ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC MÁY TÍNH

LỚP CS115.P11.KHTN
MÔN PHÂN TÍCH VÀ THIẾT KẾ THUẬT TOÁN
Bài tập về nhà nhóm 4

CỬ NHÂN NGÀNH KHOA HỌC MÁY TÍNH

Nhóm 12
Nguyễn Trọng Tất Thành
Mã số sinh viên: 23521455
Hoàng Minh Thái
Mã số sinh viên: 23521414

Giảng viên hướng dẫn: Thầy Nguyễn Thanh Sơn

TP. HỒ CHÍ MINH, NĂM 2024

Mục lục

1.

1 Bài tập 1

1.1 Phân tích chung

- Bài toán thuộc loại biểu diễn dưới dạng cây.
- Nếu p là số lẻ, người chơi có lợi thế hơn vì có thể lựa chọn hai nước đi (tăng hoặc giảm 1 đơn vị).
- Nếu p là số chẵn, người chơi chỉ có một nước đi duy nhất là giảm p xuống còn $p/2$, điều này hạn chế khả năng chiến thắng.
- Mục tiêu của người chơi là đưa đối thủ vào trạng thái thua bằng cách đưa đối thủ về trạng thái bất lợi.

1.2 Giới hạn 1: $p \leq 10$ - Sử dụng Backtracking

Ý tưởng:

- Duyệt tất cả các trạng thái có thể của p.
- Nếu tồn tại một nước đi khiến đối thủ rơi vào trạng thái thua, thì trạng thái đó là trạng thái thắng.

Mã giả:

Algorithm 1 Backtracking cho $p \leq 10$

```
function canWin(p)
  if p == 0 then
    return False
  end if
  if p%2 == 1 then
    return not canWin(p-1) or canWin(p+1)
  else
    return not canWin(p/2)
  end if
end function=0
```

Độ phức tạp:

- Thời gian: $O(2^p)$, do thử tất cả các trạng thái.
- Không gian: $O(p)$, do sử dụng đệ quy.

1.3 Giới hạn 2: $p \leq 10^6$ - Sử dụng Quy hoạch động

Ý tưởng:

- Lưu trữ kết quả thắng/thua cho mỗi giá trị p trong một mảng `dp`.
- Trạng thái thắng/thua của p được tính dựa trên trạng thái của $p-1$, $p+1$, và $p/2$.

Mã giả:

Algorithm 2 Quy hoạch động cho $p \leq 10^6$

```
dp ← [−1] × (10^6 + 1)
function canWin(p)
  if p == 0 then
    return False
  end if
  if dp[p] ≠ −1 then
    return dp[p]
  end if
  if p%2 == 1 then
    dp[p] ← not canWin(p-1) or canWin(p+1)
  else
    dp[p] ← not canWin(p/2)
  end if
  return dp[p]
end function=0
```

Độ phức tạp:

- Thời gian: $O(p)$, do mỗi trạng thái chỉ được tính một lần.
- Không gian: $O(p)$, do sử dụng mảng `dp`.

1.4 Giới hạn 3: $p > 10^6$ - Phân tích Lý thuyết Trò chơi

Ý tưởng:

- Nếu p là số lẻ, người chơi A luôn có lợi thế vì có thể đưa đối thủ vào trạng thái bất lợi.
- Nếu p là số chẵn, người chơi phải giảm p xuống $p/2$, do đó trạng thái của $p/2$ quyết định kết quả.

Mã giả:

Algorithm 3 Lý thuyết trò chơi cho $p > 10^6$

```
function canWinTheory(p)
  while p > 0 do
    if p%2 == 1 then
      return True {A thắng nếu p lẻ}
    else
      p ← p/2
    end if
  end while
  return False {Nếu về 0, A thua}
end function=0
```

Độ phức tạp:

- Thời gian: $O(\log p)$, vì mỗi lần giảm p xuống một nửa.
- Không gian: $O(1)$, vì chỉ cần sử dụng một vài biến để theo dõi trạng thái.

1.5 Kết luận

- Với $p \leq 10$, phương pháp Backtracking giúp thử tất cả các khả năng.
- Với $p \leq 10^6$, Quy hoạch động là phương pháp hiệu quả.
- Với $p > 10^6$, phân tích lý thuyết trò chơi giúp giải bài toán với độ phức tạp thấp nhất.

2.

2 Bài tập 2

2.1 Phương pháp giải

- **$n \leq 1000$:**
  - Sử dụng QHD để kiểm tra trạng thái thắng/thua cho mỗi giá trị n.
- **$n \leq 10^{18}$:**
  - Sử dụng chiến thuật với công thức:

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

2.2 Mã giả

2.2.1 Quy hoạch động ($n \leq 1000$):

Mã giả cho QHD

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

def countWinningK(n):
    count = 0
    for k in range(1, n + 1):
        for x in range(1, k + 1):
            if n - x >= 0 and not dp[n - x]:
                dp[n] = True
                break
        if dp[n]:
            count += 1
    return count
```

2.2.2 Chiến thuật lý thuyết trò chơi ($n \leq 10^{18}$):

Mã giả cho chiến thuật lý thuyết trò chơi

```
def countWinningK(n):
    divisors = []
    for d in range(1, sqrt(n) + 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
```

2.3 Phân tích độ phức tạp

- **Quy hoạch động:** Độ phức tạp $O(n \cdot k)$.
- **Lý thuyết trò chơi:** Độ phức tạp $O(\sqrt{n})$.