Report 1 - Algorithm analysis

소프트웨어학부 2021041017 김규현

Main task: Theoretical analysis of the Binary Search algorithm and Big-O notation.

Task Code:

```
int binsearch(int list[], int sum, int left, int right) {
    int mid;
    while (left <= right) {
        mid = (left + right) / 2;
        switch(comp(list[mid], sum)) {
            case -1: left = mid + 1;        // mid < snum
                break;
            case 0: return mid;             // mid == snum
            case 1: right = mid - 1;        // mid > snum
        }
        return -1;
    }
}
```

(Lecture 3 - Algorithm analysis (Part. 1) P. 49)

본론에 앞서, 이진 탐색 트리란?

이진 탐색(Binary Search) : 정렬된 배열에서 특정 값을 찾기 위한 효율적인 탐색 알고리즘

배열의 중간 요소와 탐색하려는 값을 비교하여 원하는 값을 O(n^2) 보다 빠르게 찾을 수 있다.

이러한 방식은 이 알고리즘이 탐색할 범위를 절반씩 줄이기 때문에 빠르게 찾을 수 있다.

코드 해석&분석

1. 배열의 중간 요소 선택
2. 중간 요소와 찾으려는 값을 비교 (switch 문 사용)

   case -1: 찾으려는 값이 중간 요소보다 **크면**, 배열의 오른쪽 절반에서 탐색을 계속함.

   case 1: 찾으려는 값이 중간 요소보다 **작으면**, 배열의 왼쪽 절반에서 탐색을 계속함.

   case 0: 중간 요소가 찾으려는 값과 **같으면** 탐색이 완료됨.
3. 배열의 크기가 0이 될 때까지 이 과정을 반복 (while 문 사용).

주어진 코드의 시간 복잡도 계산

Commands	s/e	Freq.	Steps
int binsearch(int list[], int sum, int left, int right) {			
int mid;	1	1	1
while (left <= right) {			
mid = (left + right) / 2;	1	Log n	Log n
switch(comp(list[mid], sum)) {	1	Log n	Log n
case -1: left = mid + 1;			
break;			
case 0: return mid;			
case 1: right = mid - 1;			
}			
return -1;	1	1	1
}			
}			
Total	2 log n + 2		

steps / execution (s/e) : steps for the command lin

Frequency : Number of iterations of the command line

Total number of steps : s/e * frequency

Total -> $Tsum(n)$ = 2 log n + 2

결과 도출:

Big-O 표기법 수학적 정의 : 모든 n ≥ n0 ≥ 0에 대하여, 0 ≤ f(n) ≤ c * g(n)이 성립하는 양의 상수 c와 n0가 존재하면, **f(n) = O(g(n))**이다. → g(n)이 f(n)의 **점근적 상한**(upper bound)인가?

Big-O 표기법 특징 : **worst case**의 Time complexity를 표기한다. 또한, 상수항과 영향력이 없는 항을 무시하고 **가장 최고항**만 고려한다.

(Explain In detail...)

$\boldsymbol{f(n)}$ = **O**($\boldsymbol{g(n)}$) ??? (이때, $f(n)$ = 2 log n + 2, $g(n)$ = log n, c는 임의의 상수)

$\boldsymbol{f(n)}$ 에서 log n에 곱해져 있는 상수 계수 2는 큰 영향이 없으므로, **무시할 수 있다**.

마찬가지로 상수 항 2의 경우는 n의 값에 관계없이 일정한 값이므로, **무시할 수 있다**.

즉, 가장 지배적인 성분은 **'log n'**이라는 것을 알 수 있다.

또한, $\boldsymbol{f(n)}$과 $\boldsymbol{c * g(n)}$의 교차점 n0 이상의 모든 n에 대해, $f(n)$은 $\boldsymbol{c * g(n)}$보다 절대로 커질 수 없다. 따라서, $\boldsymbol{g(n)}$은 $\boldsymbol{f(n)}$의 **점근적 상한**(upper bound)이다.

∴ $\boldsymbol{f(n)}$ = **O**$\boldsymbol{(g(n))}$ = $\boldsymbol{O(log\ n)}$

결론:

Time Complexity of Task code in **Big-O** asymptotic notation**:** $\boldsymbol{O(log\ n)}$