Exercise 11.1-1

> Suppose that a dynamic set Sis represented by a direct-address table Tof length m. Describe a procedure that finds the maximum element of S. What is the worst-case performance of your procedure?
>
> **Solution:** We can do a linear search to find the maximum element in Sas follows: Pre-condition: table T is not empty; m ∈ Z+, m ≥1. Post-condition: FCTVAL == maximum value of dynamic set stored in T.

```
    FindMax (T, m)
    {
        max= -∞
        for i=1 to m
        {
            if T[i] != NIL && max< T[i]
            max= T[i]
        }
        return max
    }
```

Exercises 11.1-2

> A bit vector is simply an array of bits (0's and 1's). A bit vector of length m takes much less space than an array of m pointers. Describe how to use a bit vector to Represent a Dynamic Set of Distinct Elements with no Satellite Data. Dictionary Operations Should Run in O(1) Time.
>
> **Solution**(I get the result from Rip’s blog, The original text is here for ease of reference.):
>
> Using a bit vector of length m, as our direct-address table to represent the dynamic set.Here m bits tags m slots of the table. The elements in the dynamic set is stored in the direct-address table itself. If slot k is allocated, bit k is 0. If slot k contains no element, bit k is 1.
>
> Bit Vector
>
> The free-space list is often implemented as a bit map or bit vector. If a block is free, the bit is 1. If a block is allocated, the bit is 0. Assume the following blocks are allocated, the rest free: 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 27 .The free-space bit map would be: 001111001111110001100000011100000…
>
> Adavantages of this approach include:
>
> - Relatively simple
> - Efficient to find the first free blocks or n consecutive free blocks on the disk.
>
> Bit maps are useful only when it can be kept in main memory. But, as disks get larger, this is hard to do. 1.3 gigabyte disk with 512 byte blocks would need a bit map of over 310k to track its free blocks. Clustering the blocks in intervals of foul reduces this number to 78k per disk.

Exercises 11.1-3

> Suggest how to implement a direct-address table in which the keys of stored elements do not need to be distinct and the elements can have satellite data. All three dictionary operations (INSERT, DELETE, and SEARCH) should run in O(1) time. (Don't forget that DELETE takes as an argument a pointer to an object to be deleted, not a key.)
>
> **Solution:**可以直接用链表来解决碰撞问题。对于搜索的话可以返回链表的指针，根据题 目的条件来看，对于两个用户提供的两个key相同的object我们是无法区分的，因此只能返回 一组key相同的objecet让用户用自己的方法来区分。

Exercises 11.1-4

We wish to implement a dictionary by using direct addressing on a huge array. At the start, the array entries may contain garbage, and initializing the entire array is impractical because of its size. Describe a scheme for implementing a direct-address dictionary on a huge array. Each stored object should use O(1) space; the operations SEARCH, INSERT, and DELETE should take O(1) time each; and the initialization of the data structure should take O(1) time. (Hint:Use an additional stack, whose size is the number of keys actually stored in the dictionary, to help determine whether a given entry in the huge array is valid or not.)

Solution:一直没想出好的办法来解决这个问题，今天看《算法导论教师手册》有这个题 的解答。觉得它的数据结构的思路比较巧妙：字典中每个元素带有一个索引，表示关于该元 素的的key存储在辅助栈中的位置。而辅助栈用数组实现，因此可以随机访问。辅助栈中存储 所有存在于字典中的元素的key ，于是形成了一个所谓validating cycle。要判断从字典中 直接读取到的元素是不是需要的数据，只需要取出该元素中的索引，判断这个索引是否有效 （大于栈底而小于栈顶），然后通过索引获取栈中的key ,与自己需要的元素的key比较看是 否相等即可以。相等表示该数据是要找的数据，否则不是。显而易见，通过这个数据结构模 型来进行SEARCH, INSERT都为O(1)。至于DELETE 操作，需要将被删除的数据在栈中的key与栈顶存储的key交换后再删除，同样可以达到O(1)，要注意的是删除过程中，需要更新与 原栈顶中key相关元素的索引。