CS 113 – Basic Data Structures and Algorithms

Homework Exercises (50 HA points)

Homework #2

3. Big-O notation Big Oh O

Usefulness of Big-O = It represents the upper bound. Big-oh is the most useful because represents the worst-case behavior. So, it guarantees that the program will terminate within a certain time period, it may stop earlier, but never later. //O(N) means it takes at most N steps.

a. Rank the following in order of increasing run times, if they are same list them together.

i. O(N)
ii. O(∞)
iii. O(NM)
iv. $O(\sqrt{N})$
v. O(5)
vi. $O(N^2)$
vii. O(log N)
viii. O(N log N)
ix. O(0)
x. $O(N^4)$
xi. O(2/N)
xii. $O(2^N)$
xiii. $O(N^{1.5})$

i. O(2/n) (Theory: not a real example)
ii. O(5),O(n) (linear growth), O(0) (Constant factors do not matter for Big O so these are the same as O(1) in other words, If you repeat an action independent from the size of the input n times, you'll get 5n or 10n, which are both O(n))
iii. O(√n)
iv. O(log n) (logarithmic growth)
v. O(n log n) (has a faster growth rate than n, but slower than quadratic)
vi. $O(n^{1.5})$ (is faster than quadratic growth/ so are these the same?)
vii. $O(n^2)$ (Quadratic growth)
viii. $O(n^4)$ (slower than n^2)
ix. $O(2^n)$ (exponential growth), O(nm)
x. O(∞) (this is the slowest, for any constant of infinity it will never complete).

b. What is the complexity of the following pieces code?

i.
```
sum = 0;
for (i = 0; i < n; i++) {
    sum++;
}
Complexity: O(n)
```

ii.

```
sum = 0;
for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++) {
```

iii.

```
sum = 0;
for (i = 0; i < n; i++) {
    for (j = 0; j < i; j++) {
        sum++;
    }
}
```

Complexity: $O(n^2)$

iv.

```
sum = 0;
for (i = 0; i < n * n; i++) {
    for (j = 0; j < n * n; j++) {
        sum++;
    }
}
```

Complexity: $O(n^2)$