파이썬으로 배우는 알고리즘 기초

Chap 3. 동적계획

3.1

동적계획과 이항계수

- **동적계획법: Dynamic Programming**
  - 문제를 더 작은 문제로 분할하되, 상향식으로 문제를 해결한다.
  - 1953년, Richard Bellman 교수가 제안
  - Programming: 여기서는 '계획'을 의미
    - TV프로그램. 오늘 행사의 프로그램 안내.
  - Memoization: 메모이제이션
    - 가장 작은 입력 사례의 해답을 테이블에 저장하고 필요할 때 꺼내 쓴다.

- **동적계획법으로 문제 풀기**
  1. 문제를 해결할 수 있는 **재귀 관계식**을 구한다.
  2. 가장 작은 입력사례로부터 **상향식 방법**으로 문제를 해결한다.

- **분할정복법 .vs. 동적계획법**
  - 문제를 작은 사례로 분할하여 해결한다는 점에서 동일
  - 분할정복: 재귀 호출을 통해 분할하여 정복 (Top-Down)
  - 동적계획: 메모이제이션을 통해 상향식으로 정복 (Bottom-Up)
  - ex) 피보나치 수열

3.1 이항 계수

- **이항 계수 문제**
  - 이항 계수의 정의
    - $\binom{n}{k} = \frac{n!}{k!(n-k)!}$, for $0 \le k \le n$.
    - 문제점: $n!$, $k!$의 값은 매우 크기 때문에 계산이 어렵다.
  - 이항 계수의 재귀적 정의: 분할정복(Divide-and-Conquer)
    - $$\binom{n}{k} = \begin{cases} \binom{n-1}{k-1} + \binom{n-1}{k} & 0 < k < n \\ 1 & k = 0 \text{ or } k = n \end{cases}$$

3.1 이항 계수

- **이항 계수 문제**
  - 이항 계수의 정의
    - $\binom{n}{k} = \frac{n!}{k!(n-k)!}$, for $0 \le k \le n$.
    - 문제점: $n!$, $k!$의 값은 매우 크기 때문에 계산이 어렵다.
  - 이항 계수의 재귀적 정의: 분할정복(Divide-and-Conquer)
    - $\binom{n}{k} = \begin{cases} \binom{n-1}{k-1} + \binom{n-1}{k} & 0 < k < n \\ 1 & k = 0 \text{ or } k = n \end{cases}$

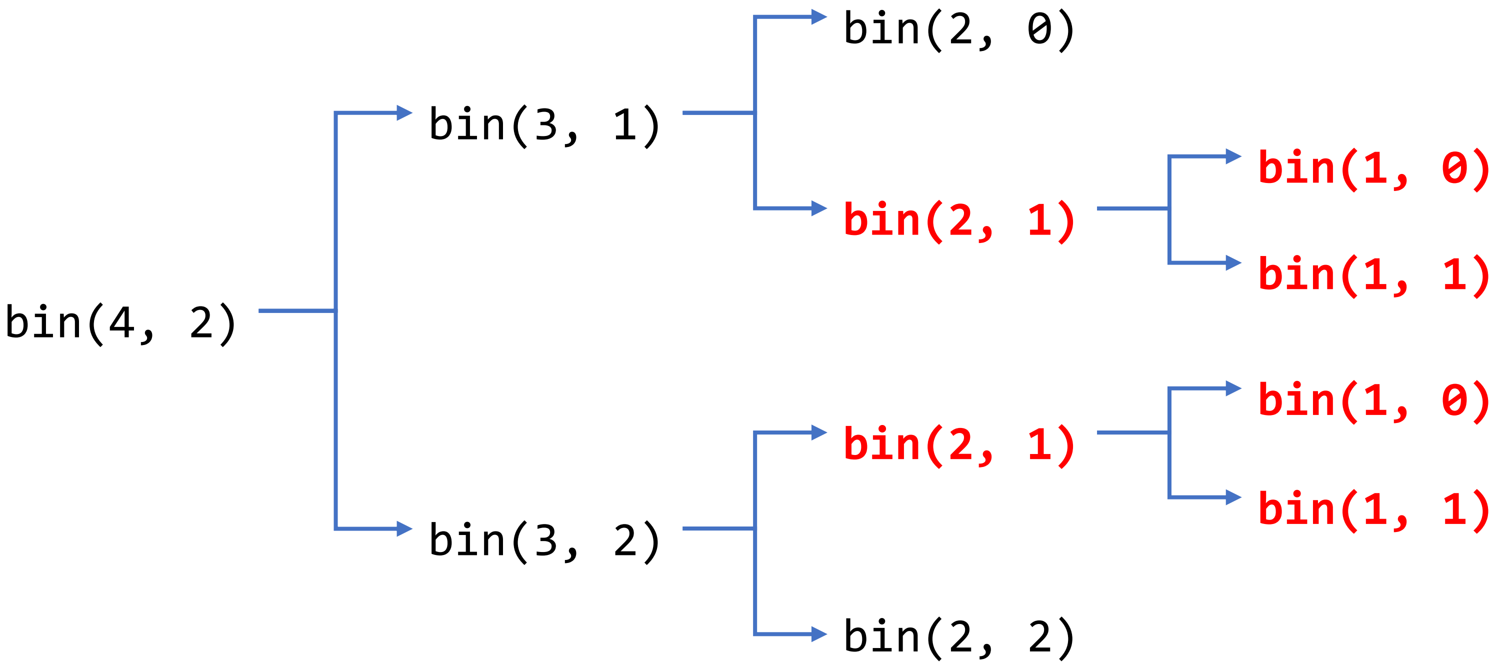
Algorithm 3.1: Binomial Coefficient (Divide-and-Conquer)

```
def bin (n, k):
    if (k == 0 or n == k):
        return 1
    else:
        return bin(n - 1, k - 1) + bin(n - 1, k)
```

```
for n in range(10):
    for k in range(n + 1):
        print(bin(n, k), end = " ")
    print()
```

3.1 이항 계수

- Algorithm 3.1의 문제점
  - 피보나치 항 구하기의 재귀적(*recursive*) 방법과 같은 문제
  - 중복 호출을 없앨 수 있는 방법은? 반복적(*iterative*) 방법



3.1 이항 계수

- 이항 계수의 성질: **파스칼의 삼각형**

1	${}_0C_0$
1 1	${}_1C_0$ ${}_1C_1$
1 2 1	${}_2C_0$ ${}_2C_1$ ${}_2C_2$
1 3 3 1	${}_3C_0$ ${}_3C_1$ ${}_3C_2$ ${}_3C_3$
1 4 6 4 1	${}_4C_0$ ${}_4C_1$ ${}_4C_2$ ${}_4C_3$ ${}_4C_4$
1 5 10 10 5 1	${}_5C_0$ ${}_5C_1$ ${}_5C_2$ ${}_5C_3$ ${}_5C_4$ ${}_5C_5$

3.1 이항 계수

- 이항 계수 구하기: 동적계획(Dynamic Programming)
  - 1단계: 재귀 관계식을 찾는다.
    - 이항 계수의 재귀적 정의를 찾았다.
    - $B[i][j] = \begin{cases} B[i-1][j-1] + B[i-1][j] & 0 < j < i \\ 1 & j = 0 \ or \ j = i \end{cases}$
  - 2단계: 상향식 방법으로 해답을 구한다.
    - 파스칼의 삼각형이 가진 특성을 이용한다.
    - $B[i][j] = 1, j = 0 \ or \ j = 1$
    - $B[i][j] = B[i-1][j-1] + B[i-1][j], 0 < j < i$

3.1 이항 계수

$j \leq k$

B[i][j]	0	1	2	3	4
0	1				
1	1	1			
2	1	2	1		
3	1	3	3	1	
4	1	4	6	4	1

$i \leq n$

계산 불필요($k \leq n$)

3.1 이항 계수

Algorithm 3.2: Binomial Coefficient (Dynamic Programming)

```python
def bin2 (n, k):
    B = [[0] * (k + 1) for _ in range(n + 1)]
    for i in range(n + 1):
        for j in range(min(i, k) + 1):
            if (j == 0 or j == i):
                B[i][j] = 1
            else:
                B[i][j] = B[i - 1][j - 1] + B[i - 1][j]
    return B[n][k]
```

3.1 이항 계수


```python
for n in range(10):
    for k in range(n + 1):
        print(bin2(n, k), end = " ")
    print()
```

- **이항 계수의 시간 복잡도와 성능 개선**
  - Algorithm 3.2(D.P)는 Algorithm 3.1(D&C)**보다 훨씬 효율적**
    - D&C**의 시간 복잡도** $\in \Theta\left(\binom{n}{k}\right)$, D.P.**의 시간 복잡도** $\in \Theta(nk)$

▪ 연습문제 3.4: 효율적인 이항계수 계산

- 다음 성질을 이용하면 성능을 더 개선할 수 있다.
  - $\binom{n}{k} = \binom{n}{n-k}$: k가 $n/2$보다 클 경우에 적용
- 2차원 리스트(배열)를 사용할 필요가 있는가?
  - 1차원 리스트(배열)만으로도 구현이 가능하다.

3.1 이항 계수


```python
def bin3 (n, k):
    if (k > n // 2):
        k = n - k
    B = [0] * (k + 1)
    B[0] = 1
    for i in range(1, n + 1):
        j = min(i, k)
        while (j > 0):
            B[j] = B[j] + B[j - 1]
            j -= 1
    return B[k]
```

3.1 이항 계수


```python
for n in range(10):
    for k in range(n + 1):
        print(bin3(n, k), end=" ")
    print()

print(bin3(9, 5))
```

주니온TV@Youtube

자세히 보면 유익한 코딩 채널

https://bit.ly/2JXXGqz

- 여러분의 구독과 좋아요는 강의제작에 큰 힘이 됩니다.
- 강의자료 및 소스코드: 구글 드라이브에서 다운로드 (다운로드 주소는 영상 하단 설명란 참고)

  https://bit.ly/3fN0q8t