Deepak Abhyankar, Maya Ingle / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com

Vol. 1, Issue 3, pp.446-448

Elements of Dynamic Programming in Sorting

Deepak Abhyankar* Maya Ingle**

* Lecturer School of Computer Science, Devi Ahilya University Indore India
** Prof. School of Computer Science, Devi Ahilya University Indore India

Abstract

Dynamic programming is an effective algorithm design method. Sorting is believed to be an unusual area for dynamic programming. Our finding is contrary to this conventional belief. Though it appears that classical sorting algorithms were designed using bottom up design approach, but we have found the evidence which suggests that some classical sorting algorithms can also be designed using Dynamic programming design method. Even the development of classical Merge algorithm shows elements of dynamic programming. This paper finds that development of sorting algorithms can be looked from an entirely different point of view. This work will reveal some new facts about the design or development of some key sorting algorithms. It was discovered that this new interpretation gives a deep insight about the design of some fundamental sorting algorithms.

1. Introduction

Dynamic programming is the most fundamental algorithm design approach and sorting is a principal computer science problem. Surprisingly Dynamic programming paradigm is absent from the majority of the sorting literature [1, 2, 3, 4]. On the other hand Divide and Conquer deservingly receives the credit for the development of quick sort and merge sort. Divide and Conquer is a great algorithm design paradigm and we have nothing against it. Some key sorting algorithms can be shown to have the elements of dynamic programming. This study shows that Dynamic programming has a role in the development of key sorting algorithms. It was observed that Heapsort and Insertion sort definitely show the aspects of dynamic programming. These sorting algorithms are such a perfect and clean instances of Dynamic programming design method that it is an arduous task to recognize the elements of Dynamic programming in their design. Our study seems to have developed valuable insight about sorting and Dynamic programming.

2. Dynamic Programming and its Applicability

Dynamic programming paradigm is the cornerstone of the algorithm design theory. It has been applied in a wide variety of situations. Dynamic programming solves problems by combining the solutions to sub problems. Divide and Conquer works in a similar way. Difference lies in the nature of sub problems. In the case of Divide and Conquer sub problems are independent but Dynamic programming combines overlapping sub problems. Overlapping sub problems is the litmus test for the applicability of Dynamic programming [4].

3. Dynamic Programming and Heapsort

Heapsort is one of the influential sorting algorithms. There are three critical routines (Heapify, BuildHeap and Heapsort) in Heapsort algorithm[4]. Dynamic programming seems to be present in the Heapify routine. Here we present a version of Heapify where dynamic programming is apparent.

```
void Heapify(Data* a, int i)
        {
                if(a[i].Heap==0)
                        {
                        int l = Left(i);
                        int r = Right(i);
                        int largest=i;
                        if(l<=HeapSize)
                                {
                                Heapify(a,l);
```

```
                    if(a[l].Info>a[i].Info))
                    largest = l;
                }
        if((r<=HeapSize)
                {
                Heapify(a,r);
                if (a[r].Info>a[largest].Info))
                largest = r;
                }
        if(largest!=i)
                {
                        swap(a,i,largest);
                        a[largest].Heap=0;
                        Heapify(a,largest);
                }
        a[i].Heap=1;
        }
}
```

It was observed that routine Heapify contains overlapping subproblems which directs us towards Dynamic programming paradigm. If we use bottom up dynamic programming we will end up designing classical Heapify function which is quite efficient.

4. Dynamic Programming and Insertion Sort:

Insertion Sort is an effective sorting algorithm for sorting small size arrays. We find that Insertion sort is a cleanest application of Dynamic programming. Here we start with a recursive sorting algorithm and after refinements we will arrive finally at classical Insertion sort algorithm. It is interesting to note that following recursive algorithm can be transformed into the tiniest sorting algorithm [5].

```
        void Sort(int* a, int n)
`               {
                if(n>0)
                        {
                                Sort(a,n-1);   //  Statement A
                                if(a[n]>=a[n-1]) return;
                                swap(a,n-1,n);
                                Sort(a,n-1);  //  Statement B
                        }
                }
```

It can be observed that statement A and statement B leads to an overlap of very high degree. That forces us to apply dynamic programming.

```
        void  Sort(Data* a, int n)
                {
                if(n>0)
                        {
                                if(a[n].Sorted!=1)
                                {
                                Sort(a,n-1);   //  Statement A
                                if(a[n]>=a[n-1])
                                        {
                                        a[n].Sorted=1;
                                        return;
                                        }
                                swap(a,n-1,n);
                                a[n-1].Sorted=0;
                                Sort(a,n-1);  //  Statement B
                                a[n].Sorted=1;
```

```
                    }
                }
            }
```

It was found that routine sort involves overlapping sub problems which dictate us towards dynamic programming paradigm. If we use bottom up dynamic programming we will finish up designing classical Insertion Sort function which is quite efficient.

5. Results

We find that Dynamic programming has the potential to design elegant sorting algorithms. In fact this paper shows that Heap sort and Insertion sort can also be designed using dynamic programming method. Same can be shown true for Tree Selection sort. It is evident from above mentioned algorithms that if return type of a function is void bottom up dynamic programming approach will require lesser amount of memory because lookup table will be eliminated. Lookup table will still be present in the top down approach. Both the Heap Sort and Insertion sort are in place sorting algorithms which are contrary to the folk belief that dynamic programming may be memory consuming.

6. Conclusion

Though in the past Dynamic programming did not take any credit for the development of Heap sort or Insertion sort the prior deserves a lot more. Evolution of algorithms is an understudied area where future work can produce a lot of surprising results. We believe that more algorithms belong to the dynamic programming than it is actually credited with. Last but not the least if return type of a function is void and applicability of dynamic programming exist then we will get elegant memory efficient solutions.

References

[1] D. E. Knuth, *The Art of Computer Programming,* Vol. 3, Pearson Education, 1998.

[2] S. Baase and A. Gelder, *Computer Algorithms:Introduction to Design and Analysis,* Addison-Wesley, 2000.

[3] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of computer algorithms.* Addison-Wesley, 1974.

[4] T. Cormen, C. Leiserson, R. Rivest, C. Stein, *Intoduction to Algorithms,* 2nd ed. MIT Press, 2001.

[5] M. D. Mcilroy, *The tiniest C sort function?,* www.cs.dartmouth.edu/~doug/tinysort.html