Bài tập CS112: Quy hoạch động

Nhóm : 2
Nguyễn Thiện Nhân 23521083
Trần Vinh Khánh 23520726

Đề Bài 1

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: $p := \frac{p}{2}$.

Cho trước số nguyên dương p. Bạn A luôn đi trước 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).

Ý Tưởng Giải Quyết

Đây là một bài toán trò chơi lý thuyết, ta có thể sử dụng quy hoạch động (dynamic programming) hoặc đệ quy với memoization để giải quyết. Cụ thể:

- Nếu p là số chẵn, người chơi chỉ có thể chia p cho 2.
- Nếu p là số lẻ, người chơi có thể chọn giảm p xuống $p-1$ hoặc tăng p lên $p+1$.

Mỗi người chơi sẽ cố gắng đưa giá trị của p về 0 một cách nhanh nhất, và chúng ta cần xác định liệu A có luôn thắng nếu cả hai chơi tối ưu.

Ta sẽ xây dựng một mảng `dp`, trong đó `dp[p]` đại diện cho việc người chơi có thể thắng nếu bắt đầu với giá trị p.

Mã Giả

Giả sử ta có hàm `dp(p)` để xác định liệu A có thắng nếu bắt đầu với giá trị p hay không:

- `dp(p) = True` nếu người chơi có thể thắng khi bắt đầu với p.
- `dp(p) = False` nếu người chơi không thể thắng (tức là đối thủ sẽ thắng).

Các bước giải quyết:

- Nếu p là số chẵn, người chơi buộc phải giảm p xuống $p/2$.
- Nếu p là số lẻ, người chơi có thể chọn giảm p xuống $p-1$ hoặc tăng p lên $p+1$.

Giải quyết bài toán đối kháng

```
def dp(p):
    if p == 0:
        return False
    if p % 2 == 1:
        return not dp(p - 1) or dp(p + 1)
    else:
        return not dp(p / 2)
```

Giải Quyết Với Các Giới Hạn

0.1 Giới Hạn 1: $p \leq 10$

Với giới hạn này, ta có thể sử dụng đệ quy quay lui vét hết trường hợp

```
def can_win(p):
    if p == 0:
        return False  # Người chơi thua nếu p = 0
    if p % 2 == 0:
        return not can_win(p / 2)  # Nếu p chẵn, bốc p/2
    else:
        return not can_win(p - 1) or not can_win(p + 1)  # Nếu p lẻ, bốc p-1 hoặc p+1
```

Giới Hạn 2: $p \leq 10^6$

Với giới hạn $p \leq 10^6$, ta có thể sử dụng quy hoạch động, nhưng Ta có thể chỉ lưu trữ giá trị của dp cho các và tính toán lần lượt.

Quy hoạch động với giới hạn $p \leq 10^6$

```
def solve(p):
    dp = [False] * (10^6 + 1)

    for i in range(1, p + 1):
        if i % 2 == 1:
            dp[i] = not dp[i - 1] or (i + 1 <= 10^6 and not dp[i + 1])
        else:
            dp[i] = not dp[i / 2]

    if dp[p]:
        print("YES")
    else:
        print("NO")
```

Kết Luận

Với các phương pháp trên, chúng ta có thể giải quyết bài toán Đối khángvới các giới hạn khác nhau của giá trị p. Các giới hạn càng lớn, chúng ta càng cần phải tối ưu hóa bộ nhớ và thời gian tính toán thông qua các kỹ thuật như memoization.

Đề Bài 2

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, 2 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, $x \leq k$). Nếu không thực hiện bốc được thì bạn thực hiện lượt đó sẽ thua. 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.

Ý Tưởng Giải Quyết

Đây là bài toán trò chơi lý thuyết, có thể giải quyết bằng phương pháp quy hoạch động (dynamic programming). Cụ thể:

- Nếu còn n đồng xu, người chơi có thể bốc từ 1 đến k đồng xu trong mỗi lượt.

- Trạng thái thắng/thua được xác định theo các bước:
  - Nếu có ít nhất một cách bốc số đồng xu sao cho đối thủ thua, thì người chơi thắng.
  - Nếu không có cách nào khiến đối thủ thua, thì người chơi thua.

Mã Giả

Giả sử ta có hàm `solve(n, k)` để xác định số lượng giá trị k thỏa mãn điều kiện A luôn thắng:
Giải quyết bài toán trò chơi đồng xu

```
def solve(n, k):
    # Mảng dp[i] = True nếu với i đồng xu, người chơi có thể thắng
    dp = [False] * (n + 1)

    # Trạng thái ban đầu, nếu còn 0 đồng xu thì thua
    dp[0] = False

    # Tính toán cho từng số lượng đồng xu từ 1 đến n
    for i in range(1, n + 1):
        for j in range(1, min(k, i) + 1):
            if not dp[i - j]:
                dp[i] = True
                break

    # Đếm số k mà người chơi A có thể thắng
    count = 0
    for k_val in range(1, k + 1):
        if dp[n % (k_val + 1)]:
            count += 1

    return count
```

Giải Quyết Với Các Giới Hạn

Giới Hạn 1: $n \leq 1000$

Với giới hạn $n \leq 1000$, chúng ta có thể sử dụng quy hoạch động (DP) với mảng kích thước $n + 1$ để tính toán cho từng số lượng đồng xu từ 1 đến n.
Quy hoạch động với giới hạn $n \leq 1000$

```
def solve(n, k):
    # Mảng dp[i] = True nếu với i đồng xu, người chơi có thể thắng
    dp = [False] * (n + 1)

    # Trạng thái ban đầu, nếu còn 0 đồng xu thì thua
    dp[0] = False

    # Tính toán cho từng số lượng đồng xu từ 1 đến n
    for i in range(1, n + 1):
        for j in range(1, min(k, i) + 1):
            if not dp[i - j]:
                dp[i] = True
                break

    # Đếm số k mà người chơi A có thể thắng
```

```
    count = 0
    for k_val in range(1, k + 1):
        if dp[n % (k_val + 1)]:
            count += 1

    return count
```

Kết Luận

Bài toán Trò chơi đồng xucó thể được giải quyết hiệu quả bằng phương pháp quy hoạch động. Chúng ta đã trình bày cách tính toán số lượng giá trị k thỏa mãn yêu cầu A luôn thắng trong trò chơi.