3.1

1. {1, 8, 12, 9, 11, 2, 14, 5, 10, 4}

Step 1: Set temp to first element. Temp = 1

Step 2: Compare element i+1 to temp. If greater, set element to temp.

Step 3: Repeat.

43

Once the next element is greater or equals, place the number before it. If reaches end, place at e

2.

5.

```
procedure unique(n, list)
    int i = 0
    while (i < n) {
        int j = i+1.
        print(list[i])
        while (j<N and list[j] == list[i]) {
            j++
        }
        i=j
    }
```

14. Linear search: Goes through everything and breaks after 8

Binary search: 5, 9, 8, 6, break

23. procedure cnt(a,b)

```
    for (item : b) {
        IF (a.not contains(item)) return fals
```

3.2

5. $(x^2+1)/(x+1) = O(x)$

for $k>1$ and $C=1$

At $x=10$ $\frac{(100+1)}{(11)} < (10)$

7.

a. $n=3$ b. $n=3$

17.

Since $f(x) \leq C_1(g(x))$

and $g(x) \leq C_2(h(x))$

$f(x) \leq C_1(g(x)) \leq C_2(h(x))$

therefore $f(x) \leq C_2(h(x)) = O(h(x))$

22

$(n!)^2$, 10^n, 1.5^n, n^{100}, $n^{an}+n^{98}$, $(\log n)\cdot\sqrt{n}$, $(\log n)^3$

25.

a. $O(n^3)$ b. $O(n^4 \log n)$ c. $O((n+3)!)$

38. $O\left(\frac{(2n)!}{2^n}\right)$

3.3. 1. $O(1)$

2. $O(n^2)$. 3. $O(n^2)$ 4. $O(\log_2 n)$

7. A binary search is actually slower

If the linear search is guaranteed in

4 first beginning, then bsearch takes $\log_2 32 = 5$,

whereas linear only takes 4.

8.13.

a. step 1: power = power · 2 = 2 step 2: power = 2 · 2 = 4

y = y + 1 · 2 = 1 + 2 y = y + 3 · 4 = 3 + 12 = 15

return

return 15.

b. There are 4 multiplications and 1 additions

15.1

a. 10^{10^9} b. 10^7 c. $\sim 10^8$

e. 10^{45} e. ~ 31 f. 2.12!

16.

a. 1224 ns b. 1.0493 ms c. 13.03 days

e. 4 ell centuries.