Summation Review

$$\sum_{i=l}^{u} c a_i = c \sum_{i=l}^{u} a_i$$

$$\sum_{i=l}^{u} c a_i = c a_l + c a_{l+1} + \dots + c a_u$$

$$= c(a_l + a_{l+1} + \dots + a_u)$$

$$= c \sum_{i=l}^{u} a_i$$

$$\sum_{i=l}^{u} (a_i + b_i) = \sum_{i=l}^{u} a_i + \sum_{i=l}^{u} b_i$$

$$= a_l + b_l + a_{l+1} + b_{l+1} + \dots + a_u + b_u$$

$$= [a_l + a_{l+1} + \dots + a_u] + [b_l + b_{l+1} + \dots + b_u]$$

$$\sum_{i=\ell}^{u} 1 = u - \ell + 1$$

$$\sum_{i=0}^{n} i = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

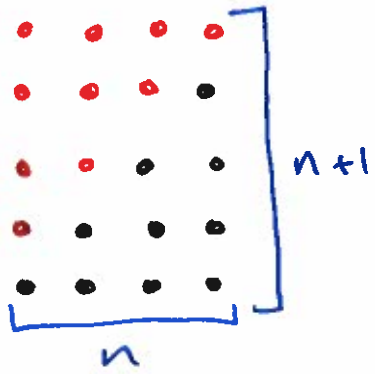
$$S = 1 + 2 + 3 + \dots + n-1 + n$$

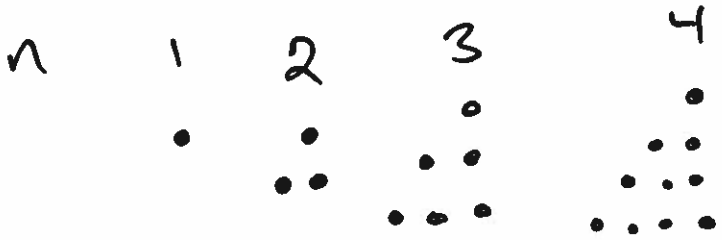
$$S = n + n-1 + n-2 + \dots + 2 + 1$$

$$2S = (n+1) + (n+1) + (n+1) + \dots + (n+1) + (n+1)$$

$$2S = n(n+1)$$

$$S = \frac{n(n+1)}{2}$$





$$- \sum_{i=1}^{n} (a_i - a_{i-1}) = a_n - a_0$$

$$= a_n - a_{n-1} + a_{n-1} - a_{n-2} + a_{n-2} - a_{n-3} + \dots + a_1 - a_0$$

$$= a_n - a_0$$

$$- \sum_{i=0}^{n} 2^i = 2^{n+1} - 1$$

$$\sum_{i=0}^{n} a^i = \frac{a^{n+1} - 1}{a - 1} \qquad a \neq 1$$

Base case: $n = 0$ $\qquad a^0 = 1$

$$\frac{a^{0+1} - 1}{a - 1} = 1$$

Induction

$$\sum_{i=0}^{k} a^i = \frac{a^{k+1}-1}{a-1} \Rightarrow \sum_{i=0}^{k+1} a^i = \frac{a^{k+2}-1}{a-1}$$

$$\sum_{i=0}^{k+1} a^i = \sum_{i=0}^{k} a^i + a^{k+1}$$

$$= \frac{a^{k+1}-1}{a-1} + a^{k+1}$$

$$= \frac{a^{k+1}-1}{a-1} + \frac{a^{k+1}(a-1)}{a-1}$$

$$= \frac{a^{k+1}-1+a^{k+2}-a^{k+1}}{a-1}$$

$$= \frac{a^{k+2}-1}{a-1}$$

Analyzing Iterative Algorithms

Basic Procedure

1. Decide on a parameter or parameters for measuring the size of the input.
2. Identify the basic operation
3. Determine if Best, Worst, and Average case performance will be different
4. Set up a summation for the number of times the basic operation is performed.
5. Find a closed form for the summation.

BUBBLESORT(A)

```
    // Input: An array A of n comparable elements
    // Output: A sorted in non-decreasing order
1   n = A.length
2   for i = 0 to n − 2
3       for j = 0 to n − 2 − i
4           if A[j + 1] < A[j]
5               exchange A[j] with A[j + 1]
6   return A
```

1. Input size? The length of the array

2. Basic Op? comparison between A[j] and A[j+1]

3. Does the number of Basic operations vary with more than just input size? No

4. $C(n) = \sum_{i=0}^{n-2} \sum_{j=0}^{n-2-i} 1$

5. $$\sum_{i=0}^{n-2} \sum_{j=0}^{n-2-i} 1 = \sum_{i=0}^{n-2} n-2-i+1$$

$$= \sum_{i=0}^{n-2} n-1-i = (n-1)+(n-2)+(n-3)+\ldots+1$$

$$= \sum_{k=1}^{n-1} k$$

$$= \frac{n(n-1)}{2} \in \Theta(n^2)$$

MATRIXSUMMATION(A)

```
    // Input: A matrix A of integers
    // Output: the sum of all elements in A
1   h = A.height
2   w = A.width
3   total = 0
4   for i = 0 to h − 1
5         for j = 0 to w − 1
6               total = total + A[i][j]
7   return total
```

1. Input size? total number of elements $n = hw$

2. Basic op? addition

3. Are Best/Worst/Average different? No

4. $C(n) = \sum_{i=0}^{h-1} \sum_{j=0}^{w-1} 1$

5. $\sum_{i=0}^{h-1} \sum_{j=0}^{w-1} 1 = \sum_{i=0}^{h-1} w$

$= w \sum_{i=0}^{h-1} 1$

$= w h = n \quad \in \Theta(n)$

```
ITERATIVEBINARYSEARCH(A, T)
    // Input: An array A of n elements in sorted order
    // and a target value T
    // Output: the index of T in A or -1 if not found
1   from = 0
2   to = A.length − 1
3   while from ≤ to
4       mid = (from + to)/2
5       if T < A[mid]
6           to = mid − 1
7       elseif T > A[mid]
8           from = mid + 1
9       else
10          return mid
11  return −1
```

1. Input size? the length of A

2. Basic op? comparison between T and A[mid]

3. Does the number of basic operations vary with more than input size?

   yes

Best Case:

T = A[mid] during the first iteration

O(1)

Worst Case:

T is not an element of A

4. How many iterations can the loop take?

- each step cuts the array in half

$$\frac{n}{2^i} = 1 \quad \text{solve for } i$$

$$2^i = n$$

$$i = \lg(n)$$

$$C(n) = \sum_{i=1}^{\lg n} 2$$

5. $$\sum_{i=1}^{\lg n} 2 = 2 \sum_{i=1}^{\lg n} 1$$

$$= 2 \lg n \in \Theta(\lg n)$$

```
UNKNOWN(A)
    // Input: An array A of n integers
    // Output: ????
1   m = 0
2   for i = 0 to n − 1
3       for j = i + 1 to n − 1
4           if |A[i] − A[j]| > m
5               m = |A[i] − A[j]|
6   return m
```

(a) What does algorithm compute?

(b) What is the basic operation?

(c) How many times is the basic operation performed on a list of n integers?

(d) What is the asymptotic complexity of this algorithm?