```
'''
Created on Feb 27, 2011

@author: ola
'''
import random,time
_MAX = 500

def create_list(n):
    return [random.randint(0,_MAX) for i in xrange(0,n)]


def mode_compA(nums):
    occs = [nums.count(n) for n in nums]
    m = max(occs)
    mindex = occs.index(m)
    return nums[mindex]

def mode_compB(nums):
    occs = [nums.count(n) for n in nums]
    m = max(occs)
    for n in nums:
        if nums.count(n) == m:
            return n

def mode_compC(nums):
    occs = [nums.count(n) for n in nums]
    for n in nums:
        if nums.count(n) == max(occs):
            return n

def mode_compD(nums):
    mn = min(nums)
    mx = max(nums)
    counts = [0]*(mx-mn+1)
    for n in nums:
        counts[n-mn] += 1
    mindex = counts.index(max(counts))
    return mindex+mn

def time_all(n):
    funcs = [mode_compA,mode_compB,mode_compC,mode_compD]
    nums = create_list(n)
    for f in funcs:
        start = time.time()
        mode = f(nums)
        end = time.time()
        print "%s mode = %d time = %4.3f\n" % (f,mode,(end-start))

if __name__ == "__main__":
    time_all(5000)
```



```
'''
Created on Nov 26, 2012

@author: ola
'''

import random,time,bisect,sys


def mergesort(values):

    def merge(low,mid,high):
        lindex = low
        rindex = mid+1
        temp = []
        while lindex <= mid and rindex <= high:
            if values[lindex] <= values[rindex]:
                temp.append(values[lindex])
                lindex += 1
            else:
                temp.append(values[rindex])
                rindex += 1
        temp.extend(values[lindex:mid+1])
        temp.extend(values[rindex:])
        index = low
        for val in temp:
            values[index] = val
            index += 1

    def dowork(low,high):
        if low < high:
            mid = (low+high)/2
            dowork(low,mid)
            dowork(mid+1,high)
            merge(low,mid,high)
    dowork(0,len(values)-1)

def qsort(values):
    def swap(a,b):
        values[a], values[b] = values[b],values[a]

    def partition(low,high):
        piv_value = values[low]
        piv = low
        for i in range(low+1,high+1):
            if values[i] < piv_value:
                piv += 1
                swap(piv,i)
        swap(piv,low)
        return piv

    def dowork(low,high):
        if low < high:
            pivot = partition(low,high)
            dowork(low,pivot-1)
            dowork(pivot+1,high)

    dowork(0,len(values)-1)


def get_number():
    return random.randint(0,sys.maxint)

def create_list(n):
    return [get_number() for i in xrange(0,n)]

def binary_search(values,target):
    low = 0
    high = len(values)-1
```

```
    while low <= high:
        mid = (low+high)/2
        if values[mid] == target:
            return mid
        elif values[mid] < target:
            low = mid+1
        else:
            high = mid-1
    return -1

def linear_search(values,target):
    for i in range(len(values)):
        if values[i] == target:
            return i
    return -1


def make_lists(size,search_size):
    nums = create_list(size)
    search = create_list(search_size/2)
    search.extend(nums[:search_size/2])
    return nums,search

def timings(values,search_for, func):
    start = time.time()
    found = 0
    for x in search_for:
        ret = func(values,x)
        if ret != -1:
            found += 1
    end = time.time()
    print "%s:size = %d\tsearch = %d\tfound=%d\ttime = %4.3f" % (func,len(values), len(search
_for),found,(end-start))

if __name__ == '__main__':
    LIST_SIZE = 10000
    for sval in range(1000,10001,1000):
        values,search = make_lists(LIST_SIZE,sval)
        qsort(values)
        timings(values,search,binary_search)
```



```
'''
Created on Feb 27, 2011

@author: ola
'''

import random,time,bisect,sys

def get_number():
    return random.randint(0,sys.maxint)

def create_list(n):
    return [get_number() for i in xrange(0,n)]

def linear_search(search,nums):
    f = 0
    for i in search:
        if i in nums:
            f += 1
    return f

def binary_search(search,nums):
    f = 0
    nums.sort()
    for i in search:
        index = bisect.bisect_left(nums,i)
        if index == len(nums) or nums[index] == i:
            f += 1
    return f

def timings(size, search_size):
    nums = create_list(size)
    search = create_list(search_size/2)
    search.extend(nums[:search_size/2])
    funcs = [linear_search,binary_search]
    for f in funcs:
        start = time.time()
        found = f(search,nums)
        end = time.time()
        print "%s:size = %d\tsearch = %d\tfound=%d\ttime = %4.3f" % (f,len(nums), len(search)
,found,(end-start))

if __name__ == "__main__":
    timings(10000,5000)
    timings(10000,10000)
    timings(20000,10000)
    timings(20000,20000)
```

```
'''
Created on Feb 27, 2011

@author: ola
'''
import random,time


def create_word():
    ls = list("etaoins")
    random.shuffle(ls)
    return ''.join(ls)

def create_list(n):
    return [create_word() for i in xrange(0,n)]

def diffA(words):
    return len(set(words))

def diffB(words):
    d = 0
    for i,word in enumerate(words):
        if words[:i].count(word) == 0:
            d += 1
    return d

def time_all(n):
    funcs = [diffA, diffB]
    nums = create_list(n)
    for f in funcs:
        start = time.time()
        diffs = f(nums)
        end = time.time()
        print "n = %d %s diffs = %d time = %4.3f\n" % (n,f,diffs,(end-start))

if __name__ == "__main__":
    time_all(2000)
    time_all(4000)
    time_all(5000)
    time_all(10000)
```