Advances in Engineering Research (AER), volume 82
2016 International Conference on Engineering and Advanced Technology (ICEAT-16)

The Dynamic Storage Structure of Double-ended Stack

Zhiguo Ren[1, a], Zhengping Zhu[1, b] and Keke Du[2, c]

[1]School of Electronics and Information Engineering, Lanzhou City University, Lanzhou 730070, China;

[2]Institute of Information Technology Application, Lanzhou City University, Lanzhou 730070, China.

[a] ren_zhiguo@qq.com, [b]20470864@qq.com, [c]32034377@qq.com

Keywords: Dynamic Stack; Dynamic extension; Dynamic recycling.

Abstract. In reference [1-4] discuss the stack's shared technology, the most frequently-used in stack's shared technology is the double-ended stack. But traditional double-ended stack always define the maximum storage space of it, which causes the wasting or lacking of storage space. Here we discuss and implement the dynamic storage technology of double-ended stack at first. The dynamic storage structure of the Double-ended stack compared with the traditional structure has following advantages. First, the free space in the dynamic double-ended stack is always kept in a certain range, which makes less waste of storage space. Second, the technology can achieve dynamic expansion of storage space and automatic recovery of storage space when the program keeps running.

1. Introduction

Stack is a kind of important data structures, stack technology is widely used in compiling software and programming. Discussing the structure characteristics and operation characteristics of the stack has an important meaning. In the application of the stack will often need to use the technology that the stack is shared. In this sharing technology, the most commonly used is the sharing of two stacks, which is also called double-ended stack. The traditional double-ended stack is always to allocation the certain storage space M in advance, which makes a shortage or a waste of storage space in the process of program running. General practice is to stop the implementation of the program and modify the value of M. If M is too big, it will cause the waste of storage space and can't achieve the purpose of dynamic expansion, it also can't be done on the automatic recovery of free space. In order to solve this problem, this paper presents a new method to realize the dynamic expansion of the double-ended sequence stack storage space, which solved the problem of the storage space lacking. At the same time, the automatic recovery of the free space is successfully achieved. The concepts and terminologies that no mentioned in the reference [1] and reference [4].

2. Traditional Double-ended Stack Storage Structure

There is a constraint relationship between the two stacks in the traditional double-ended stack: The total numbers of elements in the two stacks can be up to M. If there are more of elements in one of stacks, then there are less of elements in another stack. The sum of the elements in the two stacks can't over M. It mainly uses the characteristics that the position of stack's bottom unchanged, but the position of the stack's top is dynamic changing. Firstly, applying for a shared and one dimensional array space S[M], the bottoms of both stacks are respectively arranged at the two ends of the one-dimensional array, the bottom of the left side stack is S[0] and the bottom of the right side stack is S[M-1]. Due to the dynamic changes of the both stacks' top, which can form a complementary

structure, making the maximum available space for each stack arrived at M. The storage structure of the traditional double-ended stack is defined as follows:

```
typedef struct
{  ElemType Stack[M];
     int top[2];
 }DqStack;
```

In the application of double-end stack, if a data element insert into the stack or delete from the stack, the number of elements stored in the double-ended stack is always less than M-2, and the basic operations can be performed.

But when the data elements are filled with storage space, and there are elements that need to be pushed stack, then the above algorithm can't be solved, which have to appear "overflow" phenomenon. And push stack operation is not successful. Some people think that the M can be amplified to the maximum when defined, but they will find that the greater M, the greater waste of space; Also some people put forward a method, when the space is not enough, stopping program execution, then amplified M. But it's hard to stop program execution in the large-scale systems, once the system is put into running.

3. The Dynamic Storage Structure of Double-ended Stack

In view of the above problems, ElemType can be defined as a pointer type *Stack, rather than arrays, which does not need to set the maximum storage space for the double-ended stack M, but to set the initial value STACK_INIT_SIZE of the double-ended stack. By dynamic memory allocation function malloc, allocating storage space for the pointer, and allocating StackSize to represent the current size of the space. If in program execution appears of such a situation that stack is full, realloc function can be used to dynamically expand the space, making it increase STACK_INCREMENT units. Its new type is defined as follows:

```
# define STACK_INIT_SIZE 5
# define STACK_INCREMENT 5
# define STACK_FREESIZE 10
# define PERCENT 0.5
typedef struct
{   ElemType *Stack;
     int top[2], StackSize;
}DqStack;
```

According to the above definition method, in the initialization of the double-ended stack, need to call the malloc function to allocation the initial space of the double-ended stack. Its description as follows:

```
int InitStack(DqStack *S)
{   S->Stack=(ElemType*)malloc(STACK_INIT_SIZE*sizeof(ElemType));
     if(S->Stack==NULL) return ERROR;
     S->StackSize=STACK_INIT_SIZE;
     S->top[0]=0; S->top[1]=S->StackSize-1;
     return OK;
}
```

Researching on the basic operation of the double-ended stack will be found that only when push stack operation will appear the problem that storage space is not enough to use. If the double-ended stack's elements are less than S->StackSize-2, according to the conventional push stack operation. Otherwise, If there are S->StackSize-2 elements in the double-ended, that is the next place of S->top[0] is S->top[1], which means that the stack needs to expand. At this time, we can expand the storage space of the double-ended stack with realloc function. The specific method of expansion is: Firstly , we need increase storage space from S->StackSize to S->StackSize+STACK_INCREMENT; Then we need to move the data elements, which are from S->top[1]+1...S->StackSize-1 to S->top[1]+STACK_INCREMENT+1...S->StackSize+STACK_INCREMENT-1; Finally, modifying the value of S->StackSize+STACK_INCREMENT for S->StackSize, and modifying the value of S->top[1]+STACK_INCREMENT for S->top[1]. So that the double-ended stack can be dynamic

expansion, and the push operation can be implemented in traditional way. The push stack operation algorithm is described as follows:

```
int Push(DqStack *S,ElemType e, int i)
{   int j,k,newsize;
      ElemType *newbase;
      if(S->top[0]+1==S->top[1])
      {   newsize=(S->StackSize+STACK_INCREMENT)*sizeof(ElemType);
          newbase=(ElemType*)realloc(S->Stack,newsize);
          if(!newbase)  return ERROR;
          S->Stack=newbase;
          k=S-> StackSize+STACK_INCREMENT-1;
          for(j=S->StackSize-1;j>=S->top[1]+1;j--,k--)
              S->Stack[k]=S->Stack[j];
          S->StackSize+=STACK_INCREMENT;
          S->top[1]+=STACK_INCREMENT;
       }
       switch(i)
       {     case 0: S->Stack[S->top[0]]=e; S->top[0]++; break;
             case 1: S->Stack[S->top[1]]=e;S->top[1]--; break;
             default: return ERROR;
       }
       return OK;
}
```

Pop stack operation, Firstly, according to the conventional method, the top element of the stack top element of the first stack is S->Stack[S->top[0]-1] or second, and the top element of the stack is S->Stack[S->top[1]+1] out of the stack, the top of the stack after the stack pointer S->top[0] or S->top[1] changes. And then determine whether the size of the free space in the Double-ended stack is up to a predetermined value STACK_FREESIZE. If reached, it is required to recover some of the free space according to a certain proportion.

A Double-ended stack of capacity S->StackSize=18 has 7 data elements and two top pointers respectively, As shown in figure 1. If the provisions of which the free space STACK_FREESIZE up to 10 of the proportion of PERCENT=50% to recover part of the free space. The number of recycling is hsgs=STACK_FREESIZE*PERCENT. Therefore, when the eighth elements are out of the stack, STACK_FREESIZE to reach 10, then we need to recover space. Specific process is: first move the elemnents from {S->top[1]+1...S->top[1]-hsgs+1} to {S->StackSize-hsgs-1...S->StackSize-1}, then modify the value of S->top[1]-hsgs for S->top[1]; Finally through the realloc function so that the S->Stack refers to the unit was reduced to S->StackSize-hsgs. After the recovery of the Double-ended stack structure as shown in figure 1.

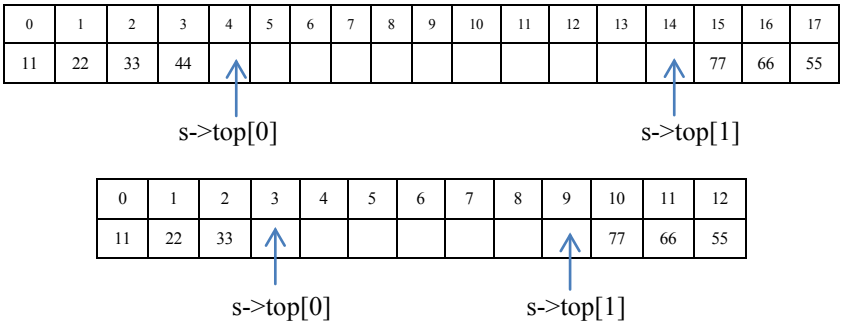

Fig.1 the results of recovery space

The pop stack operation algorithm is described as follows:

```
int Pop(DqStack *S, ElemType *e, int i)
{   int j, k;
      int LEN=S->StackSize+S->top[0]-S->top[1] -1;
      int hsgs=int(STACK_FREESIZE*PERCENT);
      ElemType *newbase;
      switch(i)
      {  case 0: if(S->top[0]==0) return ERROR;
                 *e=S->Stack[S->top[0]-1];S->top[0]--; break;
         case 1: if(S->top[1]==S->StackSize-1) return ERROR;
                 *e=S->Stack[S->top[1]+1]; S->top[1]++; break;
```

```
            default: return ERROR;
        }
        // If the provisions of which the free space STACK_FREESIZE up to 10 of the proportion of PERCENT=50% to recover
    part of the free space.
        if(S->StackSize-LEN>=STACK_FREESIZE)
        {   for(j=S->StackSize-hsgs-1,k=S->StackSize-1;k>=S->top[1]+1;j--,k--)
                S->Stack[j]=S->Stack[k];
            S->top[1]-=hsgs;
            newbase=(ElemType*)realloc(S->Stack, (S->StackSize-hsgs) *sizeof(ElemType));
            if(!newbase) return ERROR;
            S->Stack=newbase;
            S->StackSize-=hsgs;
        }
      return OK;
    }
```

The method of automatic recovery in above, which can be seen as a dynamic operation of the reverse operation for the storage space. Double-ended stack storage structure for dynamic stack depth, take the top element on the stack, waiting for operation with the traditional method of similar sentence.

4. Summary

Through the above discussion, we can see that the dynamic storage structure of the Double-ended stack compared with the traditional structure has following advantages. First, the dynamic storage of the double-end stack expanded only in the case of the full stack, and only when the free space in the stack increased to a certain value, the storage space is recovered. So the free space in the double-ended stack is always kept in a certain range, which makes less waste of space. Second, the technology can achieve dynamic expansion of storage space and automatic recovery of storage space when the program keeps running. Third, in the case that the amount of data elements to be stored is not determined, it can allocate a small initial space on the double-end stack. The technology can be automatically extended or recovered. And we don't worry about waste of storage space ,or storage space is not enough to use the situation.

References

[1] Zhiguo Ren, Data Structure(C Language Description), Science Press, Peking China, 2016.

[2] Thomas H.Cormen, Charles E.Leiserson, Ronald L.Rivest, Clifford Stein. Introduction to Algorithm, the third edition. The MIT Press,2009.

[3] Alfred V.Aho,John E.Hopcroft,and Jeffrey D.Ullman.Data structures and Algorithms. Addison-Wesley, 1983.

[4] Donald E.Knuth. Fundamental Algorithms, volume 1 of The Art of Computer Programming. Addison- Wesley, 1968.Third edition,1997.

[5] Mark Allen Weiss. Data Structures and Algorithm analysis in Java. Addison-Wesley, third edition,2007.

[6]Srba I, Bielikova M. Why is Stack Overflow Failing? Preserving Sustainability in Community Question Answering[J]. IEEE Software, 2016, 33(4):80-89.

[7]Ye D, Xing Z, Kapre N. The structure and dynamics of knowledge network in domain-specific Q&A sites: a case study of stack overflow[J]. Empirical Software Engineering, 2016:1-32.

[8] Donald E.Knuth. Seminumerical Algorithms, volume 2 of The Art of Computer Programming. Addison- Wesley,1969. Third edition,1997.

[9] Donald E.Knuth. Sorting and Searching, volume 3 of The Art of Computer Programming. Addison- Wesley,1973.Second edition,1998.

[10] Don Coppersmith and Shmuel Winograd. Matrix Multiplication via arithmetic progression. Journal of Symbolic Computation,9(3):251-280,1990.

[11] T.C.Hu and M.T.Shing. Computation of Matrix chian products.Part 2, SIAM Journal on Computing, 13(2):228-251,1984.