A Study on Adjustable Work Space Algorithms

Takahiro Seii (1110033)

School of Information Science,
Japan Advanced Institute of Science and Technology

February 12, 2014

Keywords: Arrangement of Lines,Adjustable Work Space Algorithm,Geometry,3-SUM Problem.

We can use a small device with a lot of memory capacity. The memory capacity is bigger than a desktop PC of a few years ago. Many functions could not execute our requirement on the past but it is possible now. For example, a face recognition and car navigation. Is memory capacity sufficient? An answer is No. A software require that a hardware are a high speed and big capacity. We have to consider memory capacity when we design a program because software use a lot of memory.

For example is to use Big Data. An algorithm needs to calculate in small work space because its data is bigger than a memory capacity. And an input data is a read only data. Because we execute any programs to same big data. If an input data changed by other programs, a result is not correct.

A digital camera and tablet device can calculate the path for users goal and manipulate an image.

It is no wonder because they have big capacity. The device becomes expensive as it becomes a memory capacity is increased. And the device becomes large.

If we can use adjustable work space algorithm, we can make a program and design flexible. Therefore, an algorithm which can run in small work space is very useful for embedded but also general environment.

We design two algorithms for two problems. One is a geometry problem. Geometry is related more closely to geographical information system and computer graphics. The other is a basic problem. Given a set S of n integers, are there $a, b, c \in S$ with $a + b + c = 0$? It's a 3-SUM problem. 3-SUM problem is based on 3-SUM problem class. 3-SUM problem class have many problem.

This paper presents two adjustable work space algorithms. One is to simulate algorithm for 3-SUM problem and the other is an algorithm of arrangement of lines. An algorithm for 3-SUM problem is known $O(n^2)$ time. Unfortunately the algorithm required $O(n)$ work space. But an adjustable work space algorithm for 3-SUM problem runs $O(n^3/(s \log n))$ time when a parameter s specifying the size of work space is given between $O(1)$ and $O(n/\log^2 n)$.

And this paper presents an adjustable work space algorithm for an arrangement of lines. An arrangement of lines is the partition of the plane by a collection of lines. It has many cells. By visiting all the cells one after another we can solve many problems efficiently. For the purpose algorithms such as Topological Sweep and Topological Walk are known. Unfortunately those algorithms required $O(n)$ work space. When a parameter s specifying the size of work space is given between $O(1)$ and $O(n/\log^2 n)$, out algorithm runs in $O(n^3/(s \log n))$ time.

The algorithms use an important data structure which a priority queue for read-only data. The algorithm is designed by Asano, Elmasry and Katajainen. This paper describe the data structure, insertion operation and extraction operation. Its work space is $O(s)$.

If we use its data structure, we can get a sorted read-only array data in $O(n^2/s)$times when a parameter s is between $(\lg n)$ and $O(n/\lg n)$. This is very interesting. Given a read-only array of n integers. How do you get a result of the sort? A simple idea is to use one pointer. First step is to search a minimun element in the array. Next step is to search next element in the array. We repeat n times the same step. All step time is $O(n^2)$ times. We get a result of the sort in $O(n^2/\log n)$ times if we use work space of about one pointer.

A pointer size is $O(\log n)$. Any element in the read-only array can be accessed by its address in constant time independently of n. Further, any

basic arithmetic operation takes constant time. In this paper we use our computational model. Input data are stored in an array which is often just read-only. Although it is not allowed to permute the array elements or modify the contents of any element of any elemement, constant time random access to any array element is possible. Futher, we assume that any basic arithmetic operation takes constant time. An adjustable work space algorithm can use at most some space, each between $O(\log n)$ and $o(n)$,in addtional to the read-only array for input data. Implicit storage consumption required by recursive calls is also considered as a part of work space.

Our model is RAM(Random Access Machine) model. This model require $O(\log n)$bits work space if input size is n. Its work space is minimum because a pointer size of an input array is $O(\log n)$.

We design an adjustable work space algorithm for two problems using the model. 3-SUM problem class have many problems. We think their algorithms can be solved in the almost same. And topological sweep algorithm can solve a lot of problems in the same way. Its algorithm is to scan an arrangement of lines. Our designed algorithm is to scan an arrangement of lines, too. However, a way to scan is different. Our algorithm is not adjustable work space algorithm of topological sweep. We have to carefully consider design algorithm if we design adjustable work space algorithms for some problems, which can be solved by topological sweep algorithm.

Two algorithms use a common data structure. And they use some important technique to design adjustable work space algorithm. One is to copy data structure. Its technique is used in 3-SUM problem algorithm. If we don't use the technique,it can not be used repeatedly a sorted data.

The other technique is to calculate necessary data. Its technique is described to an adjustable work space algorithm for an arrangement of lines. We do not need to save many necessary data if we use the technique. It is possible to reduce many work space by calculating when needed necessary data.

This paper presents two algorithms and some technique for adjustable work space algorithm. And, two problems is basic problems. We hope that it is possible to design some other adjustable work space algorithms when we use our designed algorithms and these techniques.