**Theory on Yaoguang's Split Peak Defect**

**Mr. Yaoguang. Luo**

**Liu yang deta software development limited company, hunan,china**

**313699483@qq.com**

***Outline：****In the common software development factory, engineer always did more and more interactions with data structure and math algorithms. Especially in the recursion, convolution, sort and generic loops, scientist likes to find a simple, more sufficiently and alignment way to face the project requirements with the large association. For instance me, I really got a real world problem at this domain while I use quicksort, also for other project such like DETA parser. What is the peak array split defect? How does it count the real world problems? Why need find it and how to get the nice solution? Cool, This paper will cause an implementation about our goals, ok now, keep forward to the context where I talking as below, thanks.*

**Peak, Array, Split, Defect, Recursion, Convolution, Sort, Generic**

**Goal one: Quicksort Yaoguang.Luo 4D**

**1 Details：**

1.1 For example the array input as below where we gave 11 digits.

11

10

9

8

7

6

5

4

3

2

1

1.2 The first split, we could see the digit-6 will auto arranged to the right part.

11

10

9

8

7

6

5

4

3

2

1

1.3 And the second split, we may see the digit-3 will be auto arranged to the right part

11

10

9

8

7

6

5

4

3

2

1

1.4 The third split, we may see the digit-4.7.10 will be auto arranged to the right part

11

10

9

8

7

6

5

4

3

2

1

**2 Thinking:**

After the split array showing, we could see clear that the big problem about the asymmetry defect, as I did an annotation of N, so the i of N will absolutely find a n/pow（2，i）value points, as an insufficiency asymmetry defect model, I fall in thinking…if I do any compute theory as the same with this model style, for example in the recursion or inner loops, it will autonomic separate to the 2 different process way, it necessary to do indifferent flows.

**3 Problems:**

So, after the above thoughts, I may get any flashes, First, the even and odd digits both are asymmetry while in the Differential loops. For this noise, I defined as (Tinoise Peak) Second，once we did a split compute under this model, it must get more unfair sets. I defined as (Tinsets defect). Third, if this model almost in the messy and timer data system, it will catch more time and asserts wastes or exceptions.

**4 Solutions:**

For the god like, I find three solutions while I currently enrolled in my projects. First: computer logic acceleration, at least it can avoid the waste of the compute by using inner process optimism. -- To avoid the deep recursion. Second, reduce the compute sets. For any less memory system, we may reduce more and more memory garbages after we reduce the inner register or temp value sets. Third, we may make an optimization of the function logic where to instead the old complex functions. Those ways include the condition, algorithm, method or discrete optimization. End, we may use mathematics of double differential, deep definition, acquisition or polynomial to get the solutions.

**5 True Instances**

Let me show the algorithms here,

**private** **int** **partition**(**int**[] a, **int** lp, **int** rp) {

**int** x= a[lp]>= a[rp]? a[lp]: a[rp];

**int** lp1= lp;

**while**(lp1< rp){

**while**(!(a[lp1]>x|| lp1>=rp)) {

lp1++;

}

**while**(a[rp]>x){

rp--;

}

**if**(lp1<rp){

**int** temp=a[rp];a[rp]=a[lp1];a[lp1]=temp;

}

}

a[lp]=a[rp];a[rp]=x;

**return** rp;

}

From this code: in a common quicksort way, the recursion based on the average deep split, suppose the initial array length is N{1,2,3…n} is an Odd, so the separate two arrays will cause an asymmetry defect, those timer asymmetry compute peak collection will cause more and more probability problems such like jam, lock, time waste and heap increment.

The odd peak binary split as below：

Defect peak

How to avoid those timer distinction peaks? I go more absolutely research where focus on these problems, first, differential flows. This flash is not suitable for here, May good for the DETA parser, I will show you later. Second, compute acceleration. Yep, this is a good way, for example find the big X as the code blew, it will cause the while loop ability accelerations.

**int** x= a[lp]>= a[rp]? a[lp]: a[rp];

Third, De Morgan condition differential as the code below, it will cause the condition ability accelerations.

**while**(!(a[lp1]>x|| lp1>=rp)) {

At last but the least, value reduce, code optimization both are very important way of the peak avoid filter.

**Goal Two: DETA parser**