import random, time

def make_strand(size):
    dna = "cgat"
    str = ""
    for i in xrange(0, size):
        str += dna[random.randint(0, 3)]
    return str

def cgratio(strand):
    """return # of c's and g's in strand"""
    cg = 0
    for nuc in strand:
        if nuc == 'c' or nuc == 'g':
            cg += 1
    return cg

def max_index(strand, windowSize):
    """return index in strand of max cg ratio in a sub-strand/窗口
    whose length is windowSize"""
    index = 0
    max = 0
    for i in range(0, len(strand) - windowSize + 1):
        cg = cgratio(strand[i:i + windowSize])
        if cg > max:
            max = cg
            index = i
    return index

def running_max(strand, windowSize):
    gc = 0
    counters = []
    for nuc in strand:
        counters.append(gc)
        if nuc == 'c' or nuc == 'g':
            gc += 1
    counters.append(gc)
    index = 0
    max = 0
    for i in range(windowSize, len(strand) + 1):
        diff = counters[i] - counters[i - windowSize]
        if diff > max:
            max = diff
            index = i
    return index-windowSize

def timings(size, window_range):
    strand = make_strand(size)
    funcs = [max_index, running_max]
    print "strand size = %d" % (size)
    for f in funcs:
        print "function = %s" % (f)
        for win in window_range:
            start = time.time()
            i = f(strand, win)
            end = time.time()
            print "win = %d index = %d time = %4.3f" % (win, i, (end - start))

if __name__ == "__main__":
timings(10000, xrange(1000, 3100, 100))
```python
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 '%d%ssearch = %dfound=%dtime = %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)
```

```python
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 '%d%sdiffs = %dtime = %4.3f' % (n, f, diffs, (end - start))

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

---

**Monday February 28, 2011**

---

**SearchTimings.py, SetTimings.py**