The Generalization of Quicksort [*]

Shyue-Horng Shiau Chang-Biau Yang[†]
Department of Computer Science and Engineering
National Sun Yat-Sen University
Kaohsiung, Taiwan 804, Republic of China
shiaush@cse.nsysu.edu.tw cbyang@cse.nsysu.edu.tw

Abstract

In this paper, we propose a generalization of quicksort to solve the problem of sorting the first k largest elements in a set of n elements. Let T_k^n denote the average number of comparisons required for solving the problem. We obtain

$$T_k^n = 2(n-(k-1)) + 2\left(\sum_{m=n-(k-2)}^{n} H_m\right),$$

where $H_m = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{m}$, which is a harmonic number. Besides, we get

$$T_{k+1}^n - T_k^n = 2\left(H_{n-(k-1)} - 1\right).$$

Key words: complexity analysis, quicksort, divide-and -conquer, generalization.

1 Introduction

The quicksort algorithm has been studied by many researchers from many various view points. One of the view points is to generalize quicksort. We shall point out three papers studing the generalization. In order to reexplain the generalization under the same base, we denote the average number of comparisons to sort n elements as T_n [2, p.278]. Then,

$$T_n = (n+1) + \sum_{i=0}^{n-1}\left(p_i T_i + p_{n-i-1} T_{n-i-1}\right), \tag{1}$$

where i is the numbers of elements in the first part, and $n-i-1$ is the size of the second part. Note that quicksort divides the n elements into two parts. If the n elements is in random order initially, then the value of p_i is $\frac{1}{n}$ as well as p_{n-i-1}. Thus, Eq.(1) can be simplified as

$$T_n = (n+1) + \frac{2}{n}\sum_{i=0}^{n-1} T_i. \tag{2}$$

The analysis can be done for uniformly distributed partitions (all probabilities have equal values). The above T_n can be found in many articles[2, p.278][3, p.17][1, p.28].

The first generalization was proposed by Veroy[6]. He showed that the probability $\frac{1}{n}$ can be generalized as follows:

$$p_i = p_{n-i-1}, \quad \sum_{i=0}^{n-1} p_i = 1.$$

Then, Eq.(1) can be simplified as

$$T_n = (n+1) + \frac{2}{n}\sum_{i=0}^{n-1} p_i T_i.$$

[*]This research work was partially supported by the National Science Council of the Republic of China under NSC89-2213-E110-005.

[†]To whom all correspondence should be sent.

The second generalization was proposed by Sedgewick et al.[3, p.252]. They specifically defined an *additive parameter* to be any parameter whose cost function satisfies the linear recursive schema. It is possible to develop a fully general treatment of the average-case analysis of any additive parameter for both of the Catalan tree model and the binary search tree model. For the binary search tree model, let E_n denote the expected value over random binary search trees of size n. They showed

$$T_n = E_n + \frac{2}{n}\sum_{i=0}^{n-1} T_i. \tag{3}$$

Comparing Eq.(3) with Eq.(2), we find that $(n+1)$ is generalized by E_n.

The third paper was proposed by Wang et al.[7]. They studied the following equation:

$$T_n = a_n T_{b_n} + f_n.$$

And now, we are going to generalize quicksort from a different view point. Before going on, we want to stipulate that the result of our sorting is in the nonincreasing order (from the maximum element to the minimum element). In this paper, we want to figure out the whole process of quicksort. The whole process not only shows the final result of the sorting, but also includes each step of finding each element (from the maximum element to the minimum element).

For given n elements, T_n^n denotes the average number of comparisons requicked sort n elements, and T_1^n denotes the average number of comparisons required for finding the maximum element. Our goal is to generalize T_k^n, for $1 \leq k \leq n$, which represents that the first k largest elements are sorted.

In this paper, we obtain

$$T_k^n = 2(n-(k-1)) + 2\left(\sum_{m=n-(k-2)}^{n} H_m\right),$$

where $H_m = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{m}$, which is a harmonic number. In addition, we get

$$T_{k+1}^n - T_k^n = 2\left(H_{n-(k-1)} - 1\right).$$

The rest of this paper is organized as follows. In Section 2, we shall present our generalization of the sorting. In Section 3, we shall give the analysis of the generalization. Finally, Section 4 concludes the paper.

2 Generalization of the quicksort algorithm

We shall use the notation $\langle x_0, x_1, \cdots, x_t\rangle$ to denote a list. Suppose $L_1 = \langle x_0, x_1, \cdots, x_t\rangle$ and $L_2 = \langle y_0, y_1, \cdots, y_{t'}\rangle$, then $\langle L_1, L_2\rangle$ represents the concatenation of L_1 and L_2, which is $\langle x_0, x_1, \cdots, x_t, y_0, y_1, \cdots, y_{t'}\rangle$.

For given n elements, the generalization of quicksort is to find the first k, $1 \leq k \leq n$, largest elements in the sorted order. It is assumed that the n input elements are all distinct. Suppose not, the index or address of each element can be used to distiguish from others. To generalize the quicksort algorithm is not difficult. The generalization algorithm can be represented as a recursive function: $(L_g, c) = GQSort(M, k)$, where L_g is the list containing the sorted sequence, and c denotes the number of elements in L_g, and M is the set of input elements. Initially, we set an initial list $L_g = NULL$ and an initial counting value $c = 0$. Our generalized algorithm is as follows:

Algorithm Generalized Quicksort:
$(L_g, c) = GQSort(M, k)$

Step 1: If M contains exactly one element and its value is y, then return $(\langle y\rangle, 1)$.

Step 2: If M contains no element , then return $(NULL, 0)$.

Step 3: If M contains more than one element, then let the first element of M be the pivot, denoted as x_h. Compare x_h with all elements in M. Then M is splitted into two subsets $M_l = \{x|x > x_h\}$ and $M_r = \{x|x < x_h\}$. Perform $(L_g, c) = GQSort(M_l, k)$.

Step 3.1: If $k \leq c$, then return (L_g, c).

Step 3.2: Set $L_g = \langle L_g, x_h \rangle$ and $c = c + 1$. If $k \leq c$, then return (L_g, c).

Step 3.3: Perform $(L'_g, c') = GQSort(M_r, k - c)$.

Step 3.4: Set $L_g = \langle L_g, L'_g \rangle$ and $c = c' + c$. Return (L_g, c).

Let T^n_k denote the average number of comparisons required for $GQSort(M, k)$, where $|M| = n$.

For example, suppose the input file contains 9 elements, which is $M = \{5, 1, 7, 0, 9, 2, 8, 3, 6, 4\}$, and $k = 4$. In other words, we want to sort the first 4 largest elements in the set of 9 elements, and we desire to calculate T^9_4. The process is shown in Figure 1.

0.		$\{5, 1, 7, 0, 9, 2, 8, 3, 6, 4\}$
1.	$s3$ p=5	$\{7, 9, 8, 6\}, 5, \{1, 0, 2, 3, 4\}$
2.	$s3$ p=7	$\{9, 8\}, 7, \{6\}, 5, \{1, 0, 2, 3, 4\}$
3.	$s3$ p=9	$\check{9}, \{8\}, 7, \{6\}, 5, \{1, 0, 2, 3, 4\}$
4.	$s1$	$\check{9}, \check{8}, 7, \{6\}, 5, \{1, 0, 2, 3, 4\}$
5.	$s32$	$\check{9}, \check{8}, \check{7}, \{6\}, 5, \{1, 0, 2, 3, 4\}$
6.	$s1$	$\check{9}, \check{8}, \check{7}, \check{6}, 5, \{1, 0, 2, 3, 4\}$

3. :	round 3
$\check{9}$:	a sorted element
$\{7, 9, 8, 6\}$:	a set waiting for sorting
$s3$:	step 3 in the algorithm
p=5 :	5 selected to be a pivot

Figure 1: an example of sorting the first 4 largest elements among 9 elements with quicksort.

In round 3, after the maximum element 9 is selected, all elements are separated into three parts with $\check{9}$,7,5 as separators. The three parts are $\{8\}$, $\{6\}$ and $\{1, 0, 2, 3, 4\}$. T^9_1 denotes the average number of comparisons to reach situation in round 3.

3 Analysis of the generalization

Expanding Eq.(2), we get

$$T_n = (n+1) + \frac{1}{n} \left[\begin{array}{c} (T_0 + T_{n-1}) \\ +(T_1 + T_{n-2}) \\ +\cdots \\ +(T_{n-1} + T_0) \end{array} \right], \quad (4)$$

where $T_0 = 0$. Thus, $T_1 = 2$.

T_n in Eq.(4) can be regarded as T^n_n. Therefore set T_i to T^i_i , for $0 \leq i \leq n$, we obtain

$$T^n_n = (n+1) + \frac{1}{n} \left[\begin{array}{c} \left(T^0_0 + T^{n-1}_{n-1}\right) \\ +\left(T^1_1 + T^{n-2}_{n-2}\right) \\ +\cdots \\ +\left(T^{n-1}_{n-1} + T^0_0\right) \end{array} \right], \quad (5)$$

and we have

$$T^0_0 = 0 \text{ and } T^1_1 = 2. \quad (6)$$

Generalizing Eq.(5), we obtain

$$T^n_k = (n+1) + \frac{1}{n} \left[\begin{array}{c} \left(T^0_0 + T^{n-1}_{k-1}\right) \\ +\left(T^1_1 + T^{n-2}_{k-2}\right) \\ +\cdots \\ +\left(T^{k-1}_{k-1} + T^{n-k}_0\right) \\ +\left(T^k_k + 0\right) \\ +\left(T^{k+1}_k + 0\right) \\ +\cdots \\ +\left(T^{n-1}_k + 0\right) \end{array} \right]. \quad (7)$$

The first term, $(n + 1)$, is the average number of comparisons for partitioning n elements into two parts.

The second term, $\frac{1}{n}$, is one of the cases that each element in M can be selected as the pivot in step 3 of $GQSort(M, k)$. Then, the pivot may be one of the n elements randomly, each case having probability $\frac{1}{n}$.

The third term, $\left(T_0^0 + T_{k-1}^{n-1}\right)$, is the case that the pivot is the largest element. Since the largest element is selected, the first part includes no element, represented by T_0^0. And in the second part, we hvae to find the $k-1$ largest elements among the $n-1$ elements, represented by T_{k-1}^{n-1}.

The fourth term, $\left(T_1^1 + T_{k-2}^{n-2}\right)$, is the case that the second largest element is selected to be the pivot. Obviously, the largest element is contained in the first part. T_1^1 represents the first part status. And in the second part, the subterm, T_{k-2}^{n-2}, means that we have to find the $k-2$ largest elements among $n-2$ elements after the largest and second largest elements are found.

The kth term, $\left(T_{k-1}^{k-1} + T_0^{n-k}\right)$, is the case that the pivot is the kth largest element. Note that, $T_0^{n-k} = 0$.

We omit the explanation of the rest of the equation, since it is quite similar to the above cases.

In the following, H_n is used to denote $1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$, which is a harmonic number.

Lemma 1

$$T_1^n - T_1^{n-1} = 2.$$

Proof. Substituting k with 1 into Eq.(7), we get

$$T_1^n = (n+1) + \frac{1}{n}\begin{bmatrix} T_0^0 \\ +T_1^1 \\ +\cdots \\ +T_1^{n-2} \\ +T_1^{n-1} \end{bmatrix}. \quad (8)$$

Substituting n with $n-1$ into Eq.(8), we get

$$T_1^{n-1} = n + \frac{1}{n-1}\left(T_0^0 + T_1^1 + \cdots + T_1^{n-2}\right)$$

Substituting the above equality into Eq.(8), we get

$$\begin{aligned} & T_1^n \\ = \; & (n+1) + \frac{1}{n}\begin{pmatrix} (n-1)\left(T_1^{n-1} - n\right) \\ +T_1^{n-1} \end{pmatrix} \\ = \; & T_1^{n-1} + 2. \end{aligned}$$

Thus,

$$T_1^n - T_1^{n-1} = 2.$$

■

Corollary 2

$$T_1^n = 2n.$$

Proof. By Lemma 1, we have

$$\begin{aligned} T_1^n - T_1^{n-1} &= 2 \\ T_1^{n-1} - T_1^{n-2} &= 2 \\ &\cdots \\ T_1^2 - T_1^1 &= 2. \end{aligned}$$

Summing the above equalities, we obtain

$$T_1^n - T_1^1 = 2\,(n-1)\,.$$

By Eq.(6), we have

$$T_1^n = 2n.$$

■

Lemma 3

$$T_k^n - T_k^{n-1} = 2 + 2\left(H_n - H_{n-(k-1)}\right).$$

Proof. Rearranging Eq.(7), we have

$$\begin{aligned} & T_k^n \\ = \; & n+1 \\ & +\frac{1}{n}\begin{bmatrix} \left(\sum\limits_{m=1}^{k} T_{k-m}^{n-m}\right) \\ +\left(\sum\limits_{m=0}^{k-1} T_m^m + \sum\limits_{m=k}^{n-1} T_k^m\right) \end{bmatrix} \quad (9) \\ = \; & n+1 \\ & +\frac{1}{n}\begin{bmatrix} \left(\sum\limits_{m=1}^{k} T_{k-m}^{n-m}\right) \\ +\left(\sum\limits_{m=0}^{k-1} T_m^m + \sum\limits_{m=k}^{n-2} T_k^m\right) \\ +T_k^{n-1} \end{bmatrix}. \quad (10) \end{aligned}$$

Substituting n with $n-1$ into Eq.(9), we get

$$\begin{aligned} & T_k^{n-1} \\ = \;& n + \tfrac{1}{n-1} \left[\begin{array}{l} \left(\sum\limits_{m=1}^{k} T_{k-m}^{n-1-m} \right) \\ + \left(\sum\limits_{m=0}^{k-1} T_m^m + \sum\limits_{m=k}^{n-1} T_k^m \right) \end{array} \right]. \end{aligned}$$

Rearranging the above equality, we have

$$\begin{aligned} & \left(\sum_{m=0}^{k-1} T_m^m + \sum_{m=k}^{n-2} T_k^m \right) \\ = \;& (n-1)\left[T_k^{n-1} - n\right] - \left(\sum_{m=1}^{k} T_{k-m}^{n-1-m} \right). \end{aligned}$$

Substituting the above equality into Eq.(10), we get

$$\begin{aligned} & T_k^n \\ = \;& n+1 \\ & +\tfrac{1}{n} \left\{ \begin{array}{l} \left(\sum\limits_{m=1}^{k} T_{k-m}^{n-m} \right) \\ +(n-1)\left[T_k^{n-1} - n\right] \\ - \left(\sum\limits_{m=1}^{k} T_{k-m}^{n-1-m} \right) + T_k^{n-1} \end{array} \right\} \\ = \;& n+1 \\ & +\tfrac{1}{n} \left\{ \begin{array}{l} nT_k^{n-1} - (n-1)n \\ + \left(\sum\limits_{m=1}^{k} T_{k-m}^{n-m} \right) \\ - \left(\sum\limits_{m=1}^{k} T_{k-m}^{n-1-m} \right) \end{array} \right\}. \end{aligned}$$

Rearranging the above equality, we have

$$\begin{aligned} & T_k^n - T_k^{n-1} \\ = \;& 2 + \tfrac{1}{n} \left\{ \begin{array}{l} \left(\sum\limits_{m=1}^{k} T_{k-m}^{n-m} \right) \\ - \left(\sum\limits_{m=1}^{k} T_{k-m}^{n-1-m} \right) \end{array} \right\}. \end{aligned}$$

Let $G_k^n = T_k^n - T_k^{n-1}$, for $1 \le k \le n-1$, we get

$$G_k^n = 2 + \tfrac{1}{n} \left\{ \sum_{m=1}^{k} G_{k-m}^{n-m} \right\}.$$

Expanding the above equation, we get

$$\begin{aligned} & G_k^n \\ = \;& 2 + \tfrac{1}{n} \left\{ \sum_{m=1}^{k-1} G_{k-m}^{n-m} + G_0^{n-k} \right\} \\ = \;& 2 + \tfrac{1}{n} \left\{ \sum_{m=1}^{k-1} G_{k-m}^{n-m} \right\}. \end{aligned} \tag{11}$$

Substituting n with $n-1$ and k with $k-1$ into the above equation, we obtain

$$G_{k-1}^{n-1} = 2 + \tfrac{1}{n-1} \left\{ \sum_{m=1}^{k-2} G_{k-1-m}^{n-1-m} \right\}$$

Substituting the above equality into Eq.(11), we get

$$G_k^n = 2 + \tfrac{1}{n} \left\{ G_{k-1}^{n-1} + \left(G_{k-1}^{n-1} - 2\right)(n-1) \right\}$$

$$G_k^n - G_{k-1}^{n-1} = \tfrac{2}{n}.$$

By the above equality, we have

$$\begin{aligned} G_k^n - G_{k-1}^{n-1} &= \tfrac{2}{n} \\ G_{k-1}^{n-1} - G_{k-2}^{n-2} &= \tfrac{2}{n-1} \\ &\cdots \\ G_3^{n-(k-3)} - G_2^{n-(k-2)} &= \tfrac{2}{n-(k-3)} \\ G_2^{n-(k-2)} - G_1^{n-(k-1)} &= \tfrac{2}{n-(k-2)} \end{aligned}$$

Summing the above equalities, we obtain

$$\begin{aligned} & G_k^n - G_1^{n-(k-1)} \\ = \;& 2\left(\tfrac{1}{n} + \tfrac{1}{n-1} + \cdots + \tfrac{1}{n-(k-3)} + \tfrac{1}{n-(k-2)} \right) \\ = \;& 2\left(\sum_{m=1}^{n} \tfrac{1}{m} - \sum_{m=1}^{n-(k-1)} \tfrac{1}{m} \right) \\ = \;& 2\left(H_n - H_{n-(k-1)} \right). \end{aligned}$$

Rearranging the above equality, we have

$$G_k^n = G_1^{n-(k-1)} + 2\left(H_n - H_{n-(k-1)} \right).$$

Since $G_k^n = T_k^n - T_k^{n-1}$, and by Lemma 1, we get

$$\begin{aligned} & T_k^n - T_k^{n-1} \\ = \; & T_1^{n-(k-1)} - T_1^{n-(k-1)-1} \\ & +2\left(H_n - H_{n-(k-1)}\right) \\ = \; & 2 + 2\left(H_n - H_{n-(k-1)}\right). \end{aligned}$$

■

Theorem 4

$$T_k^n = 2(n-(k-1)) + 2\left(\sum_{m=n-(k-2)}^{n} H_m\right).$$

Proof. By Lemma 3, we get

$$\begin{aligned} & T_k^n - T_k^{n-1} \\ = \; & 2 + 2\left(H_n - H_{n-(k-1)}\right), \\ & T_k^{n-1} - T_k^{n-2} \\ = \; & 2 + 2\left(H_{n-1} - H_{n-1-(k-1)}\right), \\ & \cdots \\ & T_k^{k+1} - T_k^{k} \\ = \; & 2 + 2\left(H_{k+1} - H_{k+1-(k-1)}\right). \end{aligned}$$

Summing the above equalities, we obtain

$$\begin{aligned} & T_k^n - T_k^k \\ = \; & 2(n-k) \\ & +2\left(\sum_{m=k+1}^{n} H_m - \sum_{m=2}^{n-(k-1)} H_m\right). \end{aligned} \quad (12)$$

Graham et al.[1, p.29] showed that the average number of comparisons required to quicksort n elements is

$$T_n^n = 2(n+1)H_n - 2n. \quad (13)$$

Thus, substituting $T_k^k = 2(k+1)H_k - 2k$ into Eq.(12), we have

$$\begin{aligned} & T_k^n \\ = \; & 2(k+1)H_k - 2k + 2(n-k) \\ & +2\left(\sum_{m=k+1}^{n} H_m - \sum_{m=2}^{n-(k-1)} H_m\right) \\ = \; & 2kH_k - 2k + 2H_k + 2(n-k) \\ & +2\left(\sum_{m=k+1}^{n} H_m - \sum_{m=2}^{n-(k-1)} H_m\right). \end{aligned} \quad (14)$$

One of important properties of harmonic summation is as follows[1, p.41 eq(2.36) p.279 eq(6.67)]:

$$nH_n - n = \sum_{m=0}^{n-1} H_m.$$

Thus, substituting $kH_k - k = \sum_{m=0}^{k-1} H_m$ into Eq.(14), we have

$$\begin{aligned} & T_k^n \\ = \; & 2\sum_{m=0}^{k-1} H_m + 2H_k + 2(n-k) \\ & +2\left(\sum_{m=k+1}^{n} H_m - \sum_{m=2}^{n-(k-1)} H_m\right) \\ = \; & 2(n-k) \\ & +2\left(\sum_{m=0}^{n} H_m - \sum_{m=0}^{n-(k-1)} H_m + 1\right) \\ = \; & 2(n-(k-1)) + 2\left(\sum_{m=n-(k-2)}^{n} H_m\right). \end{aligned}$$

■

Theorem 5

$$T_{k+1}^n - T_k^n = 2(H_{n-(k-1)} - 1).$$

Proof. Substituting k with $k+1$ into Theorem 4, we obtain

$$T_{k+1}^n = 2(n-k) + 2\left(\sum_{m=n-(k-1)}^{n} H_m\right).$$

Subtracting Theorem 4 from the above equation, we have

$$\begin{aligned} & T_{k+1}^n - T_k^n \\ = & -2 + 2H_{n-(k-1)} \\ = & 2(H_{n-(k-1)} - 1). \end{aligned}$$

■

4 Conclusion

In Lemma 2, we have $T_1^n = 2n$. In the equation, T_1^n is the average number of comparisons required for finding the maximum element in the quicksort process. With a very simple algorithm, finding the maximum element among n elements only needs $n - 1$ comparisons. In quicksort, why does it take $2n$ average numbers of comparisons to find the maximum element? Although quicksort follow is based on the divide-and-conquer approach, we want to explain the reason from a different view.

In section 2, we show an example. After the maximum element 9 is selected, all elements are separated into three parts by $\check{9}$,7,5. The three parts are $\{8\}$, $\{6\}$ and $\{1, 0, 2, 3, 4\}$.

In quicksort, after the process of maximum finding terminates, we can use these selected elements to build some seperate layers and then to distribute the data elements into those layers properly. Thus, comparisons would not be duplicated too many and the required time can be reduced. The layer concept is quite similar to the layer concept proposed by Yang [8][4][5]. The layer concept under the broadcast communication model can be used to reduce conflict resolution time. Thus, initially to pay a little bit extra computing power will repay eventually.

There are many cases which illustate how the harmonic numbers arise naturally in simple situations[1, p.273]. In this paper, we have shown that our analysis is such a case. We hope our experience can be extended from the generalization of quicksort to solve other problems and to find more other cases.

References

[1] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison-Wesley Publishing Company, USA, 1994.

[2] P. W. J. Purdom and C. A. Brown. *The Analysis of Algorithms*. Holt Publishing Company, Rinehart and Winston, New York, USA, 1985.

[3] R. Sedgewick and P. Flajolet. *An Introduction to the Analysis of Algorithms*. Addison-Wesley Publishing Company, USA, 1996.

[4] S. H. Shiau and C. B. Yang. A fast maximum finding algorithm on broadcast communication. *Information Processing Letters*, 60:81–96, 1996.

[5] S. H. Shiau and C. B. Yang. A fast sorting algorithm on broadcast communications. In *Proc. of Second International Conference on Parallel Computing Systems(PCS99), Ensenada, Baja California, Mexico*, pages 87–92, 1999.

[6] B. S. Veroy. Average complexity of divide-and-conquer algorithms. *IEEE Transactions on Computers*, 38(2):278–283, Feb. 1989.

[7] X. Wang and Q. Fu. A frame for general divide-and-conquer recurrences. *Information Processing Letters*, 59:45–51, 1996.

[8] C. B. Yang. Reducing conflict resolution time for solving graph problems in broadcast communications. *Information Processing Letters*, 40:295–302, 1991.