All about Quick Sort

이윤승

2019년 6월 6일

차 례

1 소개

- 일반적으로 가장 많이 사용하는 정렬 알고리즘
- 비교 정렬
- 내부정렬

- 불안정 정렬
- 평균 복잡도: $O(n \lg n)$
- 최악의 복잡도 : $O(n^2)$
- C++ std::sort의 내부구현이 퀵소트로 되어있음

2 의사코드 및 동작

분할 정복(divide and conquer) 방법을 통해 설계 되었음

```
QUICKSORT(A,p,r)
    if p < r
        q = PARTITION(A,p,r)
        QUICKSORT(A,p,q-1)
        QUICKSORT(A,q+1,r)
```

```
PARTITION(A ,p ,r)
    x= A[r]
    i = p-1
    for j = p to r-1
        if A[j]<= x
            i = i + 1
            exchange A[i] with A[j]
    exchange A[i+1] with A[r]
    return i + 1
```

PARTITION 프로시저의 시간복잡도는 $\Theta(n)$입니다. https://www.youtube.com/watch?v=hq4dpyuX4Uw&list=PL52K_8WQO5oUuH06MLOrah4h05TZ4n38l&index=11

성능 참고 : https://www.acmicpc.net/blog/view/58

3 복잡도 분석

3.1 최악의 분할 케이스

최악의 경우 분할 케이스를 생각해보자 이는 왼쪽 오른쪽 분할이 한쪽으로 쏠려(오름차순,내림차순) 극도로 불균형하게 일어났을때이다. 피봇값에 의한 분할이 아예 일어나지 않을 때 최악의 케이스가 된다. 이때 비용을 나타낸 재귀함수다.

$$T(n) = T(n-1) + cn$$

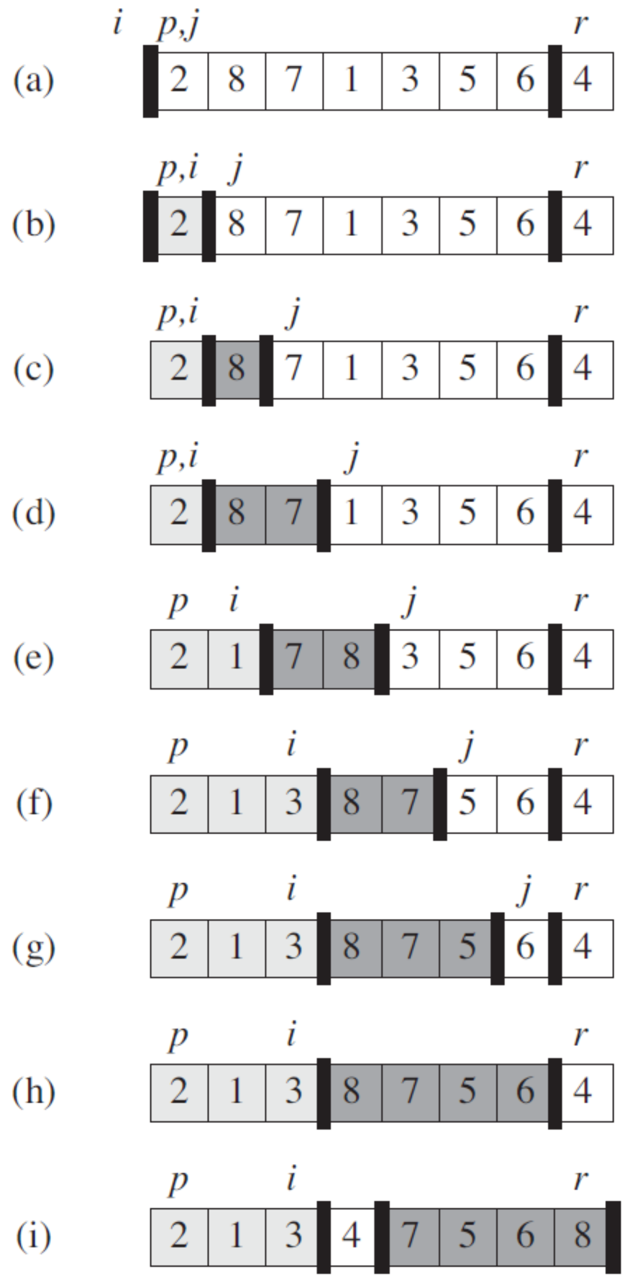


그림 1: quick sort 작동 예시

$$
\begin{aligned}
T(n) &= T(n-1) + cn \\
&= T(n-2) + c(n-1) + cn \\
&= c\sum_{k=1}^{n} k \\
&= \frac{1}{2}cn^2 \\
&= \Theta(n^2)
\end{aligned}
$$

따라서 시간복잡도는 $\Theta(n^2)$이다.

3.2 최선의 분할 케이스

정확하게 반으로 나누어 졌을때 최선의 분할 케이스이다.

이때의 비용을 나타낸 재귀함수는

$$T(n) = 2T(n/2) + cn$$

이다.

그림 2의 재귀트리를 통해서 전체 비용을 계산하여 시간복잡도를 구하면 $\Theta(n \lg n)$ 이다.

3.3 직관적인 일반적인 케이스 방법

랜덤한 분할의 퀵소트를 가정하기 위해서 최선의 분할 케이스와 최악의 분할 케이스가 번갈아 나타난다고 생각해보자

T(n) 일때의 시간복잡도와 T(n-1)일때의 시간복잡도는 둘다 $\Theta(n)$이다. 따라서 이 둘의 시간복잡도를 합쳐도 결국 $\Theta(n)$이고 이를 합쳐서 보면 결국에 최선의 분할 케이스가 된다

따라서 이때의 시간복잡도는 결국 $\Theta(n \lg n)$이다.

4 상세 분석

4.1 최악의-케이스 분석

실제 재귀함수를 일반화 식은 다음이 된다.

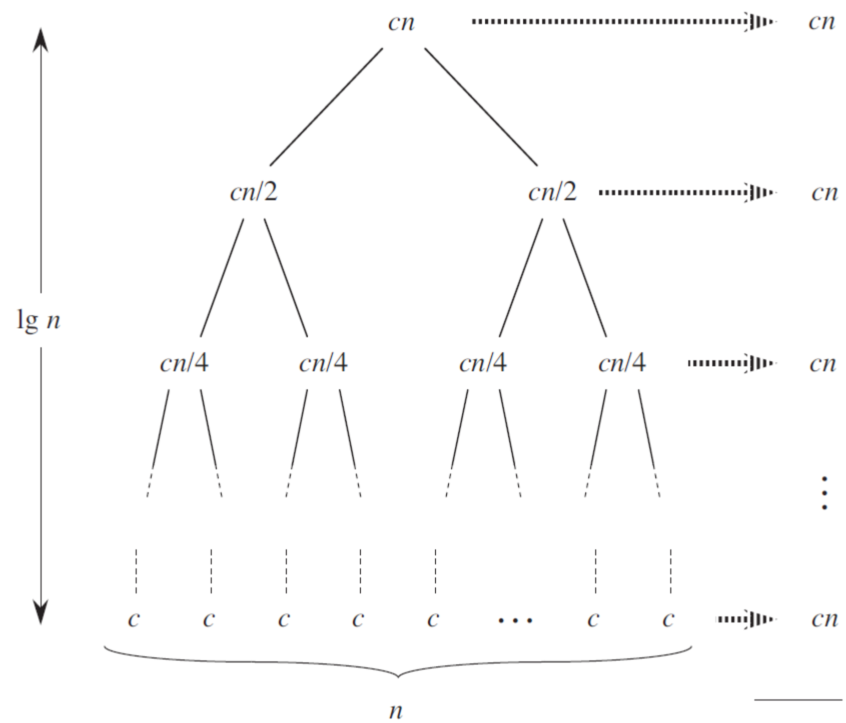


그림 2: quick sort 최선의 분할 케이스 재귀 트리

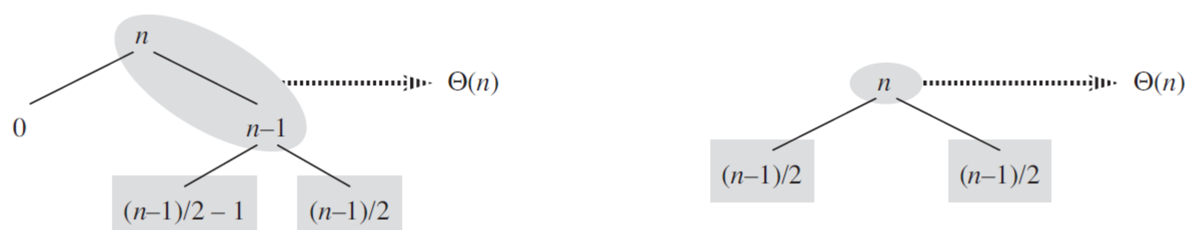


그림 3: quick sort 일반적인 재귀 트리

$$T(n) = \max_{0 \le q \le n-1} (T(q) + T(n-q-1)) + \Theta(n)$$

이때 $T(n)$이 $\Theta(n^2)$임을 보이면된다. 치환법을 이용해서 이 점화식을 풀수있다. $T(n) \le c_1 n^2$이라 가정하자. 그러면

$$T(n) \le \max_{0 \le q \le n-1} (c_1 q^2 + c_1(n-q-1)^2) + \Theta(n)$$

이 성립한다.

$0 \le q \le n-1$인 $c_1 q^2 + c_1(n-q-1)^2$에 대해서

$f(q) = c_1 q^2 + c_1(n-q-1)^2$ 이라하고 $f'(q) = 2c_1 q - 2c_1(n-q-1)$이므로 $q = \dfrac{n-1}{2}$일때 극솟값을 가진다. 이 극솟값은 q의 존재 범위에 포함되고 따라서 이차함수의 특성에 따라 양 끝점이 최댓값이 될수있는 후보가된다. $q = 0$일때와 $q = n-1$일때 극댓값을 가지는데 이때의 두 함숫값은 같아서 둘중 어느값을 택해도 최댓값이 된다.

$$\begin{aligned} T(n) &\le c_1(n-1)^2 + \Theta(n) \\ &\le c_1 n^2 - c(2n-1) + \Theta(n) \\ &\le c_1 n^2 \end{aligned}$$

이때 $\Theta(n) = dn$에서 $c_1 > d$인 상수 c_1을 가지게 함으로 써 결과적으로 $T(n)$이 $c_1 n^2$보다 작거나 같음을 보일수있다. 반대로 $c_2 n^2 \le T(n)$임을 가정하고 $c_2 < d$인 상수를 잡음으로 $T(n)$이 $c_2 n^2$보다 크거나 같음을 보일수있다. 따라서 $T(n) = \Theta(n^2)$를 얻을 수 있다.

5 여러 기초 지식

5.1 시간복잡도

정의 5.1 (복잡도) 상한, 하한,

- 상한 big O

  함수 $f(n), g(n)$에 대해서 $0 \leq f(n) \leq cg(n)(\forall n \leq n_0)$을 만족하는 n_0, 양의 상수 c가 존재할때 $f(n) = O(g(n))$이라한다.

- 하한 omega

  함수 $f(n), g(n)$에 대해서 $0 \leq cg(n) \leq f(n)(\forall n \leq n_0)$을 만족하는 n_0, 양의 상수 c가 존재할때 $f(n) = \Omega(g(n))$이라한다.

- Theta

  $\Theta(g(n))$일 필요충분 조건은 $f(n) = O(g(n))$이고 $f(n) = \Omega(g(n))$이 성립할때 이다.

5.2 조화 급수의 상한과 하한

$\sum_{k=1}^{n} \frac{1}{k} = \Theta(\lg n)$

감소 함수 $f(k)$에 대해서 다음이 성립한다

$$\int_{m-1}^{n} f(x)dx \leq \sum_{k=m}^{n} f(k) \leq \int_{m}^{n+1} f(x)dx$$

증가함수 $f(k)$에 대해 다음이 성립함을 그림 4를 통해서 이해 할 수있다. 감소함수는 이와 반대로 생각하면 쉽게 해당 부등식을 이해할 수 있다.

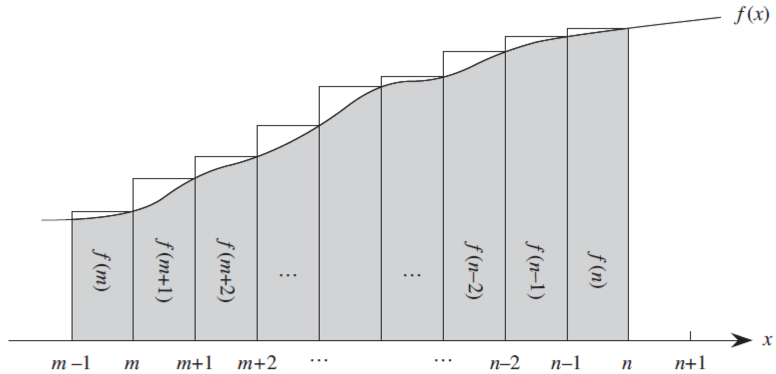
$$\int_{m}^{n+1} f(x)dx \leq \sum_{k=m}^{n} f(k) \leq \int_{m-1}^{n} f(x)dx$$

다음 두가지 방식으로 계산한다

$\sum_{k=2}^{n} \frac{1}{k} + 1 \leq \int_{2}^{n+1} f(x)dx + 1 = \ln(x) + 1 = O(\ln x)$

$\int_{1}^{n+1} f(x)dx = \ln(x+1) = \Omega(\ln x) \leq \sum_{k=1}^{n} \frac{1}{k}$

따라서 $\sum_{k=1}^{n} \frac{1}{k} = \Theta(\lg n)$



(a)

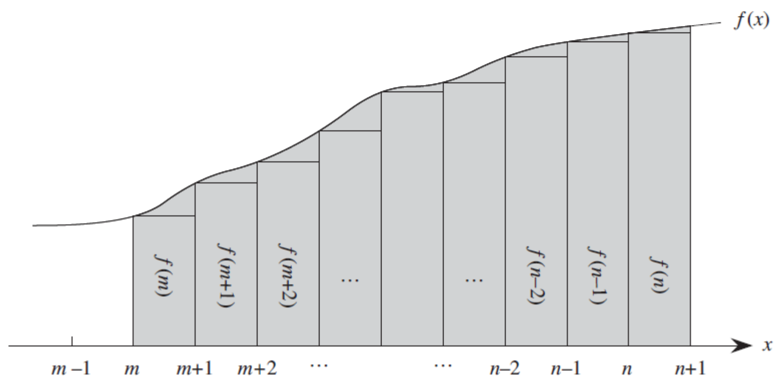


그림 4: 증가함수의 대소비교

5.3 확률

Indicator random variables

$$I\{A\} = \begin{cases} 1 & (\ H \text{ 발생}) \\ 0 & (\ \bar{H} \text{ 발생}) \end{cases}$$

$$
\begin{aligned}
E[X_A] &= E[I\{A\}] \\
&= 1 \times \Pr(A) + 0 \times \Pr(\bar{A})^{1} \\
&= \Pr(A)
\end{aligned}
$$

6 기대수행시간

1 부터 모든 n에 대해서 모든 비용의 평균

$$
\begin{aligned}
E[X] &= E\left[\sum_{i=1}^{n} X_i\right] \\
&= \sum_{i=1}^{n} E[X_i]
\end{aligned}
$$

시간복잡도를 분석하기위해서 다음의 보조정리를 이용한다.

Lemma 6.1. *X*가 길이가 *n*인 배열에서 *QUICKSORT*의 전체 실행에 대해서 *PARTITION*의 *4*행에서 수행된 비교문의 실행 수라고 가정하면 *QUICKSORT*의 실행시간은 *O(N+X)*이다.

Proof. 알고리즘은 PARTITION을 최대 n 회 호출한다. 각 호출은 for 루프를 실행하는데, for 루프의 각 반복은 4행 비교문을 실행한다. □

따라서 모든 호출에 대해서 총 비교수 X를 구하는 문제로 바뀌는데 각 호출에 대한 비교를 분석하지않고 총 비교수를 계산한다. 그렇게 하기 위해 배열 $A = \{z_1, z_2, ..., z_n\}$의 각 요소가 내림차순으로 정렬되어있다고 생각한다. 또한 집합 $Z_{ij} = \{z_i, z_{i+1}, ..., z_j\}$라 정의한다. 여기서 z_i와 z_j는 최대 한번 비교된다. 이유는 PARTITION에서 비교를 하는 경우는 하나의 원소가 pivot으로 선택 되었을 때인데, 이후에 이 pivot은 절대로 다른 원소와 비교하지 않는다. 따라서 $X_{ij} = I\{z_i$가 z_j와 비교한다 $\}$

$$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$$

$$
\begin{aligned}
E[x] &= E\left[\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} X_{ij}\right] \\
&= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} E\left[X_{ij}\right] \\
&= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} Pr\{z_i\text{가}z_j\text{와 비교한다}\}
\end{aligned}
$$

$$
\begin{aligned}
Pr\{z_i\text{가}z_j\text{와 비교한다} &= Pr\{Z_{ij}\text{에서}z_i\text{또는}z_j\text{가 첫번째로 선택된다.}\} \\
&= Pr\{Z_{ij}\text{에서}z_i\text{가 첫번째로 선택된다.}\} + Pr\{Z_{ij}\text{에서}z_j\text{가 첫번째로 선택된다.}\} \\
&= \frac{1}{j-i+1} + \frac{1}{j-i+1} \\
&= \frac{2}{j-i+1}
\end{aligned}
$$

$$
E[X] = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \frac{2}{j-i+1}
$$

이는 $j-i$을 k로 치환해서 상한을 얻을 수 있다.

$$
\begin{aligned}
E[X] &= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&= \sum_{i=1}^{n-1}\sum_{k=1}^{n-i} \frac{2}{k+1} \\
&\le \sum_{i=1}^{n-1}\sum_{k=1}^{n} \frac{2}{k} \\
&\le \sum_{i=1}^{n-1} c\lg n \\
&\le cn\lg n
\end{aligned}
$$

$$
E[X] = O(n\lg n)
$$

하한은 다음과 같이 직접구한다.

$$\begin{aligned}
E[X] &= \sum_{i=1}^{n-1}\sum_{k=1}^{n-i}\frac{2}{k+1} \\
&= \sum_{k=1}^{n-1}\frac{2}{k+1} + \sum_{k=1}^{n-2}\frac{2}{k+1} + \cdots + \sum_{k=1}^{1}\frac{2}{k+1} \\
&= (n-1)\frac{2}{1+1} + (n-2)\frac{2}{2+1} + \cdots + 1 \times \frac{2}{(n-1)+1} \\
&= \sum_{k=1}^{n-1}\frac{2}{k+1} \times (n-k) \\
&= \sum_{k=1}^{n-1}\left(\frac{2n}{k+1} - \frac{2k}{k+1}\right) \\
&= 2n\sum_{k=1}^{n-1}\frac{1}{k+1} - 2\sum_{k=1}^{n-1}\frac{k}{k+1} \\
&\geq 2nc\lg n - 2(n-1)\left(\because -\sum_{k=1}^{n-1}\frac{k}{k+1} \geq -\sum_{k=1}^{n-1}\left(\frac{k}{k+1} + \frac{1}{k+1}\right)\right) \\
&\geq \Omega(n\lg n)
\end{aligned}$$

따라서

$$\Theta(n\lg n)$$

참고 문헌

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7.