TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

BTVN NHÓM 4:
LÝ THUYẾT TRÒ CHƠI

PHÂN TÍCH VÀ THIẾT KẾ THUẬT TOÁN

Nhóm 9

Sinh viên thực hiện:

Họ và tên	MSSV
Bùi Ngọc Thiên Thanh	23521436
Nguyễn Thái Sơn	23521356

Thành phố Hồ Chí Minh, 2024

Mục lục

1 Phương pháp giải **2**

1.1 $p \leq 10$ 2

1.2 $p \leq 10^6$ 2

1.3 $p > 10^6$ 3

2 Phân tích độ phức tạp **3**

1 Phân tích trò chơi **4**

1.1 Quy hoạch động (DP) 4

2 Mã giả **5**

3 Phân tích độ phức tạp **6**

4 Kết luận **7**

Bài 1: Đối kháng

Hằng năm ở thành phố X, sẽ tổ chức một cuộc thi đối kháng hai người. Ban đầu người ta sẽ giao cho 2 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 một nửa $p := p/2$.

Cho trước số nguyên dương p. Bạn A luôn đi trước bạn B, nếu cả 2 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 Phương pháp giải

1.1 $p \leq 10$

Sử dụng phương pháp **backtracking** (nhánh cận) để thử tất cả các trạng thái và kiểm tra trạng thái thắng/thua. Ta thử tất cả các nước đi từ trạng thái hiện tại và xác định liệu một người chơi có thể bắt đầu ở trạng thái thắng hay không.

Backtracking:

```
function canWin(p):
if p == 0:
    return False
if p is odd:
    return not canWin(p - 1) or not canWin(p + 1)
else:
    return not canWin(p / 2)
```

1.2 $p \leq 10^6$

Sử dụng **quy hoạch động**. Ta dùng một mảng ‘dp‘ để lưu trữ trạng thái thắng/thua cho mỗi giá trị p. Mỗi giá trị p chỉ được tính một lần, giúp tối ưu thời gian tính toán.

Quy hoạch động:

```
dp = [-1] * (10^6 + 10)

function canWin(p):
if p == 0:
    return False
if dp[p] != -1:
    return dp[p]
if p is odd:
    dp[p] = not canWin(p - 1) or not canWin(p + 1)
else:
    dp[p] = not canWin(p / 2)
return dp[p]
```

1.3 $p > 10^6$

Sử dụng **chiến thuật lý thuyết trò chơi**. Dựa trên tính chất của trò chơi, ta nhận thấy rằng nếu p là lẻ, người chơi luôn có lợi thế vì có hai lựa chọn khả thi. Nếu p là chẵn, trạng thái thắng/thua phụ thuộc vào $p/2$.

Chiến thuật lý thuyết trò chơi:

```
function canWinTheory(p):
while p > 0:
    if p is odd:
        return True
    p = p / 2
return False
```

2 Phân tích độ phức tạp

- **Backtracking**: Độ phức tạp $O(2^p)$, do thử tất cả các trạng thái.

- **Quy hoạch động**: Độ phức tạp $O(p)$, vì mỗi trạng thái chỉ tính một lần.
- **Chiến thuật lý thuyết trò chơi**: Độ phức tạp $O(\log p)$, vì mỗi lần giảm p là chia đôi.

Bài 2: Trò chơi đồng xu

1 Phân tích trò chơi

Bài toán này thuộc dạng trò chơi đối kháng (game theory), trong đó hai người chơi A và B thay phiên nhau bốc đồng xu từ một chồng có n đồng xu. Mỗi người có thể chọn từ 1 đến k đồng xu trong lượt của mình. Ai không thể bốc đồng xu thì sẽ thua.
Mục tiêu của bài toán là xác định số lượng các giá trị k (với $k \leq n$) sao cho A luôn có thể thắng nếu chơi tối ưu.

- Trạng thái thắng/thua: Nếu $n = 0$, người chơi không thể bốc đồng xu nào và sẽ thua.
- Cách xác định trạng thái thắng/thua: Nếu tại trạng thái có n đồng xu, A có thể bốc từ 1 đến k đồng xu sao cho đối thủ (B) rơi vào một trạng thái thua (trạng thái mà không còn nước đi hợp lệ), thì trạng thái n là trạng thái thắng đối với A.

1.1 Quy hoạch động (DP)

Sử dụng mảng dp để đánh dấu các trạng thái thắng/thua cho mỗi giá trị n đồng xu còn lại.

- $dp[i] =$ True nếu A có thể thắng khi bắt đầu với i đồng xu.
- $dp[i] =$ False nếu A không thể thắng khi bắt đầu với i đồng xu.

Ý tưởng giải quyết bài toán

Giới hạn $n \leq 1000$:

- Dùng phương pháp quy hoạch động để tính trạng thái thắng/thua cho mỗi giá trị n. Với mỗi n, ta kiểm tra các giá trị x (từ 1 đến k), và nếu A bốc x đồng xu, đối thủ sẽ rơi vào trạng thái thua.
- Đếm số lượng giá trị k sao cho A luôn thắng.

Giới hạn $n \leq 1e18$:

- Sử dụng lý thuyết trò chơi với công thức:

$$n \mod (k+1) \neq 0 \implies \text{A thắng.}$$

- Tìm tất cả các ước của n và loại trừ những giá trị k thỏa mãn $k + 1 = d$ (d là ước của n).
- Tính tổng số giá trị k thỏa mãn yêu cầu.

2 Mã giả

Quy hoạch động ($n \leq 1000$):

```
dp = [False] * (n + 1)

def canWin(n, k):
dp = [False] * (n + 1)
for i in range(1, n + 1):
    for x in range(1, min(k, i) + 1):
        if not dp[i - x]:
            dp[i] = True
            break
return dp[n]

def countWinningK(n):
count = 0
for k in range(1, n + 1):
    if canWin(n, k):
        count += 1
```

```
return count
```

Chiến thuật lý thuyết trò chơi ($n \leq 1e18$):

```
def countWinningK(n):
divisors = []
for d in range(1, int(n**0.5) + 1):
    if n % d == 0:
        divisors.append(d)
        if d != n // d:
            divisors.append(n // d)

total_k = n - 1
for d in divisors:
    if d - 1 > 0:
        total_k -= 1

return total_k
```

3 Phân tích độ phức tạp

- Quy hoạch động:
  - Độ phức tạp thời gian: $O(n \times k)$, vì với mỗi giá trị k, ta cần duyệt qua mảng dp có độ dài n để tính toán trạng thái thắng/thua.
  - Độ phức tạp không gian: $O(n)$, vì mảng dp cần có kích thước $n + 1$.
- Chiến thuật lý thuyết trò chơi:
  - Độ phức tạp thời gian: $O(\sqrt{n})$, vì chúng ta chỉ cần tìm tất cả các ước của n, và số ước của n là $O(\sqrt{n})$.
  - Độ phức tạp không gian: $O(\sqrt{n})$, do lưu trữ các ước trong mảng.

4 Kết luận

Đây là một bài toán đối kháng có thể giải quyết bằng cả phương pháp quy hoạch động (dành cho $n \leq 1000$) và lý thuyết trò chơi (dành cho $n \leq 1e18$). Cả hai phương pháp đều có thể tính toán số lượng các giá trị k sao cho A luôn thắng trong trò chơi này.