Volume XXIV 2021
ISSUE no.1
MBNA Publishing House Constanta 2021

Scientific Bulletin of Naval Academy

SBNA PAPER • OPEN ACCESS

Automatic organization of a set in a heap

To cite this article: Paul VASILIU, Florenţiu DELIU and Tiberiu PAZARA, Scientific Bulletin of Naval Academy, Vol. XXIV 2021, pg.117-121.

Submitted: 13.03.2021
Revised: 15.06.2021
Accepted: 22.07.2021

Available online at www.anmb.ro

ISSN: 2392-8956; ISSN-L: 1454-864X

doi: 10.21279/1454-864X-21-I1-014

Automatic organization of a set in a heap

Paul Vasiliu[1], Florenţiu Deliu[2], Tiberiu Pazara[3]

[1, 2, 3] "Mircea cel Bătrân" Naval Academy, Constanţa, România

E-mail: [1] paul.vasiliu@anmb.ro [2] florentiu.deliu@anmb.ro, [3] tiberiu.pazara@anmb.ro

Abstract. The organization of a set in the form of a heap is the basis for the construction of binary search trees. The algorithm for organizing a set in the form of a pile is laborious and requires a high volume of resources. In this paper we have automated the algorithm for organizing a set in the form of a heap. We developed a program in the C++ programming language that implements the heap generation algorithm and I analyzed a numerical example.

1. The heap

A heap is a vector that can also be viewed as a binary tree. The nodes are traversed from left to right and from top to bottom. A necessary property for a binary tree to be called a heap is for all levels to be complete, except for the last one, which is completed starting from the left and continuing to a certain point.

The height of a tree is the maximum depth of a node, considering the root as the depth node 0. It can be shown that the height of a heap with n nodes is equal to $\log_2 n$. The number n of nodes of a heap of height h is between 2^h and $2^{h+1} - 1$.

The parent of a node i is the node $\left\lfloor \frac{i}{2} \right\rfloor$, and the children of the node i are the nodes $2 \cdot i$ and $2 \cdot i + 1$. If $n = 2 \cdot i$, then node $2 \cdot i + 1$ does not exist, and node i has only one child. If $2 \cdot i > n$, then node i is the leaf and has no child.

The most important property of the heap, which makes it useful in operations to find the maximum, is that the value of any node is greater than or equal to the value of any of its children. Since the operator $\geq$ is transitive, it follows that the value of a node is greater than or equal to the value of any of its grandchildren.

The value of any node is greater than or equal to the value of any node in the subtree whose root it is. A partially ordered tree is a tree in which each vertex has received a label. The label of a peak has higher priority than the label of the descendants.

We consider the set $X = \{1,2,\cdots,n\}$ and $H = (X,U)$ a tree. Note with $\lfloor x \rfloor$ the floor of the real number x, $u \in U$, $u = \left\lfloor \frac{i}{2} \right\rfloor$, $i \in U$. In these conditions $(H,\leq)$ it's a totally orderly set. Let's consider the set $= \{a_1, a_2, \cdots, a_n\}$. If a new partial order relation is defined on H such that $a_i \leq a_{\left\lfloor \frac{i}{2} \right\rfloor}$, for all $i \in \{2,3,\cdots,n\}$ then it is defined that H is organized in a heap.

To the set H, organized in a heap, is attached the binary tree $H = (X,U)$.

2. The algorithm

We consider the set $H = \{a_1, a_2, \cdots, a_n\}$. An algorithm for organizing the H set in the form of a heap is:

```
start
heap ← 0
while heap=0
do
i ← 2
while i $\le$ n
do
if $a_i \le a_{\lfloor \frac{i}{2} \rfloor}$
then
heap ← 1
else
temp ← $a_i$
$a_i \leftarrow a_{\lfloor \frac{i}{2} \rfloor}$
$a_{\lfloor \frac{i}{2} \rfloor}$ ← temp
i ← n+1
heap ← 0;
▪ endif
i ← i+1
▪ endwhile
▪ endwhile
end
```

3. An example

Let's consider the set $H = \{3,20,44,47,11,7,2,37\}$. Obvious $n = 8$. We will organize the set H into a heap. We will require that for any $i \in \{2,3,\cdots,8\}$ the inequality $a_i \le a_{\lfloor \frac{i}{2} \rfloor}$ is satisfied.

The assignment is made $i = 2$. The inequality $a_2 \le a_{\lfloor \frac{2}{2} \rfloor} = a_1$ is false. In this case a_2 is exchanged with a_1 and we obtain: $H = \{20,3,44,47,11,7,2,37\}$. The comparisons are resumed:

The assignment is made $i = 2$. The inequality $a_2 \le a_{\lfloor \frac{2}{2} \rfloor} = a_1$ is true.

The assignment is made $i = 3$. The inequality $a_3 \le a_{\lfloor \frac{3}{2} \rfloor} = a_1$ is false. In this case a_3 is exchanged with a_1 and we obtain: $H = \{44,3,20,47,11,7,2,37\}$. The comparisons are resumed:

The assignment is made $i = 2$. The inequality $a_2 \le a_{\lfloor \frac{2}{2} \rfloor} = a_1$ is true.

The assignment is made $i = 3$. The inequality $a_3 \le a_{\lfloor \frac{3}{2} \rfloor} = a_1$ is true.

The assignment is made $i = 4$. The inequality $a_4 \le a_{\lfloor \frac{4}{2} \rfloor} = a_2$ is false. In this case a_4 is exchanged with a_2 and we obtain: $H = \{44,47,20,3,11,7,2,37\}$. The comparisons are resumed:

The assignment is made $i = 2$. The inequality $a_2 \le a_{\lfloor \frac{2}{2} \rfloor} = a_1$ is false. In this case a_2 is exchanged with a_1 and we obtain: $H = \{47,44,20,3,11,7,2,37\}$. The comparisons are resumed:

The assignment is made $i = 2$. The inequality $a_2 \le a_{\lfloor \frac{2}{2} \rfloor} = a_1$ is true.

The assignment is made $i = 3$. The inequality $a_3 \le a_{\lfloor \frac{3}{2} \rfloor} = a_1$ is true.

The assignment is made $i = 4$. The inequality $a_4 \le a_{\lfloor \frac{4}{2} \rfloor} = a_2$ is true.

The assignment is made $i = 5$. The inequality $a_5 \le a_{\lfloor \frac{5}{2} \rfloor} = a_2$ is true.

The assignment is made $i = 6$. The inequality $a_6 \le a_{\lfloor \frac{6}{2} \rfloor} = a_3$ is true.

The assignment is made $i = 7$. The inequality $a_7 \le a_{\lfloor \frac{7}{2} \rfloor} = a_3$ is true.

The assignment is made $i = 8$. The inequality $a_8 \le a_{\lfloor \frac{8}{2} \rfloor} = a_4$ is false. In this case a_8 is exchanged with a_4 and we obtain: $H = \{47,44,20,37,11,7,2,3\}$. The comparisons are resumed:

The assignment is made $i = 2$. The inequality $a_2 \le a_{\lfloor \frac{2}{2} \rfloor} = a_1$ is true.

The assignment is made $i = 3$. The inequality $a_3 \le a_{\lfloor \frac{3}{2} \rfloor} = a_1$ is true.

The assignment is made $i = 4$. The inequality $a_4 \le a_{\lfloor \frac{4}{2} \rfloor} = a_2$ is true.

The assignment is made $i = 5$. The inequality $a_5 \le a_{\lfloor \frac{5}{2} \rfloor} = a_2$ is true.

The assignment is made $i = 6$. The inequality $a_6 \le a_{\lfloor \frac{6}{2} \rfloor} = a_3$ is true.

The assignment is made $i = 7$. The inequality $a_7 \le a_{\lfloor \frac{7}{2} \rfloor} = a_3$ is true.

The assignment is made $i = 8$. The inequality $a_8 \le a_{\lfloor \frac{8}{2} \rfloor} = a_4$ is true.

It follows that for any $i \in \{2,3,\cdots,8\}$ the inequality $a_i \le a_{\lfloor \frac{i}{2} \rfloor}$ occurs. The set H organized in a heap is: $H = \{47,44,20,37,11,7,2,3\}$.

The binary tree associated with the heap set $H = \{47,44,20,37,11,7,2,3\}$ was generated with the application sagem of platform cocalc:

```
g=DiGraph({0:[2,1],1:[4,3],2:[6,5],3:[7]})
g.vertices()
g.relabel({0:'47',1:'44',2:'20',3:'37',4:'11',5:'2',6:'7',7:'3'})
g.vertices()
g.show(layout='tree', tree_root='47')
```

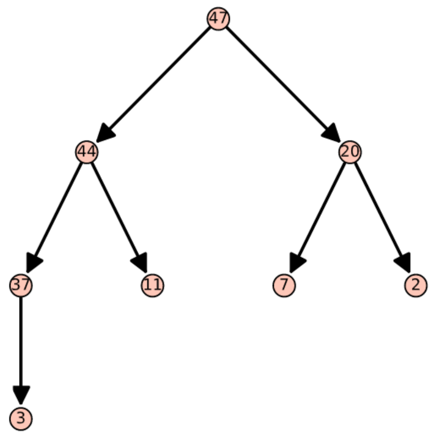

Figure 1 The binary tree associated with the $H = \{47,44,20,37,11,7,2,3\}$ heap

4. Implementation in the C++ language

```
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
int * aloc(int n)
{
        int *p;
        p=(int*)malloc(n*sizeof(int));
        return p;
}
void readv(int n, int *a)
```

```
{
        int i;
        for(i=1;i<=n;i++)
        {
        printf("a[%d] = ",i);
        scanf("%d",&a[i]);
        }
}
void writev(int n,int *a)
{
        int i;
        for(i=1;i<=n;i++)
        printf(" a[%d] = %d ",i,a[i]);
        printf("\n");
}
void heap(int n, int *a)
{
        int heap=0,i,temp;
        while(heap==0)
        {
                for(i=2;i<=n;i++)
                {
                if(a[i]<=a[i/2])
                {
                heap=1;
                continue;
                }
                else
                {
                        temp=a[i];
                        a[i]=a[i/2];
                        a[i/2]=temp;
                        printf("\n a[%d] is exchanged with a[%d] \n\n",i,i/2);
                        writev(n,a);
                        i=n+1;
                        heap=0;
                }
                }
        }
        printf("\n The heap is : \n\n");
        writev(n,a);
}
int main()
{
        int n,*a;
        printf(" n = ");
        scanf("%d",&n);
        a=aloc(n);
        printf("Type the values \n\n");
        readv(n,a);
        printf("The values are : \n\n");
        writev(n,a);
        heap(n,a);
        getch();
}
```

The results generated by the program for $n = 8$ and the set $H = \{3,20,44,47,11,7,2,37\}$ are:

```
Type the values

a[1] = 3
a[2] = 20
a[3] = 44
a[4] = 47
a[5] = 11
a[6] = 7
a[7] = 2
a[8] = 37

The values are :
a[1] = 3  a[2] = 20  a[3] = 44  a[4] = 47  a[5] = 11  a[6] = 7  a[7] = 2  a[8] = 37
a[2] is exchanged with a[1]
a[1] = 20  a[2] = 3  a[3] = 44  a[4] = 47  a[5] = 11  a[6] = 7  a[7] = 2  a[8] = 37
a[3] is exchanged with a[1]
a[1] = 44  a[2] = 3  a[3] = 20  a[4] = 47  a[5] = 11  a[6] = 7  a[7] = 2  a[8] = 37
a[4] is exchanged with a[2]
a[1] = 44  a[2] = 47  a[3] = 20  a[4] = 3  a[5] = 11  a[6] = 7  a[7] = 2  a[8] = 37
a[2] is exchanged with a[1]
a[1] = 47  a[2] = 44  a[3] = 20  a[4] = 3  a[5] = 11  a[6] = 7  a[7] = 2  a[8] = 37
a[8] is exchanged with a[4]
a[1] = 47  a[2] = 44  a[3] = 20  a[4] = 37  a[5] = 11  a[6] = 7  a[7] = 2  a[8] = 3
The heap is :
a[1] = 47  a[2] = 44  a[3] = 20  a[4] = 37  a[5] = 11  a[6] = 7  a[7] = 2  a[8] = 3
```

5. Conclusions and further developments

In this paper we've automatically organized a lot in a heap. Practical situations frequently require the choice of one or more of a dynamic set of values which meet certain conditions. For example, a company is looking to honor, first and foremost the most cost-effective orders. It is therefore necessary, as an appropriate dynamic data structure, to provide with minimum "effort" of processing, the information required by a certain optimal criterion. I see that it is about selecting information from a volume of data, organized according to an established criterion. The next step consists in the automatic generation of the binary tree associated with the obtained pile. The authors aim to write the code needed to generate the associated binary tree.

References

[1] Cormen T., Leiserson C., Rivest R. – Introducere în algoritmi, Computer Libris Agora, 2000.

[2] Horowitz E., Sahni S. - Fundamentals of Computer Algoritms, Computer Science Press, 1985.

[3] Knuth E., D. – Tratat de programarea calculatoarelor, vol. 1, Algoritmi fundamentali, Editura Tehnică, Bucureşti, 1974.

[4] Knuth E., D. - Tratat de programarea calculatoarelor, vol. 2, Sortare şi căutare, Editura Tehnică, Bucureşti, 1976.

[5] Stroustrup B.- The C++ Programming Laguage Ed.Addison – Wesley 1996.