Asymptotic Analysis

1. Programs A and B are analyzed and are found to have worst-case running times no greater than 150N log N and N^2, respectively.

Answer the following questions (the answer may be "cannot determine based on given information").

- **Which program has the better guarantee on the running time for large values of N (N > 10,000)?**
- **Which program has the better guarantee on the running time for small values of N (N < 100)?**
- **Which program will run faster on average for N = 1,000?**
- **Can program B run faster than program A on all possible inputs?**

Solution:

A - 150N log N
B - N^2

- **Given,**
  N > 10000

  For N = 10000: A - 6*(10^6)*log10
  B - 10^8
  For N = 20000: A - 12*10^6*log10
  B - 4*10^8

  Conclusion: A will perform better

- **Given,**
  N < 100

  For N = 100: A - 30000log10
  B – 10000

  Conclusion: B will perform better

- **Given,**
  N = 1000

  Considering worst case scenario,
  A - 450000log10
  B - 10^6

  Conclusion: A will run faster

- No. As the no of inputs increase, A will perform better compared to B.

2. Solving a problem requires running an O(N) algorithm, and then performing N binary searches on an N-element array, and then running another O(N) algorithm. What is the total time complexity to solve the problem?

Solution:
Time Complexity of Binary Search = O (n log n)

Therefore, Total Time Complexity = O (n) + O (n log n) + O (n)

We can approximate this to complexity of O (n log n).

3. An algorithm takes 0.5 milliseconds for input size 100. How large a problem can be solved in 1 minute (assuming that low-order terms are negligible) if the running time is

1. **O(N)**
2. **O(N log N)**
3. **$O(N^2)$**
4. **$O(N^3)$**

Solution:
Given, 0.5ms - 100 inputs

We know, 1 minute = 60000ms

1. **O (N) :**
   Given,
   0.5ms - 100 inputs
   Therefore,
   1 min – (100/0.5ms) * 1 * 60000ms = 12000000 inputs

   So, in 1 minute no of inputs would be nearly equal to 12,000,000

2. **O (N log N) :**
   Given,
   0.5ms - 100 log100
   1 min - 100 log100 * 60000 / 0.5

   Therefore,
   N log N = 24000000
   N = 3656807

   So, in 1 minute no of inputs would be in order of 10^6

3. **O(N^2) :**
   Given,
   0.5ms - 100 * 100
   1 min - 100 * 100 * 60000 / 0.5 = 1200000000

Therefore,
N^2 = 1200000000
N = 34641.01

So, in 1 minute no of inputs would be in order of 10^4

4. **O(N^3) :**
   Given,
   0.5ms - 100*100*100
   1 min - 100 * 100 * 100 * 60k / 0.5 = 120000000000

   Therefore,
   N^3 = 120000000000
   N = 4932

   So, in 1 minute no of inputs would be in order of 10^3

4. Give an efficient algorithm to determine whether an integer i exists such that A[i] = i in an array of increasing integers. What is the running time of your algorithm in big-O notation as a function of n the length of A?

Your answer can be conceptual, with pseudo-code for clarification.

Solution:

ASSUMPTION:
- Array is sorted in ascending order
- All the integers in the array are distinct

LOGIC:
Since the array is sorted and distinct,

For each j < i: array[j] - j <= 0
For each j > i: array[j] - j >= 0

(Because at each step j vary of 1 but array[j] vary of at least 1).

PSEUDO-CODE:

```
A= [-5, -3, -2, 0, 1, 3, 5, 7]
min = 0
max = len(A) - 1
flag = 0

while min <= max and flag == 0:
   mid = (min + max) / 2

   temp = A[mid] - mid
```

```
if temp < 0:
   min = mid + 1

elif temp > 0:
   max = mid – 1

else:
   print "Match found: i is %d and A[i] is %d” % mid A[mid]
   flag = 1
```

CONCLUSION:

Time Complexity: O (log N)

Python Applications

From Python for Informatics Text
S12.3 Use urllib to replicate Exercices 1 and 2 in 12.10, i.e., to (1) retrieve the document from a URL, (2) display up to 3000 characters, and (3) count the overall number of characters in the document. (Don't worry about the headers for this exercise, simply show the first 3000 characters of the document contents.)

Solution:

```
import urllib

while True :

 try:

   url=raw_input("Enter url: ")

   page = urllib.urlopen(url)
   print "Web Address is valid"
   break

 except:

    print ("Invalid urlname - Enter a valid one")
    continue

fhand = page.read()
print fhand[:3000]

count=0

for i in fhand:
  count +=1

print "The number of characters in document are %d" % count
```

S13.11 Change www.py4inf.com/code/geojson.py to print out the two-character country code from the retrieved data. Add error checking so your program does not traceback if the country code is not there. Once you have it working, search for "Atlantic Ocean" and make sure it can handle locations that are not in any country.

Solution:

```
import urllib
import json

serviceurl = 'http://maps.googleapis.com/maps/api/geocode/json?'

while True:
    address = raw_input('Enter location: ')
    if len(address) < 1 : break

    url = serviceurl + urllib.urlencode({'sensor':'false', 'address': address})
    print 'Retrieving', url
    uh = urllib.urlopen(url)
    data = uh.read()
    print 'Retrieved',len(data),'characters'

    try: js = json.loads(str(data))
    except: js = None
    if 'status' not in js or js['status'] != 'OK':
        print '==== Failure To Retrieve ===='
        print data
        continue

    print json.dumps(js, indent=4)

    lat = js["results"][0]["geometry"]["location"]["lat"]
    lng = js["results"][0]["geometry"]["location"]["lng"]
    print 'lat',lat,'lng',lng
    location = js['results'][0]['formatted_address']
    print location
    addressCode = js['results'][0]['address_components']
    temp=""
    for address in addressCode:
       if "country" in address["types"]:
            temp = address["short_name"].encode('ascii','ignore')
    if temp == "":
            print "Country code: None"
    else:
            print temp
```