Algorithm Assignment #5

채진석 교수님

201702797 채승민

순차 탐색 알고리즘

ADL

sequentialSearch(a[], search\_key, n)

i ← 0;

|  |
| --- |
| While (i < n and a[i].key != search\_key) do  i <- i + 1;  if (i == n) then return -1;  Else return i; |

end sequentialSearch()

N 값의 변화에 따른 시간복잡도 Test

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

이진 탐색 알고리즘

ADL

binarySearch(a[], search\_key, n) ;

|  |
| --- |
| r <- len(a) – 1;  l <- 0;  while (l <= r) do {  mid <- (l + r) // 2;  if (search\_key == a[mid].key)  then return mid;  else if (search\_key < a[mid].key)  then r <- mid – 1;  else l <- mid + 1;  } |

return -1;

end binarySearch()

N 값의 변화에 따른 시간복잡도 Test

텍스트, 스크린샷, 폰트, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

이진 트리 탐색 알고리즘

ADL

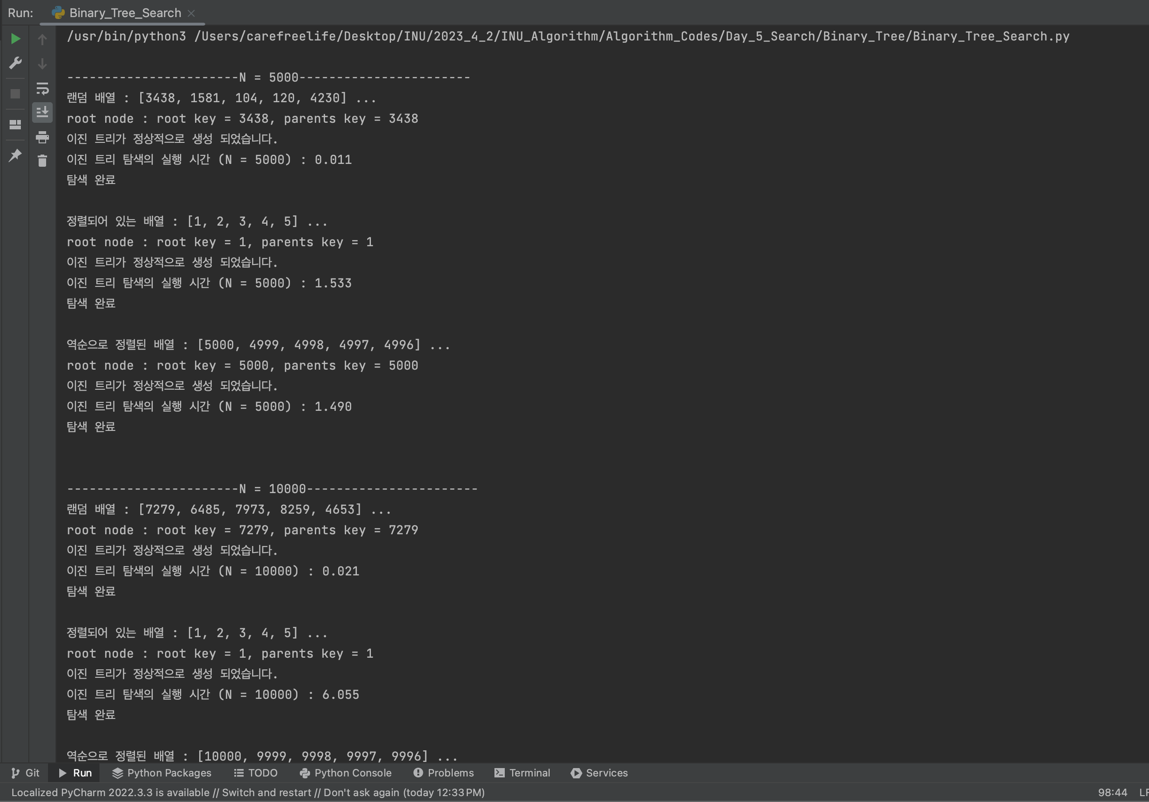
binaryTreeSearch(T, search\_key)

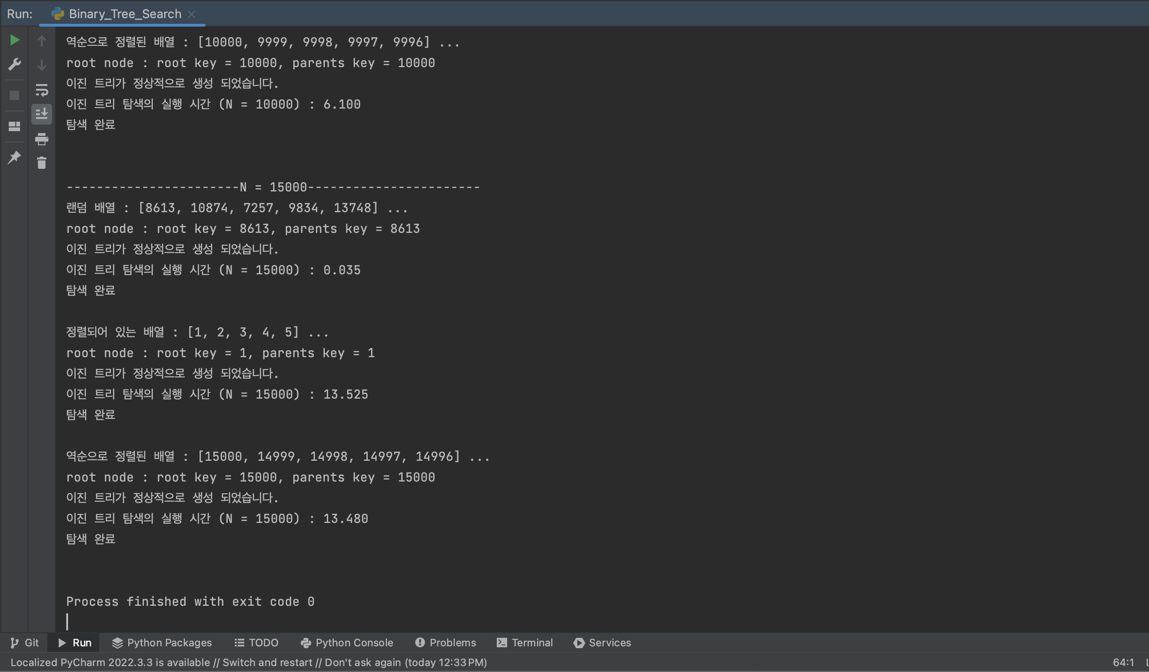
|  |
| --- |
| x = T;  parents <- x;  while (x != z) do {  rs <- {x, parents};  if (x.key <- search\_key) then return rs;  else if (x.key < search\_key) then {  parents <- x  x <- x.right  }  Else {  parents <- x  x <- x.left  }  }  return -1 |

end binaryTreeSearch()

N 값의 변화에 따른 시간복잡도 Test

N 값에 따른 최선 및 최악 실행 속도 분석





최선의 경우, 랜덤 Data set 이 입력되어 비교적 균형적인 이진 트리가 생성되는 경우 O(logN) 시간복잡도를 가져 상당히 빠른 탐색속도를 보여줌.

최악의 경우, 오름/내림차 순으로 정렬된 편향 트리가 생성되어 O(N) 의 시간 복잡도를 가져 순차적 탐색과 비슷하게 느린 탐색 속도를 보임.

이진 탐색 트리에 대한 정확성 검사

def check(self, search\_key):  
 rs = self.search(search\_key)  
 if rs[0] == rs[1]:  
 print(f"root node : root key = {rs[0].key}, parents key = {rs[1].key}")  
 return True  
 if rs[0] == rs[1].left:  
 return rs[0].key <= rs[1].key  
 elif rs[0] == rs[1].right:  
 return rs[0].key >= rs[1].key  
 else:  
 print("이진 트리 Error 발생")  
 return False

1. search() 의 결과로 해당 노드와 그 부모 노드를 Tuple 로서 반환.

* rs(child node, parent node)

1. rs[0] 과 rs[1]이 같다면 Root Node 이므로 True.
2. rs[0] 이 rs[1] 의 좌/우 자식인지 확인하고 해당 위치에 따라 자식노드의 값과 부모노드의 값을 비교하여 적절한지 분석 하여 Boolean 값 반환

# main 함수

# 이진 탐색 트리 정확성 검사  
for i in range(N):  
 if not d.check(key[i]):  
 print(f"이진 트리가 잘못 생성되어 종료합니다. key = {key[i]}")  
 sys.exit(1)

Coding Test Pracice #3

**□ 최소값 찾기**

[문제]

N개의 수 A1, A2, ..., AN과 L이 주어진다.

Di = Ai-L+1 ~ Ai 중의 최소값이라고 할 때, D에 저장된 수를 출력하는 프로그램을 작성하시오. 이때, i ≤ 0 인 Ai는 무시하고 D를 구해야 한다.

[입력]

- 첫째 줄에 N과 L이 주어진다. (1 ≤ L ≤ N ≤ 5,000,000)

- 둘째 줄에는 N개의 수 Ai가 주어진다. (-109 ≤ Ai ≤ 109)

[출력]

첫째 줄에 Di를 공백으로 구분하여 순서대로 출력한다.

import sys  
  
  
# L // i == 0 인 경우  
# 아직 (-) Index 가 존재함. -> L % i 연산 결과 까지의 i 중 최소값 찾기.  
#  
# L // i > 0 인 경우부터는 Index 가 0부터 맞춰졌으므로 i - L 부터 i 까지의 원소 중 최소값 반환.  
def find\_minimum(arr, n, l):  
 # print(f"목표 배열에서 L({l}) 범위 index에 존재하는 최소값은 다음과 같습니다.")  
 rs = []  
 for i in range(1, n + 1):  
 if l > i:  
 temp\_arr = arr[:i]  
 # print(f"arr[:{i}] = {temp\_arr}")  
 temp\_arr.sort()  
 # rs.append(temp\_arr[0])  
 else:  
 temp\_arr = arr[i - l:i]  
 # print(f"arr[{i-l}:{i}] = {temp\_arr}")  
 temp\_arr.sort()  
 rs.append(temp\_arr[0])  
 output = ' '.join(map(str, rs))  
 # print(output)  
 sys.stdout.write(output)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 while True:  
 # num = input("N, L 을 입력하시오 (1 ≤ L ≤ N ≤ 5,000,000): ").split()  
 num = input().split()  
 N, L = int(num[0]), int(num[1])  
 if 1 <= L <= N <= 5000000:  
 break  
 # print("조건이 맞지 않습니다. 다시 입력하세요. (1 ≤ L ≤ N ≤ 5,000,000)")  
 # N = 12  
 # L = 3  
 # a = [1, 5, 2, 3, 6, 4, 5, 7, 3, 5, 2, 6]  
 # print(f"N = {N} 개의 정수를 입력하세요: ")  
 a = input().split()  
 for i in range(N):  
 a[i] = int(a[i])  
 # print(f"a = {a}")  
 # for i in range(N):  
 # a.append(int(input()))  
 # print(f"초기 Array = {a}")  
 find\_minimum(a, N, L)

2023.10.18 (수)

201702797 채승민