai đều chơi tối ưu thì bạn A luôn thắng được không? (Xuất ra màn hình `YES` nếu A luôn thắng hoặc ngược lại B luôn thắng).

1.2 Yêu cầu

1. Cho biết đây thuộc loại trò chơi nào đã trình bày trong buổi thảo luận.
2. Có 3 giới hạn dành cho p, mỗi giới hạn hãy:
   - Trình bày ý tưởng, phương pháp dùng thiết kế thuật toán để giải quyết bài toán.
   - Trình bày mã giả bài toán.

   Các giới hạn bao gồm:

   (a) $p \leq 10$

   (b) $p \leq 10^6$

   (c) $p \leq 10^{18}$ (khuyến khích, không ép buộc các bạn làm).

1.3 Phân loại trò chơi

Đây là một **trò chơi impartial** thuộc lý thuyết trò chơi. Cả hai người chơi đều có cùng tập hợp các nước đi hợp lệ từ mỗi trạng thái, và kết quả phụ thuộc vào trạng thái hiện tại và người chơi nào đang tới lượt.

1.4 Giải quyết bài toán

Để xác định xem người chơi A (đi trước) luôn thắng hay không khi cả hai đều chơi tối ưu, chúng ta cần phân tích từng trạng thái của p và xác định xem trạng thái đó là trạng thái thắng hay thua cho người chơi đang tới lượt.

1.4.1 Phương pháp giải quyết theo từng giới hạn

$p \leq 10$: Sử dụng Backtracking với Memoization Ý tưởng:

- Sử dụng phương pháp **backtracking** để duyệt tất cả các khả năng từ mỗi trạng thái p.
- Áp dụng **memoization** để lưu trữ kết quả của từng trạng thái p nhằm tránh tính lại nhiều lần.
- Xác định trạng thái thắng/thua dựa trên các nước đi có thể từ trạng thái hiện tại.

Phương pháp:

- Đệ quy để kiểm tra tất cả các nước đi có thể từ p.
- Nếu tồn tại một nước đi dẫn đến trạng thái thua cho đối thủ, thì trạng thái hiện tại là trạng thái thắng.

Mã giả:

```
memo = {}


Hàm Win(p):
    Nếu p == 0:
        Trả về False  // Người chơi hiện tại thua
    Nếu p đã có trong memo:
        Trả về memo[p]
```

```
    Nếu p lẻ:
        // Có thể tăng hoặc giảm p
        nếu Win(p + 1) == False hoặc Win(p - 1) == False:
            memo[p] = True
            Trả về True
    Ngược lại:
        // p chẵn, bắt buộc giảm p xuống 1/2
        nếu Win(p / 2) == False:
            memo[p] = True
            Trả về True

    memo[p] = False
    Trả về False

// Chương trình chính
Nếu Win(p):
    In "YES"
Ngược lại:
    In "NO"
```

$p \leq 10^6$: Sử dụng Quy hoạch động (Dynamic Programming - DP) Ý tưởng:

- Sử dụng một mảng để lưu trữ kết quả thắng/thua cho từng giá trị của p.
- Tính dần kết quả từ $p = 0$ lên đến $p = 10^6$.

Phương pháp:

- Khởi tạo $Win[0] = False$ (vì người chơi không thể làm gì khi $p = 0$).
- Duyệt từ $p = 1$ đến $p = 10^6$, xác định $Win[p]$ dựa trên các nước đi có thể từ p.

Mã giả:

```
// Khởi tạo mảng Win
Tạo mảng Win[0..10^6]
Win[0] = False

Cho p từ 1 đến 10^6:
    Nếu p lẻ:
        // Có thể tăng hoặc giảm p
        Nếu (p + 1 <= 10^6 và Win[p + 1] == False) hoặc (p - 1 >= 0 và Win[p - 1] == False)
            Win[p] = True
        Ngược lại:
            Win[p] = False
    Ngược lại:
        // p chẵn, bắt buộc giảm p xuống 1/2
        Nếu Win[p / 2] == False:
            Win[p] = True
        Ngược lại:
            Win[p] = False

// Chương trình chính
Nếu Win[p]:
    In "YES"
Ngược lại:
    In "NO"
```

$p \leq 10^{18}$: Sử dụng Phân tích Toán học Ý tưởng:

- Với các giá trị rất lớn của p, việc lưu trữ hoặc tính toán từng trạng thái là không khả thi.
- Phân tích toán học để tìm ra quy luật tổng quát dựa trên tính chất của p.

Phương pháp:

- Quan sát rằng khi p chẵn, người chơi buộc phải chia p cho 2.
- Đếm số lần chia p cho 2 cho đến khi p trở thành số lẻ.
- Quy luật: Nếu số lần chia là chẵn, người chơi hiện tại thắng; nếu lẻ, thua.

Mã giả:

```
Hàm Win(p):
    Nếu p == 0:
        Trả về False
    Nếu p lẻ:
        Trả về True  // Người chơi hiện tại thắng
    c = 0
    Trong khi p chẵn:
        p = p / 2
        c = c + 1
    Nếu c mod 2 == 0:
        Trả về True  // Người chơi hiện tại thắng
    Ngược lại:
        Trả về False  // Người chơi hiện tại thua


// Chương trình chính
Nếu Win(p):
    In "YES"
Ngược lại:
    In "NO"
```

2 Bài 2: Trò Chơi Đồng Xu

2.1 Đề bài

Để cạnh tranh sức hút trò chơi ở thành phố X, tại thành phố Y cũng đã tổ chức một cuộc thi đối kháng hai người, A luôn đi trước B. Ban đầu, hai người chơi được giao một chồng gồm n đồng xu. Trong lượt chơi, bạn được phép chọn 1 hoặc 2, ... hoặc k đồng xu và bốc nó ra khỏi chồng (sau lượt này n sẽ giảm đi x đồng xu, x là số đồng xu bạn bốc ra, $x \leq k$). Nếu không thực hiện bốc được thì bạn thực hiện lượt đó sẽ thua.
Bài toán khá quen thuộc để tăng độ hấp dẫn. Phước sẽ dành cho các bạn câu đố sau. Với những giá trị nào của k ($k \leq n$) mà đảm bảo A luôn thắng. In ra số lượng k thỏa mãn.

2.2 Yêu cầu

1. Cho biết đây thuộc loại trò chơi nào đã trình bày trong buổi thảo luận.
2. Có 2 giới hạn dành cho n, mỗi giới hạn hãy:
   - Trình bày ý tưởng, phương pháp dùng thiết kế thuật toán để giải quyết bài toán.
   - Trình bày mã giả bài toán.

   Các giới hạn bao gồm:

   (a) $n \leq 1000$

   (b) $n \leq 10^{18}$ (khuyến khích, không ép buộc các bạn làm).

2.3 Phân loại trò chơi

Đây cũng là một **trò chơi impartial** thuộc lý thuyết trò chơi , tương tự như trò chơi Nim cổ điển. Người chơi có thể lấy một số đồng xu từ đống theo quy định, và người lấy đồng xu cuối cùng thắng.

2.4 Giải quyết bài toán

Để xác định số lượng giá trị k ($k \leq n$) sao cho người chơi A (đi trước) luôn thắng khi cả hai đều chơi tối ưu, chúng ta cần phân tích từng giá trị k và kiểm tra điều kiện thắng thua dựa trên quy luật của trò chơi.

2.4.1 Phương pháp giải quyết theo từng giới hạn

$n \leq 1000$: Sử dụng Quy hoạch động (Dynamic Programming - DP) Ý tưởng:

- Mỗi giá trị k từ 1 đến n xác định số lượng đồng xu tối đa có thể lấy trong mỗi lượt.
- Trò chơi tương đương với trò chơi Nim với một đống, và người chơi A thắng nếu n không chia hết cho $k + 1$.
- Do đó, đếm số lượng k từ 1 đến n sao cho $n \mod (k + 1) \neq 0$.

Phương pháp:

- Duyệt từng giá trị k từ 1 đến n.
- Kiểm tra điều kiện $n \mod (k + 1) \neq 0$.
- Đếm số k thỏa mãn.

Mã giả:

```
Hàm CountWinningK(n):
    count = 0
    Cho k từ 1 đến n:
        Nếu n mod (k + 1) != 0:
            count = count + 1
    Trả về count

// Chương trình chính
```

```
Đọc n
In CountWinningK(n)
```

$n \leq 10^{18}$: Sử dụng Số học (Number Theory) và Đếm Ước Ý tưởng:

- Tìm số lượng k từ 1 đến n sao cho $k+1$ không phải là ước của n.
- Tổng số k thỏa mãn là $n-$ số lượng ước của n trừ đi 1 (vì $k+1$ bắt đầu từ 2 đến $n+1$).

Phương pháp:

- Tìm tất cả các ước của n.
- Đếm số lượng ước d của n sao cho $d \geq 2$ và $d \leq n+1$.
- Số lượng k thỏa mãn là $n-$ số lượng ước thỏa mãn trên.

Mã giả:

```
Hàm CountDivisors(n):
    count = 0
    d = 1
    Trong khi d * d <= n:
        Nếu n mod d == 0:
            Nếu d >= 2 và d <= n + 1:
                count = count + 1
            Nếu d != n / d và (n / d) >= 2 và (n / d) <= n + 1:
                count = count + 1
        d = d + 1
    Trả về count

Hàm CountWinningK_Large(n):
    số_ước = CountDivisors(n)
```

```
    count = n - số_ước
    Trả về count

// Chương trình chính
Đọc n
In CountWinningK_Large(n)
```