CS 61B Spring 2019 — Asymptotics II & Search Trees

Exam Prep 7: March 4, 2019

1 More, MORE, MOREEEE (Spring 2016, MT2)

For all the methods below, give the runtime in $\Theta(\cdot)$ notation as a function of N. Your answer should be simple, with no unnecessary leading constants or summations.

```java
public static void p1(int N) {
    for (int i = 0; i < N; i += 1) {
        for (int j = 1; j < N; j = j + 2) {
            System.out.println("hi !");
        }
    }
}
```

```
N

N / 2

N (N / 2) = N^2 / 2 ~ N^2

(N^2)
```

P1 answer: $\Theta(\quad)$

```java
public static void p2(int N) {
    for (int i = 0; i < N; i += 1) {
        for (int j = 1; j < N; j = j * 2) {
            System.out.println("hi !");
        }
    }
}
```

```
N

log N

(NlogN)
```

P2 answer: $\Theta(\quad)$

```java
public static void p3(int N) {
    if (N <= 1) return;
    p3(N / 2);
    p3(N / 2);
}
```

P3 answer: $\Theta(\quad)$

```
           N
        /     \
     N/2       N/2
    /  \      /   \
N/4 N/4   N/4   N/4

Height: logN
#node/layer: 2^i
(0~logN)
Sum: 1 + 2^1 + 2^2 + … +
2^logN
   = 2^logN x 2 - 1 =
2N-1 ~ N
```

```java
public static void p4(int N) {
    int m = (int)((15 + Math.round(3.2 / 2)) *
        (Math.floor(10 / 5.5) / 2.5) * Math.pow(2, 5));
    for (int i = 0; i < m; i++) {
        System.out.println("hi");
    }
}
```

m = (int) (17 x 0.4 x 32) = (int) 217.6 = 217

m = 217

(1)

P4 answer: $\Theta($ $)$

```java
public static void p5(int N) {
    for (int i = 1; i <= N * N; i *= 2) {
        for (int j = 0; j < i; j++) {
            System.out.println("moo");
        }
    }
}
```

1~N^2 / 2

1 + 2 + 4 + 8 + … +
N^2 / 2 + N^2
= N^2 x 2 - 1 ~ N^2

(N^2)

P5 answer: $\Theta($ $)$

2 A Wild Hilfinger Appears! (Fall 2017, Final)

a. Given the following function definitions, what is the worst-case runtime for `p(N)`? Assume `h` is a boolean function requiring constant time.

Answer: $\Theta($ $)$

```java
int p(int M) {
    return r(0, M);
}

int r(int i, int M) {
    if (i >= M) return 0;
    if (s(i) > 0) return i;
    return r(i + 1, M);
}

int s(int k) {
    if (k <= 0) return 0;
    if (h(k)) return k;
    return s(k - 1);
}
```

#call to r(int,int)

s(i) > 0 => if (i <= 0) return 0;
if (h(i)) return k;
else return s(i - 1)

recursive for M times

#call to s(int)

k times of calls to s
that function means:
if run k times: return 0
if h(k) holds true: return k (> 0) => trigger return i in r(int,int)

Consider worst case: h(k) never holds true, so in every r(int,int), s(int) will be called for i times.

Cost model: #call to s(int); Runtime: 1 + 2 + 3 + 4 + … + M ~ M^2 = (N^2)

b. What is the worse-case runtime for the call `p(N)`? Assume that calls to `h` require constant time.

Answer: $\Theta($ $\quad$ $)$

```
void p(int M) {            L is like count
    int L, U;
    for (L = U = 0; U < M; L += 1, U += 2) {    U: 0 ~ M / 2
        for (int i = L; i < U; i+= 1) {
            h(i);          i: L ~ U (U-L)
        }
    }
}
```

```
L:   0  1  2  3  4  5  6  7  8  9
U:   0  2  4  6  8  10 12 14 16 18
U-L: 0  1  2  3  4  5  6  7  8  9
sum ~ (M/2)(M/2-1)/2 = M^2/8-M/4
~ M^2 = (N^2)
```

3 Tree Time (Spring 2018, Midterm 2)

a. Draw the 2-3 tree that results from inserting 1, 2, 3, 7, 8, 9, 5 in that order.

```
     2   7
   /   |  \
  1   3 5  8 9
```

b. Draw a valid BST of minimum height containing the keys 1, 2, 3, 7, 8, 9, 5.

1, 2, 3, 5, 7, 8, 9

```
      5
    /   \
   2     8
  / \   / \
 1   3 7   9
```