CS 61B Spring 2018

Asymptotic Analysis

Discussion 7: February 27, 2018

1 Asymptotic Notation

1.1 Order the following big-O runtimes from smallest to largest.

```
O(\log n), O(1), O(n^n), O(n^3), O(n \log n), O(n), O(n!), O(2^n), O(n^2 \log n)
```

```
O(1) < O(\log n) \bigcirc O(n) < O(n \log n) < O(n^2 \log n) < O(n^3) < O(2^n)
```

Are the statements in the right column true or false? If false, correct the asymptotic notation $(\Omega(\cdot), \Theta(\cdot), O(\cdot))$. Be sure to give the tightest bound. $\Omega(\cdot)$ is the opposite of $O(\cdot)$, i.e. $f(n) \in \Omega(g(n)) \iff g(n) \in O(f(n))$.

```
f(n) = 20501
                                                                f(n) \in O(g(n))
                                g(n) = 1
f(n) = n^2 + n
                                g(n) = 0.000001n^3
                                                                f(n) \in \Omega(g(n)) 
f(n) = 2^{2n} + 1000
                                g(n) = 4^n + n^{100}
                                                                f(n) \in O(g(n))
f(n) = \log(n^{100})
                                g(n) = n \log n
                                                                f(n) \in \Theta(g(n))
                                g(n) = n^2 + n + \log n
f(n) = n\log n + 3^n + n
                                                                f(n) \in \Omega(g(n)) \checkmark
                                g(n) = \log n + n^2
f(n) = n \log n + n^2
                                                                f(n) \in \Theta(g(n)) \checkmark
                                                                f(n) \in O(g(n)) \times \int
f(n) = n \log n
                                g(n) = (\log n)^2
```

2 Analyzing Runtime

Give the worst case and best case runtime in terms of M and N. Assume ping is in $\Theta(1)$ and returns an **int**.

```
int j = 0;
for (int i = N; i > 0; i--) {
for (; j <= M; j++) {

if (ping(i, j) > 64) {

break;

break;

for (int i = N; i > 0; i--) {

if (ping(i, j) > 64) {

break;

break;

break;

continuous (int)

break;

continuous (int)

continuous (int)
```

}

2.2 Give the worst case and best case runtime where N= array.length. Assume mrpoolsort(array) is in $\Theta(N\log N)$.

```
public static boolean mystery(int[] array) {
           array = mrpoolsort(array);
    2
           int N = array.length;
    3
           for (int i = 0; i < N; i += 1) {
               boolean x = false;
               for (int j = 0; j < N; j += 1) {
                   if (i != j && array[i] == array[j])
                       x = true;
               }
               if (!x) {
   10
                   return false; (YN) OV YOTUM
   11
   12
           }
   13
           return true:
   14
   15
       Achilles Added Additional Amazing Asymptotic And Algorithmic Analysis Achievements
                                            any unique int
                                                                              the
                                                                       1 N
        (a) What is mystery() doing?
              find if there is duplicate items in array
        (b) Using an ADT, describe how to implement mystery() with a better runtime.
           Then, if we make the assumption an int can appear in the array at most twice,
           develop a solution using only constant memory.
                                                     for cint 1=0, ic N-1.7+=1)
A hash set
                                                        for ( int j= i+1, in, j+=1)
       Give the worst case and best case running time in \Theta(\cdot) notation in terms of M and
  2.3
       N. Assume that comeOn() is in \Theta(1) and returns a boolean.
                                                                                       true break
       for (int i = 0; i < N; i += 1) {
           for (int j = 1; j \le M; ) {
    2
               if (comeOn()) {
    3
                   j += 1;
               } else {
                   j *= 2;
           }
```

3 Have You Ever Went Fast?

Given an **int** x and a *sorted* array A of N distinct integers, design an algorithm to find if there exists indices i and j such that A[i] + A[j] == x.

Let's start with the naive solution.

(a) How can we improve this solution? Hint: Does order matter here?

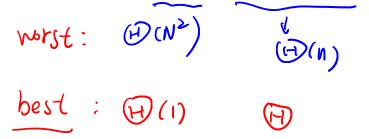
$$i=0$$
, $j=A$. legath -1

while $(i < j)$
 $i+A$ [$i+A$ [$i+A$ [$i+A$] = -1 .

return true

 $i+1$
 $i+1$

(b) What is the runtime of both the original and improved algorithm?



4 Asymptotic Analysis

4 CTCI Extra

4.1 Union Write the code that returns an array that is the union between two given arrays. The union of two arrays is a list that includes everything that is in both arrays, with no duplicates. Assume the given arrays do not contain duplicates. For example, the union of {1, 2, 3, 4} and {3, 4, 5, 6} is {1, 2, 3, 4, 5, 6}.

 Hint : The method should run in O(M+N) time where M and N is the size of each array.

hash-set

no overlup overlap

code?

4.2 **Intersect** Now do the same as above, but find the intersection between both arrays. The intersection of two arrays is the list of all elements that are in both arrays. Again assume that neither array has duplicates. For example, the intersection of {1, 2, 3, 4} and {3, 4, 5, 6} is {3, 4}.

Hint: Think about using ADTs other than arrays to make the code more efficient.

hash-set

(-) (M+N)

code?