A New Algorithm MFS and Testifying for Binary Searching

Fan Ce, Huang Hongtao
Faculty of Information Science and Technology, Guangdong University of Foreign Studies, GuangZhou, , China
fancemail@yahoo.com.cn

Abstract: BS (Binary Searching) is a known basic technique for keys searching in a sorted list. Fibonacci searching is another binary searching which splits remainder of the list by adopting no equilibrium criterion, or irregular partition. For MFS(Minimal Fibonacci Searching), because each shift distance is always the shorter F(m-2) of F(m)=F(m-1)+ F(m-2) for i=1,2,…,k, if within f(m-1), shift distance is F(m-3) based on F(m-1)=F(m-2)+ F(m-3), and if within F(m–2), similarly, it is F(m-2)=F(m-3)+F(m-4), so that it's movement is shorter. This paper shows the irregular partition of Fibonacci and its PASCAL-likely algorithm, and also indicates, from the view point of searching length, Fibonacci is superior to the general binary searching, and also mainly in two ways that the algorithm of MFS is superior to that of BS, one just addition and subtraction to complete partition of sorted lists, and the other Min shift to search sorted lists. The latter is important if n is big enough and data is stored in file storage.

Keyword: sort; binary searching length; movement; irregular partition

.

1 Intr oduction

Binary searching is a standard technique known for linear searching in sorted lists for a given key[1][2][5][6],and its searching always attempt to aim at the middle key in reminder of sorted lists. Since binary searching itself does not implicate using the same partition, we call the binary searching above mentioned as BS, while imply the generalized binary searching as partition searching. Fibonacci searching using Fibonacci numbers as another partition criterion is superior to BS in the shortest movement distance. A Fibonacci's algorithm is shown in section 2, shift analysis for BS and MFS is shown in section 3 and conclusion in section 4.

2 A Searching Algorithms of Fibonacci

To facilitate discussion, we use symbols as follows before describing the algorithm:

F_i ith Fibonacci number, also $F_i= F_{i-1}+F_{i-2}(i\geq 2, F_0=F_1=1)$

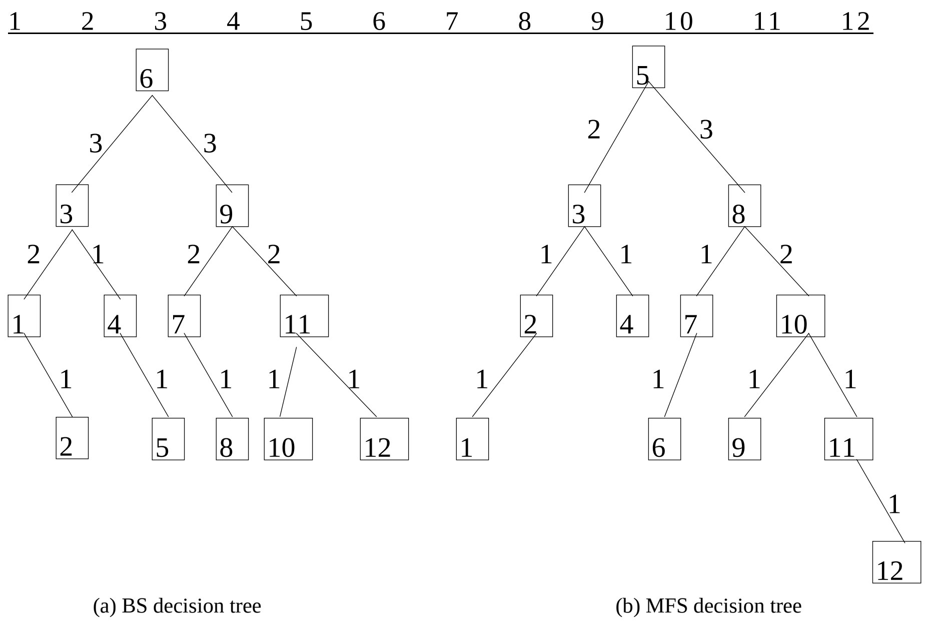

Figure 1. BS and MFS decision trees of a sorted list (digits on edges are shift distance) for n=12

n denotes the length of sorted list（$F_m=F_{m-1}+F_{m-2}=n$）

k =key；k[i] as ith key in the list

We define the shortest Fibonacci searching algorithm as MFS (Minimal Fibonacci Searching) for movement distance of searching. In any Fibonacci searching, the size of sublist pair partitioned always forms two successive Fibonacci numbers which are expressed as F_{i-1} and F_{i-2}. In MFS, the algorithm mainly determines the position where length of one of partitioned sublist pair is the smaller F_{i-2} that is current position close to searching. Hence, since searching shift is only F_{i-2} unit not F_{i-1} in each step, searching movement keep the smallest shift. the MFS algorithm described is as follows:

```
procedure MFS(T,n,a,k)
  begin
    a:=$F_{k-2}$;p:=$F_{k-2}$;q:=$F_{k-3}$;r:=$F_{k-4}$;sw=1;
    while a$\neq$0 do
      case
        k=k(a): return(a);
        (k<k(a) and sw=1)or(k>k(a) and sw=-1):
          if p=1 then a:=0
          else[sw:=-sw;a:=a+sw$\times$r;p:=r;
               q:=q-r;r:=p-q];
        (k<k(a) and sw=-1)or(k>k(a) and sw=1):
          if q=0 then a:=0;
          else[a:=a+sw$\times$q;p:=q;q:=r;
               r:=p-q]
      end
  end
```

where T is a list searched for key k, switch variable sw represents sub trees, of which is right, or is being searched left(see Figure 1). We suppose that searching position is set left in lists initially. So, sw is set initially 1 which represents right sub tree for the beginning of a given list, while sw=-1 represents the left sub tree being searched for the former tree. Figure 1 is the binary decision trees from BS and MFS.

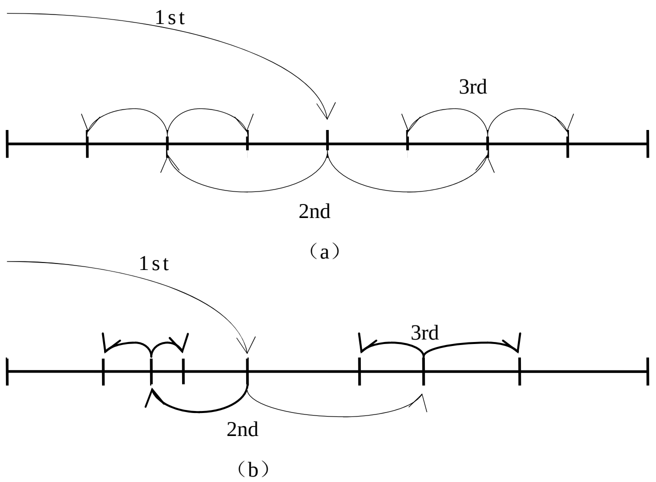

Figure 2. Searching sequence (a) BS (b) MFS (only initial 3 times are shown)

Figure 2 shows how searching is made in BS and MFS(only 3 times are shown).Searching in the algorithm always start from the left in a given list. In the given list, keys are sorted from left to right by ascension. Though Figure 2 represents only first three times in searching procedure, it clearly shows that MFS is superior to BS, that is, searching length (shift distance) keep the least. Their shift distance BS=18, MFS=15 respectively for a given decision tree in Figure 1.

3 Movement Analysis of BS and MFS

In this section, we show general description of MFS superior to BS. Let the size of sorted lists=n(suppose some f(m)).

3.1 The sam e Length Movem ent on BS in Each Step

For BS, remainder lists are divided into the same two parts. Initial movement is n/2 (responding to the root node of a tree), then two binary partition occur after first comparesion and also two movement so, of which distance is $n/2^2$. hence, total movement distance=$2\times n/2^2$. By parity of reasoning, each movement distance is $n/2^k$ on ith level(based on binary decision trees) and there are 2^{k-1} shifts, so that total movement distance= $2^{k-1}\times n/2^k$ on ith level. For level i, each shift is as follows:

1: $2^0\times n/2$

2: $2^1\times n/2^2$

3: $2^2\times n/2^3$

…

i: $2^{i-1}\times n/2^i$

…

the shift distance for all levels is $B(k)=2^0\times n/2 +2^1\times n/2^2+... +2^{i-1}\times n/2^i+...+2^{k-1}\times n/2^k=n/2+n/2 +...+n/2=k\times n/2$

where k as times of binary partition, or deep of trees.

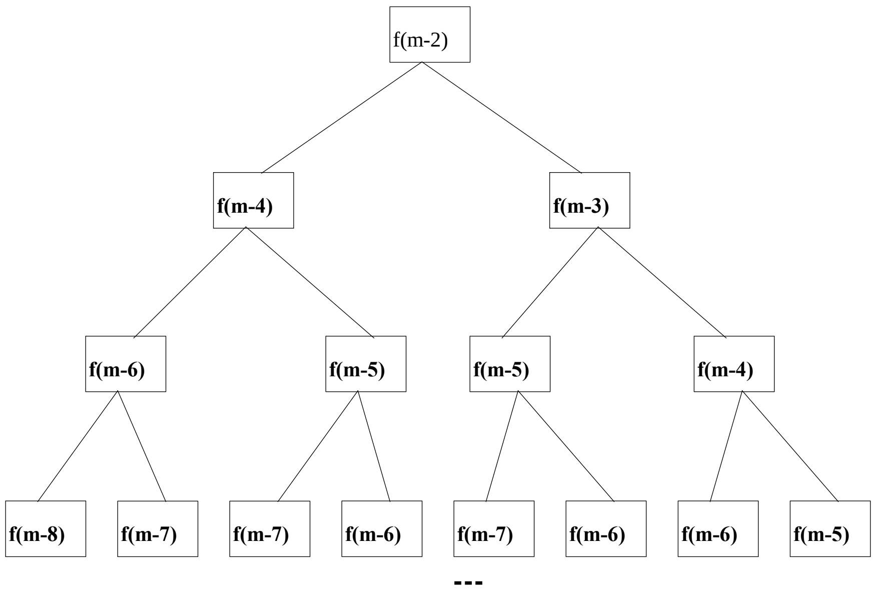

Figure 3. Functions expression of a MFS partition tree (only first 3 times)

3.2 Fibonacci Trees of Min Movement

For MFS, because each shift distance is always the shorter F(m-2) of F(m)=F(m-1)+ F(m-2) for i=1,2,…,k, if within f(m-1), shift distance is F(m-3) based on F(m-1)=F(m-2)+ F(m-3), and if within F(m–2), similarly, it is F(m-2)=F(m-3)+F(m-4) (see Figure 3), so shift on every level is

1: F(m-2)

2: F(m-3)+F(m-4)

3: F(m-4)+F(m-5)+F(m-5)+F(m–6)

---

i: F(m-i- 1)+F(m-i-2)+F(m-i-2)+ F(m-i -3)+...

Owing to F(m-3)+F(m-4)=F(m-2) and F(m-4)+ F(m-5)+F(m-5)+F(m-6)=F(m-3)+F(m-4) =F(m-2), total of shift distance always notates F(m-2) on any level, in this way, MFS $=k\times F(m-2)$, or the definition $F(k)=k\times F(m-2)$.

3.3 F(k) ≤B(k)

There is a problem whether MFS less than BS for each shift. $F(k)\le B(k)$ if $F(L_i)\le B(L_i)(1\le L\le k)$ for any ith shift, but when n is big enough, not always MFS less than BS. We take n=377=F(m)=F(m-1)+F(m-2) for example (m=13) and analyses shift distance based on the worst instance.

0	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	2	3	5	8	13	21	34	55	89	144	233	377

3.3.1 Keys i n T wo E nds (Di gits Unl ined are $MFS(k_i)>BS(k_i)$)

1). →shift direction

BS: 188,94,47,23,11,5,2,1

MFS: 144,89,<u>55,34,21,13</u>,5,2,1

2).$\leftarrow$ shift direction

BS: the same as $\rightarrow$

MFS: 144,55,34,21,13,5,2,1

3.3.2 Keys in Middle

BS: the same as $\rightarrow$(owing to binary searching, it is the same as in two ends)

MFS: because $\rightarrow$ shift direction probably occurs, there exists $\geq$BS in recording to $\rightarrow$ analysis of shift direction.

3.3.3 F(k) $\leq$B(k)

As mentioned above, there exists $B(k_i)<F(k_i)$ sometime for any i, but always $B(k)\geq F(k)$ for total of shift distance, or average shift distance. The cause that $B(k)\geq F(k)$ is $B(k)=k\times n/2$, and $F(k)=k\times F(m-2)$, so testify only $n/2 \geq F(m-2)$.

Owing to $F(m)=F(m-1)+F(m-2)=n$, $n/2=(F(m-1)+F(m-2))/2$, and when $k\geq 1$, there is always $F(m-1)\geq F(m-2)$. So $n/2=(F(m-1)+F(m-2))/2\geq(2F(m-2))/2=F(m-2)$ is tenable. When k is big enough (or n bigger), then will be $n/2> F(m-2)$, that is, $B(k)>F(k)$.

4 Conclusions

It mainly in two ways that the algorithm of MFS is superior to that of BS, one just addition and subtraction to complete partition of sorted lists, and the other Min shift to search sorted lists. The latter is important if n is big enough and data is stored in file storage. Because searching is a mechanical movement, total searching length becomes the evaluation of time. In MFS, owing to avoiding massive movement (see Figure 2) and attempting to search the key close to the current position, total searching distance is less than BS, so that MFS is faster than BS. It is very important to access sorted lists stored in file storage, or database.

Acknowledgments

This work is supported by the Natural Science Foundation of Guangdong University of Foreign Studies under Grant No. GW2006-TB-012

References

[1] D. E. knuth, The Art of computer Programming, Volumn 3/Sorting and Searching [M], Addison-Wesley Publishing Company, Inc. 1973.

[2] A. V. Aho, J. E. Hopcroft, and J. D. Uuman, The Design and Analysis of Computer Algorithms [M], Addison-wesley, Reading mass, 1974.

[3] E. Escortt Adrin, Binary Huffman Equivalent Code, IEEE Transactions on Information Theory [M], 1998, 44(1), 346–351.

[4] Wang Guangfang and Cao Lanbin, Data Structure[M], Publishing House of Hu Nan Science and Technology, 1983.

[5] A. V. Aho, J. E. Hopcroft, and J.D. Uuman, Data Structure and Algorithms[M], Academic Publishing House, 1987.

[6] Cao Xinpu, The Design and Analysis of Computer Algorithms [M], Publishing House of Hu Nan Science and Technology, 1984.

Part 3

Digital Signal Processing

第三部分

数字信号处理