파이썬으로 배우는 알고리즘 기초

Chap 3. 동적계획

3.5 최적 이진검색트리

3.5 최적 이진검색트리

주니온TV@Youtube
자세히 보면 유익한 코딩 채널

- **최적 이진검색트리 문제**
  - 주어진 n개의 키로 최적 이진검색트리를 구하시오.
  - 엄밀한 문제 정의
    - 주어진 n개의 키: $K_1, K_2, \cdots, K_n$
    - 각 키의 검색 확률 p_i: 전체 검색 횟수 중에서 K_i를 검색하는 확률
    - 각 키의 비교 횟수 c_i: K_i를 검색하는 데 필요한 키의 비교 횟수
    - 각 키의 평균 검색 비용(시간): 검색 확률 × 비교 횟수 $(p_i \times c_i)$
    - 전체 키의 평균 검색 비용(시간): $T_{avg} = \sum_{i=1}^{n} p_i c_i$
  - 최적 이진검색트리 문제는 **최적화 문제**
    - 전체 키의 평균 검색 비용을 최소화하는 이진검색트리 찾기

3.5 최적 이진검색트리

주니온TV@Youtube
자세히 보면 유익한 코딩 채널

- **이진검색트리** (BST: Binary Search Tree)
  - 다음의 조건들을 모두 만족하는 이진트리
    - 각 노드는 하나의 **유일한 키**를 가지고 있다
    - 모든 노드가 가진 키의 값은 그 노드의 왼쪽 서브트리의 키의 값보다 크다
    - 모든 노드가 가진 키의 값은 그 노드의 오른쪽 서브트리의 키의 값보다 작다

3.5 최적 이진검색트리

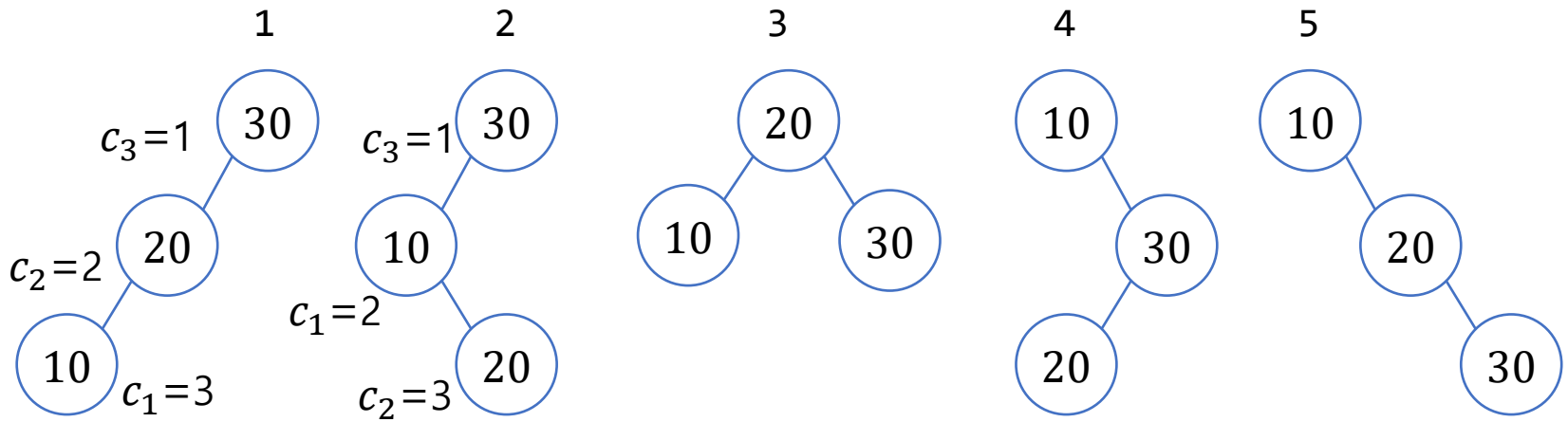
주니온TV@Youtube
자세히 보면 유익한 코딩 채널

- **최적 이진검색트리**: 단순무식하게 풀기(Brute-Force Approach)
  - 모든 경우의 수에 대해서 계산해 보고 최적의 이진트리 선택
  - 이진검색트리의 가능한 경우의 수는?
    - **카탈란 수**: $C(n) = \frac{1}{n+1}\binom{2n}{n} \sim \frac{4^n}{n^{3/2}\sqrt{\pi}}$
    - n개의 키로 만들 수 있는 이진 트리의 수 = $C(n)$

3.5 최적 이진검색트리

주니온TV@Youtube
자세히 보면 유익한 코딩 채널

- **최적 이진검색트리의 입력 사례**
  - $n = 3, K=[10, 20, 30], p = [0.7, 0.2, 0.1]$



$T_1=2.6$ $T_2=2.1$ $T_3=$ $T_4=$ $T_5=1.4$

(Optimal BST)

3.5 최적 이진검색트리

주니온TV@Youtube
자세히 보면 유익한 코딩 채널

- 최적 이진검색트리: 동적계획(Dynamic Programming)
  - 1단계: 재귀 관계식을 찾는다.
    - A: 이진검색트리를 만드는데 최적 검색비용의 행렬
    - $A[i][j]$: K_i에서 K_j까지 이진검색트리를 만드는데 최적 검색 비용
    - 목표: $K_i \cdots K_j$를 $(K_i \cdots K_{k-1})K_k(K_{k+1} \cdots K_j)$로 분할하는 재귀 관계식 찾기
  - 2단계: 상향식 방법으로 해답을 구한다.
    - 초기화: $A[i][i] = p_i$ (주대각선을 p_i 으로)
    - 최종 목표:$A[1][n]$.
    - 상향식 계산: 대각선 1번, 대각선 2번, $\cdots$, 대각선 $n-1$번

3.5 최적 이진검색트리

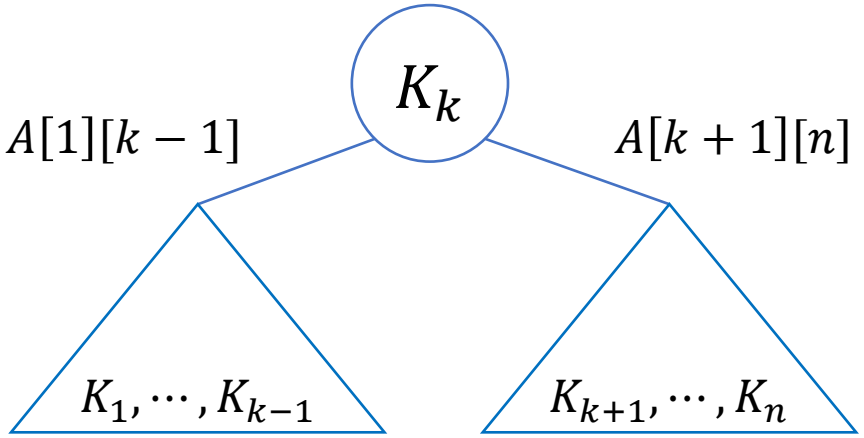
주니온TV@Youtube
자세히 보면 유익한 코딩 채널

- 최적 이진검색트리의 재귀 관계식 구하기
  - 트리 k: 키 K_k가 루트 노드에 있는 최적 이진검색트리
  - 키 비교 횟수: 서브 트리의 비교 횟수에 루트에서 비교 한 번 추가
  - $m \neq k$인 K_m에 대해서 트리 k에 K_m을 놓기 위한 비교 한 번 추가
    - K_m의 평균 검색비용에 p_m을 추가
    - $A[1][k-1]+A[k+1][n]+\sum_{m=1}^{n} p_m$
  - 최적 트리: k개의 트리 중 평균 검색비용이 가장 작은 트리
    - $A[1][n] = \underset{i \le k \le j}{\text{minimum}}(A[1][k-1]+A[k+1][n]+\sum_{m=1}^{n} p_m)$

3.5 최적 이진검색트리

주니온TV@Youtube
자세히 보면 유익한 코딩 채널

- **최적 이진검색트리의 재귀 관계식**
  - $A[i][i] = p_i$
  - $A[i][i-1] = 0$
  - $A[j+1][j] = 0$
  - $\boldsymbol{A[i][j] = \underset{i \le k \le j}{\text{minimum}}(A[i][k-1] + A[k+1][j]) + \sum_{m=i}^{j} p_m}, i < j$



주니온TV@Youtube
자세히 보면 유익한 코딩 채널

Algorithm 3.9: Optimal Binary Search Tree

```
def optsearchtree (p):
    n = len(p) - 1
    A = [[-1] * (n + 1) for _ in range(n + 2)]
    R = [[-1] * (n + 1) for _ in range(n + 2)]
    for i in range(1, n + 1):
        A[i][i - 1] = 0
        A[i][i] = p[i]
        R[i][i - 1] = 0
        R[i][i] = i
    A[n + 1][n] = 0
    R[n + 1][n] = 0
    for diagonal in range(1, n):
        for i in range(1, n - diagonal + 1):
            j = i + diagonal
            A[i][j], R[i][j] = minimum(A, p, i, j)
    return A, R
```

주니온TV@Youtube
자세히 보면 유익한 코딩 채널

Algorithm 3.9: Optimal Binary Search Tree

```
def minimum (A, p, i, j):
    minValue = INF
    minK = 0
    for k in range(i, j + 1):
        value = A[i][k - 1] + A[k + 1][j]
        for m in range(i, j + 1):
            value += p[m]
        if (minValue > value):
            minValue = value
            minK = k
    return minValue, minK
```

3.5 최적 이진검색트리


```
INF = 999
keys = [0, 10, 20, 30, 40, 50]
p = [0, 35, 12, 22, 8, 15]
A, R = optsearchtree(p)
print('A = ')
for i in range(1, len(A)):
    print(A[i])
print('R = ')
for i in range(1, len(R)):
    print(R[i])
```

```
A =
[0, 35, 59, 115, 139, 182]
[-1, 0, 12, 46, 62, 100]
[-1, -1, 0, 22, 38, 76]
[-1, -1, -1, 0, 8, 31]
[-1, -1, -1, -1, 0, 15]
[-1, -1, -1, -1, -1, 0]
R =
[0, 1, 1, 1, 1, 3]
[-1, 0, 2, 3, 3, 3]
[-1, -1, 0, 3, 3, 3]
[-1, -1, -1, 0, 4, 5]
[-1, -1, -1, -1, 0, 5]
[-1, -1, -1, -1, -1, 0]
```

3.5 최적 이진검색트리

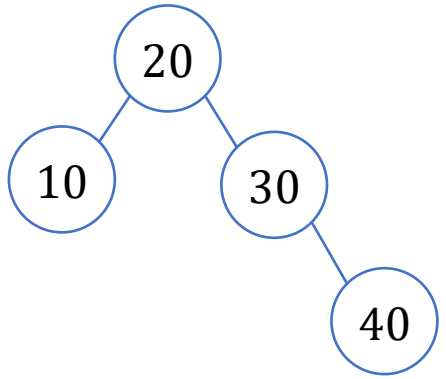
주니온TV@Youtube
자세히 보면 유익한 코딩 채널

- **최적 이진검색트리 구하기**
  - $R[i][j]$: i번째에서 j번째까지 최적 트리의 루트 노드 인덱스
  - **재귀 호출을 통한 분할 정복**

- n =4, K = K_1, K_2, K_3, K_4=[10, 20, 30, 40], $p = p_1, p_2, p_3, p_4$=[3, 3, 1, 1].

A	1	2	3	4
1				
2				
3				
4				

R	1	2	3	4
1				
2				
3				
4				



주니온TV@Youtube
자세히 보면 유익한 코딩 채널

Algorithm 3.10: Build Optimal Binary Search Tree

```
def tree (R, i, j):
    k = R[i][j]
    if (k == 0):
        return None
    else:
        node = BSTNode(keys[k])
        node.left = tree(R, i, k - 1)
        node.right = tree(R, k + 1, j)
        return node
```

Algorithm 3.10: Build Optimal Binary Search Tree

```
class BSTNode:
    def __init__ (self, key):
        self.key = key
        self.left = None
        self.right = None

def preorder (node):
    if (node is None):
        return
    else:
        print(node.key, end = ' ')
        preorder(node.left)
        preorder(node.right)
```

```
def inorder (node):
    if (node is None):
        return
    else:
        inorder(node.left)
        print(node.key, end = ' ')
        inorder(node.right)
```

3.5 최적 이진검색트리

주니온TV@Youtube
자세히 보면 유익한 코딩 채널

```
root = tree(R, 1, len(p) - 1)
print('inorder: ', end = ' ')
inorder(root)
print('\npreorder: ', end = ' ')
preorder(root)
```

```
inorder:  10 20 30 40 50
preorder:  30 10 20 50 40
```

주니온TV@Youtube

자세히 보면 유익한 코딩 채널

https://bit.ly/2JXXGqz

- 여러분의 **구독**과 **좋아요**는 강의제작에 큰 힘이 됩니다.
- 강의자료 및 소스코드: **구글 드라이브**에서 다운로드
  (다운로드 주소는 영상 하단 설명란 참고)

  https://bit.ly/3fN0q8t