141019 김연우



****

QuickSort의 T(n) = T(n1) + T(n2) + c\*n 이다. 이 때 감소 순 정렬되어있는 배열에 QuickSort를 정렬하면 pivot이 맨 마지막 원소라고 가정하는 경우 n1(pivot보다 작은 수 개수) = 0 이고 n2(pivot보다 큰 수 개수) = n - 1이된다. 이 경우 T(n) = T(n-1) + c\*n이 성립된다. 그리고 모든 원소가 pivot보다 크기 때문에 endofLowBlock은 start – 1이며, 반복 문 내부에서 어떤 원소도 위치가 바뀌지 않는다. 마지막에 pivot이 endofLowBlock 바로 뒤 원소와 스왑되므로, 다음 피봇은 가장 큰 원소가 선택된다.

이제 T(n-1)을 구하면 T(n)을 구할 수 있다. 이번에 pivot이 모든 원소보다 크기 때문에 n1 = n-2이고 n2 = 0 이다. 따라서 T(n-1) = T(n-2) + c\*n이 성립한다. 이 경우 모든 반복 문에서 if문 내부를 수행한다. endofLowBlock이 posTobeCheck와 함께 계속해서 증가하므로 반복 문 내부에서 변경되는 원소는 없다. 그리고 posTobeCheck가 end-1이 되어 루프를 종료할 때 endofLowBlock은 end -2 가 되며 결국 pivot은 end-1번째 원소 즉, 남은 원소중 가장 작은 원소와 스왑 된다. 다음 피봇은 n-2개의 원소중에 가장 작은 원소가 되며 모든 n-3개의 원소가 내림차순으로 정렬되어있기 때문에 재귀적으로 T(n-3) = T(n-4) + c\*n이 성립한다. 이렇게 반복해서 T(1)이 될 때 까지 QuickSort가 수행되므로, 임의의 자연수 n에 대하여 T(n) = T(n-1) + c\*n 이 성립된다. 결국 worst case와 같은 점화 식으로 의 수행시간이 보장된다.

****

QuickSort의 종료조건이 n < k 일 때 이면, 전체 배열이 최대 n/k개의 단위로 쪼개진다. 파티션을 log n/k 번 수행한 것이라고 생각할 수 있다. 파티션 수행할 때 마다 c\*n 번의 수행시간이 들기 때문에 본 문제에 제시한 QuickSort는 O(n\*log(n/k))의 수행시간이 들어간다. 그리고 이렇게 주어진 배열에 대해서 Insertion Sort를 수행한다. 각각의 모든 원소는 이미 위치관계가 분명한 다른 원소를 제외하고 최대 k개의 원소와 비교를 수행하면 모든 배열이 정렬된다. 따라서 전체 O(k\*n)의 수행시간으로 주어진 배열을 정렬할 수 있다. 두 수행시간을 합쳐서 전체 O(n\*k + n\*log(n/k))의 수행시간이 소요된다.

Insertion sortd의 expected time을 c1\*n^2, QuickSort의 expected time을 c2\*n\*log(n)이라고 하면 전체 위 알고리즘의 수행 시간은 c1\*n\*k + c2\*n\*log(n/k) 로 표현할 수 있다. 이때 k에 어떤 고정된 값을 넣어도 이론적으로는 상수이므로 결국 이다. 그러나 c1\*n\*k + c2\*n\*log(n/k) < c2 \* n \* log n 라는 방정식을 풀어보면 c1/c2 < log k / k 이다. 우변을 미분해서 최대가 되는 k를 구하면 k는 log의 밑수와 같을 때 최대가 되며 c1/c2 < 1/k 를 만족한다면 실제로 이 알고리즘은 일반적인 QuickSort보다 더 빠른 성능을 낼 수 있다.

****

****

Induction을 통한 증명

P(i) : m >= i >= 0 인 정수 i에 대해서, 1 ~ n – i 는 RANDOM\_SAMPLE( m – i, n – i ) 을 통해 m –i / n - i의 확률로 동일하게 선택된다.

1. P(m) : RANDOM\_SAMPLE ( 0, n – m ) 을 통해 공집합이 나온다. 따라서 1 ~ n-m 는 0 / n – m의 확률로 선택된다.
2. P(k)를 가정하면 P(k + 1)이 성립한다.
   1. RANDOM\_SAMPLE ( m – k + 1 , n – k + 1 ) 를 수행했을 때 새로 들어오는 m – k + 1이 선택될 확률을 구한다.
      1. 1 ~ n – k + 1숫자중에 랜덤으로 선택될 확률 1 / n - k + 1
      2. 이미 고른 숫자가 선택되어 m – k + 1 번째 원소가 S에 들어가는 확률
         1. 이미 고른 숫자의 개수 m – k
         2. 확률 m – k / n – k + 1
      3. 총합 m – k + 1 / n – k + 1
   2. 1부터 m – k 사이의 다른 원소들이 선택될 확률을 구한다.
      1. 가정에 의해 이미 선택되었을 확률 m - k / n – k
      2. 아직 선택되지 못한 원소들이 이번에 선택된 확률
         1. 선택받지 못할 확률 n – m / n – k
         2. 이번에 선택될 확률 1 / n – k + 1
      3. i + ii 는 m – k + 1 / n – k + 1이 성립
      4. 따라서 1 ~ n – k + 1 까지의 모든 원소는 m – k + 1 / n – k + 1의 확률로 선택받는다.
   3. A. B의 수학적 귀납법에 의해서 P(i)가 성립한다