"HENRI COANDA"
AIR FORCE ACADEMY
ROMANIA

"GENERAL M.R. STEFANIK"
ARMED FORCES ACADEMY
SLOVAK REPUBLIC

INTERNATIONAL CONFERENCE of SCIENTIFIC PAPER
AFASES 2014
Brasov, 22-24 May 2014

AN ASYMPTOTIC PROPERTY OF THE MERGING ALGORITHM

Paul VASILIU

Faculty of Marine Engineering, "Mircea cel Bătrân" Naval Academy, Constanta, Romania

Abstract: *This paper focuses on the study of the asymptotic behavior of the medium number of comparisons from the merging algorithm on two sorted arrays, depending on the total length of the resulting array obtained by merging. This paper presents a program written in C++ that supports the theoretical result that was obtained.*

Keywords: *arrays, data analysis, merging, sorting.*
***MSC2010*:** 26A03, 26A06.

1. INTRODUCTION

Let it be the following arrays sorted in ascending order $V=\left(v_1,v_2,\ldots,v_p\right)$ and $W=\left(w_1,w_2,\ldots,w_q\right)$. The result of the merging operation of two arrays sorted in ascending or descending order, *V* and *W*, is a new array, *X*, that contains all the elements from *V* and *W* sorted in ascending or descending order. The algorithm of merging two arrays is well known. The algorithm consists of iterating through the *V* and *W* arrays and comparing each current element v_i from *V* with the current element w_j from *W*. The minimum value between v_i and w_j is written to the resulting array, *X*. In the merging algorithm, one of the *V* or *W* arrays is iterated first and the elements of the other one are written to the *X* array.

Let it be the following merging function written in C++:

```
void merge(int *v,int *w,int *x,int p,int q)
{
int i,j,k;
i=j=k=0;
while(i<p && j<q){
    if (v[i]<w[j]){
        x[k++]=v[i++];
        }
    else
        x[k++]=w[j++];
}
 while (i<p)
      x[k++]=v[i++];
 while (j<q)
      x[k++]=w[j++];
}
```

Let it be the sorted array in an ascending order $X=\left(x_1,x_2,\ldots,x_n\right)$ with $n=p+q$

elements, obtained by merging the V and W arrays. Let it be C, the medium number of comparisons, defined by the total number of comparison divided by the number of possible arrays V and W. The [1] and [2] papers present the limits of the medium required number of comparisons, C, depending on the lengths, p and q of the V and W arrays. In this paper we will prove that the medium number of comparison required for obtaining the X array sorted in ascending order can be approximated with $n-2$. It must be mentioned that this number does not depend on the lengths of V or W arrays.

The obtained result is validated with a program written in C++ language.

2. FINDING THE C NUMBER

The operation of merging the V and W arrays involves comparing the elements of these arrays and updating the X array. Also, for merging, one array will be iterated first. Obviously, the total number of comparisons required for building the X array is equal to $n-m$, where m is the number of remained elements from the array that was not iterated. The minimum number of comparisons equals to the minimum value between p and q in the following case: if all the elements of the array with the smallest number of elements, are smaller than all the elements from the array with more elements that the first one. If the elements with the greatest values from V and W will be the greatest elements from X array, then the number of comparisons is equal to $p+q-1$.

Without reducing the generality we will suppose that the V array will be iterated first. In the end, we will double the value of the number of comparisons by including the case when W will be iterated first.

It is clear that the greatest element from V will influence the value of C. Let it be the final array sorted in ascending order, with n elements $X=(x_1,x_2,\ldots,x_n)$, with $x_1<x_2<\cdots<x_n$. In order to compute C, we have to follow the next steps:

1. It is assumed that x_i is the greatest element from V and then there are i comparisons required;

2. Compute the number of possible arrays V for which x_i is the greatest element from V;

3. Repeat steps 1 and 2 for all the possible choices of x_i;

4. Compute the value of C.

If x_1 is the greatest element from V, than there is an array $V=(x_1)$, with no element that precede x_1, which means that only one comparison is needed, $(1\cdot 2^{1-1})$.

If x_2 is the greatest element from V, than there are two possible array that contain this element: $V=(x_2)$ and $V=(x_1,x_2)$, where x_1 might be an element from V. For both these possible arrays, 2 comparisons are needed for each array, (x_2 with x_1 and x_2 with x_2 or with x_3), in total 4 comparisons, $(2\cdot 2^{2-1})$.

If x_3 is the greatest element from V, then there are 2 possible values that can exists in V: x_1 and/or x_2. In this case there are 4 possible V arrays with x_3, being the greatest element, $V=(x_3)$, $V=(x_1,x_3)$ with $x_1<x_3$, $V=(x_2,x_3)$ with $x_2<x_3$, $V=(x_1,x_2,x_3)$ with $x_1<x_2<x_3$ or $x_2<x_1<x_3$. For each of the four arrays, 3 comparisons are needed, (x_3 wit x_1, x_3 with x_2 and x_3 with x_3 or with x_4), which makes a total of 12 comparisons, $(3\cdot 2^{3-1})$.

In general, if x_i is the greatest element from V, there are $i-1$ possible elements that can precede it and that can be elements of V. The number of subsets with $i-1$ elements is equal to 2^{i-1}, so there are 2^{i-1} V arrays with x_i being the greatest element. The total number of comparisons equals to $i\cdot 2^{i-1}$.

Because we supposed that the V array will be first iterated, this implies that $i<=n-1$. Suppose by contradiction that $i=n$, W could be iterated first because the greatest element from X is now in V.

From the above considerations it follows that

the total number of comparisons required for iterating first the V array, and then W is equal to:

$$\sum_{i=1}^{n-1} i \cdot 2^{i-1} \tag{1}$$

The total number of possible V arrays is equal to:

$$\sum_{i=1}^{n-1} 2^{i-1} = 2^{n-1} - 1 \tag{2}$$

By symmetry, supposing that W is iterated first, and after that V, it follows that the number of comparisons equals:

$$\sum_{i=1}^{n-1} i \cdot 2^{i-1} \tag{3}$$

and the number of possible arrays W equals to:

$$\sum_{i=1}^{n-1} 2^{i-1} = 2^{n-1} - 1 \tag{4}$$

This leads to a total number of comparisons equal to:

$$2 \cdot \sum_{i=1}^{n-1} i \cdot 2^{i-1} = \sum_{i=1}^{n-1} i \cdot 2^{i} \tag{5}$$

and the total number of possible arrays V and W is equal to:

$$2 \cdot \left(2^{n-1} - 1\right) = 2^{n} - 2 \tag{6}$$

From the equality:

$$1 + x + x^2 + \cdots + x^n = \frac{x^{n+1} - 1}{x - 1} \tag{7}$$

which occurs for $x \neq 1$, through derivation, and then multiplication with x and substitution of x with 2 we have the following equality:

$$\sum_{i=1}^{n-1} i \cdot 2^{i} = (n-2) \cdot 2^{n} + 2 \tag{8}$$

The medium number of comparisons becomes:

$$C = \frac{(n-2) \cdot 2^{n} + 2}{2^{n} - 2} \tag{9}$$

By processing (9), we obtain:

$$C = n - 2 + \frac{2 \cdot n - 2}{2^{n} - 2} \tag{10}$$

From the equality:

$$\lim_{n \to \infty} \frac{2 \cdot n - 2}{2^{n} - 2} = 0 \tag{11}$$

results that for values of n big enough, C can be approximate with $n - 2$.

The above chart of the function $y(x) = \frac{2 \cdot x - 2}{2^{x} - 2}$, $x \geq 0$, $x \neq 1$ and $y(1) = \frac{1}{\ln 2}$ proves that the approximation is very good for values of $n \geq 10$.

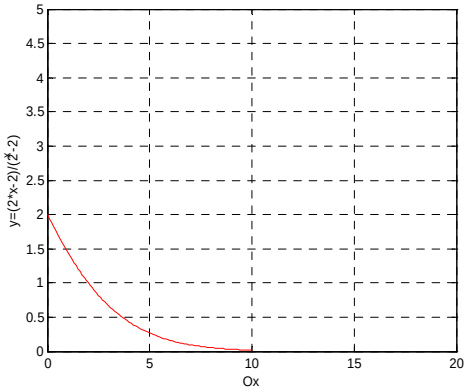

Figure 1. The optimal value of n

2.1. A VERIFICATION OF THE THEORETICAL RESULT

This theoretical result can be experimentally checked with the following C++ program :

```
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
#include <math.h>
#include <stdlib.h>
int * aloc(int n)
{
```

```
 int *p;
 p=(int *)malloc(n*sizeof(int));
 return p;
}
void sort_ascending(int *v,int n)
{
int i,j,temp;
for(i=0;i<n;i++)
for(j=i+1;j<n;j++){
if(v[i]>v[j]){
   temp=v[i];
   v[i]=v[j];
   v[j]=temp;
}}}
void gen(int *v,int n)
{
int i;
for(i=0;i<n;i++)
v[i]=rand();
}
int interclas(int *v,int *w,int *x,int p,int q)
{
int i,j,k,ncomp;
ncomp=i=j=k=0;
while(i<p && j<q){
ncomp++;
if (v[i]<w[j])
x[k++]=v[i++];
else
x[k++]=w[j++];
}
while (i<p)
x[k++]=v[i++];
while (j<q)
x[k++]=w[j++];
return ncomp;
}
int main()
{
int p,q,*V,*W,*X,nc;
printf(" Size of V = ");
scanf("%d",&p);
printf(" Size of  W = ");
scanf("%d",&q);
V=aloc(p);
W=aloc(q);
X=aloc(p+q);
gen(V,p);
gen(W,q);
sort_ascending(V,p);
sort_ascending(W,q);
nc=interclas(V,W,X,p,q);
printf(" C = %d\n",nc);
getch();}
```

The input of the program are: size p and q of the arrays $V=\left(v_1,v_2,\ldots,v_p\right)$ and $W=\left(w_1,w_2,\ldots,w_q\right)$ respectively. The program random generate the arrays, ascending sort the arrays, merge the arrays and generate $X=\left(x_1,x_2,\ldots,x_n\right)$ array with $n=p+q$ components. Are counted comparisons.

An example is:

Size of V = 78
Size of W = 67
C = 143

In the above example $p=78$, $q=67$, $n=p+q=78+67=145$. The medium number of comparisons is equal with: $n-2=145-2=143$.

3. CONCLUSIONS

In papers [1, 2] is determined the medium number of comparisons needed for merging the arrays $V=\left(v_1,v_2,\ldots,v_p\right)$ and $W=\left(w_1,w_2,\ldots,w_q\right)$. This number depends on the lengths, p and q, of these two arrays. In the presented paper we proved that the medium number of comparisons can be computed depending on the sum of the length of these two arrays $n=p+q$. Moreover, we showed that for $n\geq 10$, the medium number of comparisons can be approximated with $n-2$.

REFERENCES

1. Dijkstra, E. W., *Some beautiful arguments using mathematical induction.* Acta Informatica 13(1982).
2. Knuth, D. E. *The Art of Computer Programming, Volume 3: Sorting and Searching.* Addison Wesley (1973).

"HENRI COANDA"
AIR FORCE ACADEMY
ROMANIA

"GENERAL M.R. STEFANIK"
ARMED FORCES ACADEMY
SLOVAK REPUBLIC

INTERNATIONAL CONFERENCE of SCIENTIFIC PAPER
AFASES 2014

Brasov, 22-24 May 2014