

# 동적프로그래밍

practice 자료구조 · 2024. 10. 15. 18:43

동적 프로그래밍(DP)은 문제를 작은 부분 문제로 나누고, 각 부분 문제의 결과를 저장해 중복 계산을 피하는 알고리즘 기법입니다. **최적 부분 구조**와 **중복된 부분 문제**가 있는 문제에서 효과적입니다. 이를 푸는 방법은 두 가지입니다:

1. **탑다운 방식(메모이제이션)**: **재귀적으로** 문제를 풀고, 계산된 값을 저장해 중복 계산을 방지.
2. **바텀업 방식(테이블링)**: 작은 문제부터 해결해 큰 문제를 푸는 반복문 방식.

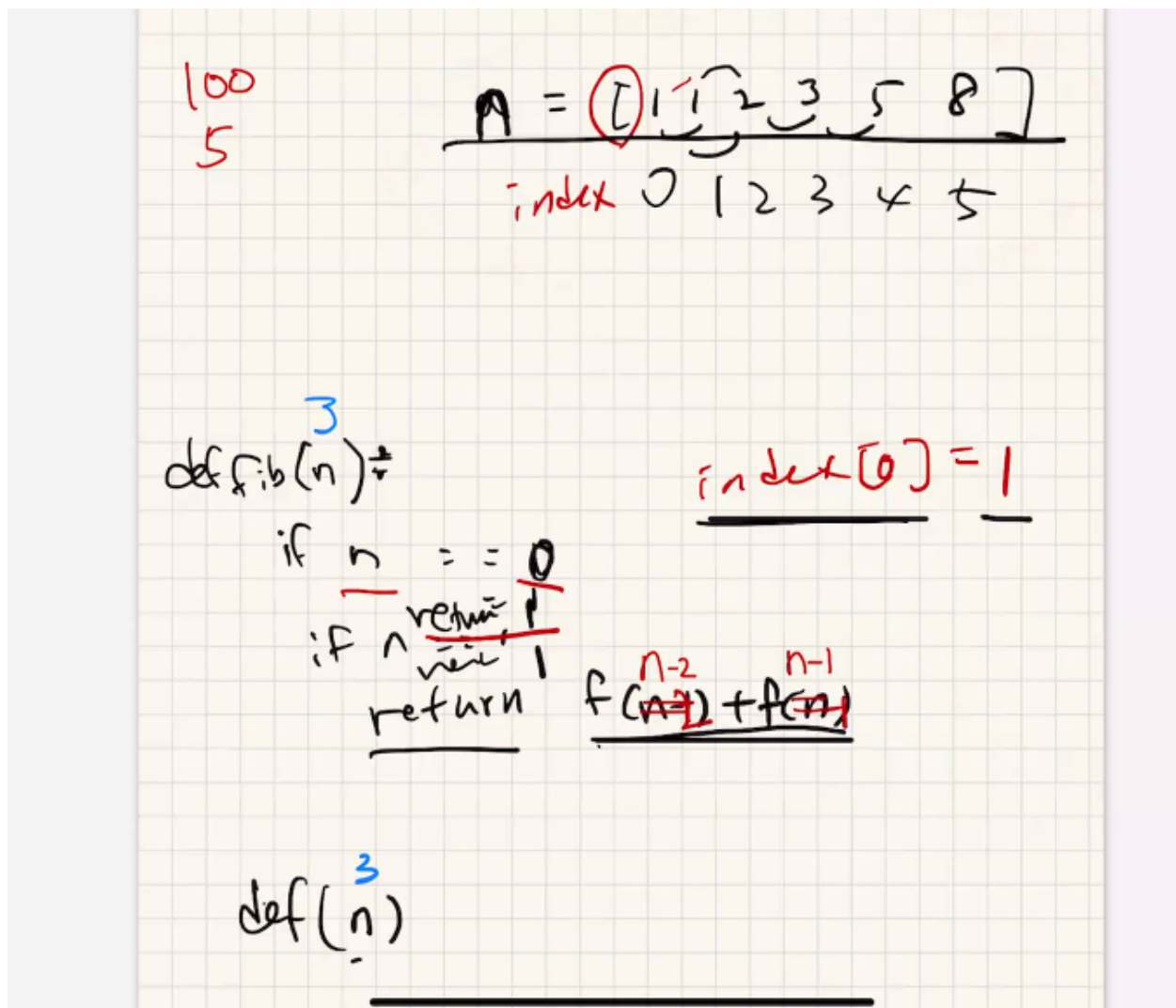
예시로는 피보나치 수열, 최단 경로 문제, 배낭 문제 등이 있으며, 중복된 계산을 피하고 효율성을 높입니다.

## 피보나치수열

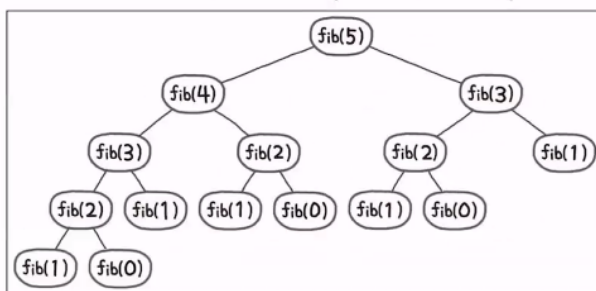
- 피보나치 수열의 정의를 바탕으로, 단순한 재귀 (Recursive) 알고리즘을 짤 수 있음

0 1 1 2 3 5 8 13 21  
n = 0 1 2 3 4 5 6 7 8

- ```
def fib(n):  
    if n == 0:  
        return 0  
    elif n == 1:  
        return 1  
    else:  
        return fib(n-1) + fib(n-2)
```



- 하지만 위 알고리즘은 매우 느림 (몇시간 이상 소요)
- 이는 많은 중복된 (불필요한) 계산을 하기 때문





1 2 3 4 5

- 위의 예시에서, fib(0)은 3번, fib(1)은 4번, fib(2)는 3번, 호출
- 이러한 중복 계산의 수는 주어진 n에 따라 기하급수적으로 증가

• 따라서, 이미 계산한 **값들을 적어 놓고 재사용한다면 (Memoization)**, 효율을 크게 올릴 수 있을 것

```
def dyn_fib(n):
    F = list()
    F[0] = 0
    F[1] = 1

    for i in range(2, n+1):
        F[i] = F[i-1] + F[i-2]
    return F[n]
```

♡ 공감  

구독하기

'practice' 자료구조 카테고리의 다른 글

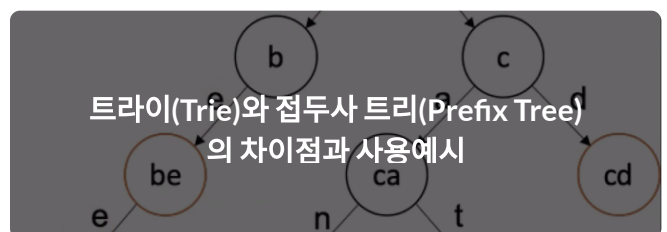
|                                                                             |            |
|-----------------------------------------------------------------------------|------------|
| <a href="#">KMP 문자열 검색 알고리즘 (0)</a>                                         | 2024.10.14 |
| <a href="#">트라이(Trie)와 접두사 트리(Prefix Tree)의 차이점과 사용예시 (0)</a>               | 2024.10.14 |
| <a href="#">벨만-포드(Bellman-Ford) 알고리즘과 다익스트라(Dijkstra) 알고리즘의 차이점과 사용 (0)</a> | 2024.10.14 |
| <a href="#">(Red-Black Tree)의 기본 속성과 그 왜곡 방지 방법 (0)</a>                     | 2024.10.14 |
| <a href="#">B- / B+ 트리 차이점 (0)</a>                                          | 2024.10.14 |

관련글

[관련글 더보기](#)

|   |   |   |
|---|---|---|
| 0 | A | 0 |
| 1 | B | 0 |
| 2 | A | 1 |
| 3 | B | 2 |
| 4 | A | 1 |
| 5 | A | 1 |
| 6 | B | 2 |
| 7 | A | 3 |

KMP 문자열 검색 알고리즘



트라이(Trie)와 접두사 트리(Prefix Tree)의 차이점과 사용예시

| 특징     | 벨만-포드 알고리즘                     | 다익스트라 알고리즘              |
|--------|--------------------------------|-------------------------|
| 시간 복잡도 | $O(V \times E)$                | $O((V + E) \log V)$     |
| 음수 가중치 | 지원                             | 지원하지 않음                 |
| 음수 사이클 | 발견 가능                          | 발견 불가                   |
| 사용 사례  | 음수 가중치가 있는 그래프, 음수 사이클 탐지 필요 시 | 양수 가중치만 있는 그래프, 대규모 그래프 |
| 작동 방식  | 동적 프로그래밍 방식으로 모든 간선을 반복 확인     | 탐욕적 방식으로 가까운 노드부터 탐색    |

벨만-포드(Bellman-Ford) 알고리즘과 다익스트라(Dijkstra) 알고리즘의 차이점...

(Red-Black Tree)의 기본 속성과 그 왜곡 방지 방법

## 자연어(NLP)

네이쳐2024 님의 블로그입니다.



구독하기 +

댓글 0



이름

비밀번호

내용을 입력하세요.



댓글  
등록