ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Khoa Khoa học máy tính

Môn học Phân Tích Và Thiết Kế Thuật Toán

Vận dụng thiết kế thuật toán: Lý thuyết trò chơi

Cao Lê Công Thành
MSSV: 23521437 Đặng Quang Vinh
MSSV: 23521786

Mục lục

1 Bài tập 1

Phân tích và Phương pháp giải

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 $\frac{p}{2}$, 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:
        return False
    if p%2 == 1:                                  # Nếu p lẻ
        return not canWin(p-1) or not canWin(p+1)
    else:                                         # Nếu p chẵn
        return ¬ canWin(p // 2) =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 $\frac{p}{2}$.

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

```
Khởi tạo: dp = [-1] * 10^6 + 1
function canWin(p):
    if p == 0:
        return False
    if dp[p] != -1:
        return dp[p]
    if p%2 == 1:
        dp[p] = not canWin(p-1) or not canWin(p+1)
    else:
        dp[p] = not canWin(p // 2)
    return dp[p] =0
```

Mã giả:
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 $\frac{p}{2}$, do đó trạng thái của $\frac{p}{2}$ quyết định kết quả.

Mã giả:

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

```
function canWinTheory(p):
    while p > 0:
        if p%2 == 1:
            return True                          # A thắng nếu p lẻ
        p ← p//2
return False                                     # Nếu về 0, A thua =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 Bài tập 2

2.1 Phân loại bài toán

Đây là **trò chơi có tổng bằng 0**, vì:

- Hai người chơi lần lượt tham gia, không có yếu tố ngẫu nhiên hay bất đối xứng (mọi thông tin đều được biết trước).
- Trò chơi kết thúc khi $n = 0$, và người không thể thực hiện lượt bốc đồng xu sẽ thua.
- Kết quả chỉ phụ thuộc vào chiến lược của người chơi, không bị tác động bởi yếu tố bên ngoài.

2.2 Phương pháp giải

2.2.1 Trường hợp $n \leq 1000$

Sử dụng **quy hoạch động (QHD)** để kiểm tra trạng thái thắng/thua cho mỗi giá trị n:

- **Ý tưởng chính:** Sử dụng mảng $dp[i]$ để lưu trạng thái i:
  - $dp[i]$ = True: trạng thái thắng (người đến lượt có chiến lược thắng).
  - $dp[i]$ = False: trạng thái thua (người đến lượt sẽ thua nếu cả hai đều chơi tối ưu).
- Quy tắc chuyển trạng thái:

$$dp[i] = \text{True} \quad \text{nếu tồn tại } x \ (1 \ \text{x} \ \text{k}) \text{ sao cho } dp[i - x] = \text{False}.$$

  Ngược lại:

$$dp[i] = \text{False}.$$

- **Các bước thực hiện:**
  1. Khởi tạo: $dp[0]$ = False (không còn đồng xu để bốc là trạng thái thua).
  2. Tính $dp[i]$ cho $i = 1$ đến n:
     - Với mỗi k, kiểm tra trạng thái thắng/thua của $dp[n]$.
  3. Đếm số lượng k thỏa mãn $dp[n]$ = True.

2.2.2 Trường hợp $n \leq 10^{18}$

Với n lớn, không thể duyệt tuần tự từng trạng thái. Thay vào đó, áp dụng chiến thuật toán học:

- **Ý tưởng chính:** Nếu $n \mod (k+1) \neq 0$, A có chiến lược đảm bảo chiến thắng.
  - **Giải thích:**
    - Với k, trạng thái thua xảy ra khi $n \mod (k+1) = 0$.
    - Khi đó, bất kỳ số xu x ($1 \leq x \leq k$) mà A bốc ra sẽ khiến trạng thái trở thành trạng thái thắng cho B.
    - Ngược lại, nếu $n \mod (k+1) \neq 0$, A luôn có thể bốc số xu $x = n \mod (k+1)$ để đưa trò chơi về trạng thái thua cho B.
- **Các bước thực hiện:**
  1. Với mỗi k, kiểm tra $n \mod (k+1)$.
  2. Nếu $n \mod (k+1) \neq 0$, thì k thỏa mãn.
  3. Đếm số lượng k thỏa mãn.

2.3 Mã giả bài toán

2.3.1 Trường hợp $n \leq 1000$

```
function count_k_dp(n):
    result = 0
    for k in range(1, n + 1):  # duyet tung gia tri k
        dp = [False] * (n + 1)  # mang trang thai
        dp[0] = False  # trang thai thua
        for i in range(1, n + 1):
            for x in range(1, min(k, i) + 1):  # xet cac gia tri x <= k
                if not dp[i - x]:  # neu trang thai i - x la thua
                    dp[i] = True  # trang thai i la thang
                    break
        if dp[n]:  # neu trang thai n la thang
            result += 1
    return result
```

2.3.2 Trường hợp $n \leq 10^{18}$

```
function count_k_mod(n):
    result = 0
    for k in range(1, int(n ** 0.5) + 1):  # chi xet cac k nho
        if n % (k + 1) != 0:
            result += 1
    return result
```

2.4 Phân tích độ phức tạp

- **Trường hợp $n \leq 1000$:**
  - Duyệt k từ 1 đến n.
  - Mỗi k, tính trạng thái $dp[i]$ từ 1 đến n.
  - Độ phức tạp: $O(n^2)$.
- **Trường hợp $n \leq 10^{18}$:**
  - Tính toán $n \mod (k+1)$ cho k nhỏ hơn $\sqrt{n}$.
  - Độ phức tạp: $O(\sqrt{n})$.