## Mergesort

```
MERGE-SORT ( A )
    if len(A) ≤ 1
        return copy of A
    else if len(A)=2
        if A[0] ≤ A[1]
            return copy of A
        else
            return reverse copy of A
    else
        L ← 1st half of A          ] divide
        R ← rest of A              ]
        L ← MERGE-SORT (L)         ] conquer
        R ← MERGE-SORT (R)         ]
        return MERGE(L, R)         ] combine
```

```
MERGE (L, R)

    A ← empty list
    i ← 0
    j ← 0
    while i < len(L) and j < len(R)
        if L[i] ≤ R[j]
            append L[i] to A
            i ← i+1
        else
            append R[j] to A
            j ← j+1

    append L[i ... len(L)-1] to A
    append R[j ... len(R)-1] to A

    return A
```

Divide-and-Conquer

divide
conquer
combine

1 3 2 8 | 4 9 5 6

~~1~~ ~~2~~ ~~3~~ 8 ~~4~~ 5 6 9

1 2 3 4 . . . .

assume $= 2^k$

$T(n)$ = worst-case time on list of size $n$

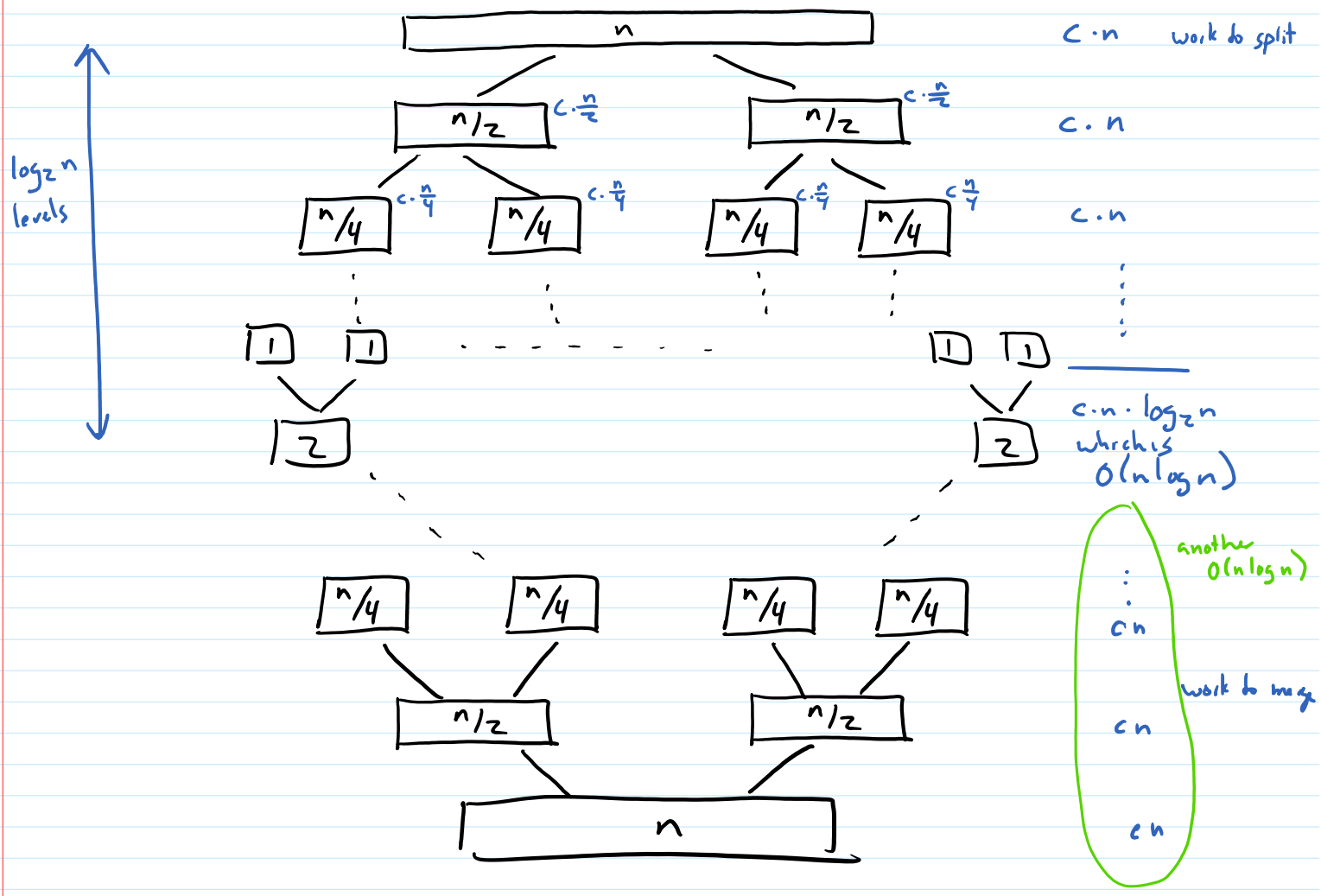
$= c_1$ for $n = 0, 1, 2$

$\leq c_2 \cdot n + 2 \cdot T(\frac{n}{2}) + c_3 \cdot n$

$c = \max(c_1, c_2 + c_3)$

$T(2) \leq c$

$n > 2 \quad T(n) \leq 2 \cdot T(\frac{n}{2}) \cdot cn$




$T(n) \leq cn \log_2 n$

$T(n) \leq 2 \cdot T(\frac{n}{2}) + cn$

$\leq 2 \cdot c\frac{n}{2} \cdot \log_2 \frac{n}{2} + cn$

$= c \cdot n \cdot \log_2 \frac{n}{2} + cn$

$= cn(\log_2 n - \log_2 2) + cn$

$= cn(\log_2 n - 1) + cn$

$= cn \log_2 n - \cancel{cn + cn}$

## Binary Search on Linked List (and why it is pointless)

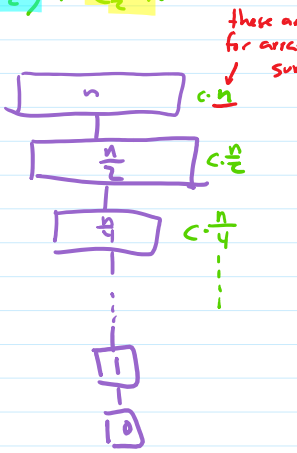
LINKED-BINARY-SEARCH (A, key)

```
if A is empty
    return FALSE
else
    mid = size(A) / 2          ← linear time for random access
    if A[mid] = key
        return TRUE
    else if key < A[mid]
        return LINKED-BINARY-SEARCH( A_{0...mid-1} )
    else
        return LINKED-BINARY-SEARCH( A_{mid+1...len(A)-1} )
```

linear time to copy half of list

$T(0) = c_1$

$T(n) \leq T(\frac{n}{2}) + c_2 \cdot n$




$c \cdot \log_2 n$ so $O(\log n)$

$$\text{total} = \sum_{i=0}^{\log_2 n} c \cdot n \cdot (\tfrac{1}{2})^i$$

$$= cn \cdot \sum_{i=0}^{\log_2 n} (\tfrac{1}{2})^i$$

$$\leq cn \cdot \sum_{i=0}^{\infty} (\tfrac{1}{2})^i$$

$$= cn \cdot \frac{1}{1-\frac{1}{2}}$$

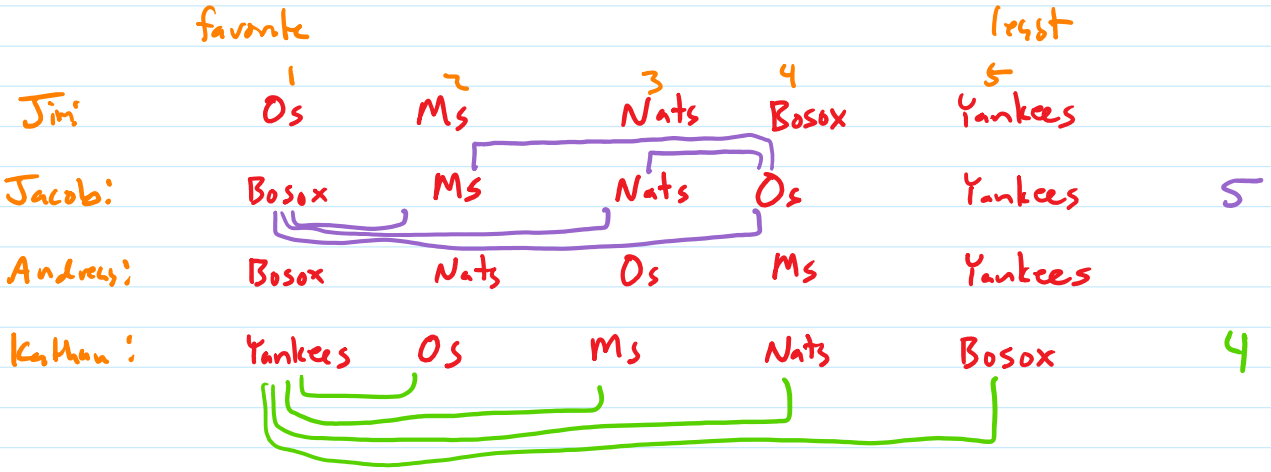
$$= 2cn$$

which is $O(n)$

## Counting Inversions

Rank the following

Orioles  Mariners  Nationals  Red Sox  Yankees




Brute force: for each pair, check if inverted

$O(n^2)$ pairs  O(1) per pair  $O(n^2)$ total

4 2 3 | 1 5 2 3 4 ✗ 5

1 ...  ↑ inverted wrt rest of L

MERGE-AND-COUNT(L, R)

```
count ← 0
A ← empty list
i ← 0
j ← 0
while i < len(L) and j < len(R)
    if L[i] ≤ R[j]
        append L[i] to A
        i ← i+1
    else count ← len(L) − i
        append R[j] to A
        j ← j+1
append L[i ... len(L)−1] to A
append R[j ... len(R)−1] to A
return A
```

COUNT-AND-SORT(A)

```
base cases
L ← 1st half of A
```

$T(n) \leq 2 \cdot T(\frac{n}{2}) + cn$

COUNT-AND-SORT (A)

base cases

L ← 1st half of A
R ← 2nd half of A

L, countL ← COUNT-AND-SORT (L)
R, countR ← COUNT-AND-SORT (R)
M, countM ← MERGE-AND-COUNT (L, R)

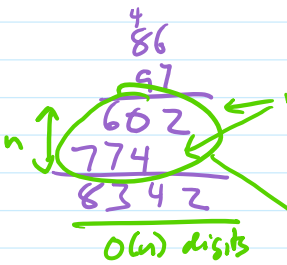
return M, countL + countR + countM

$T(n) \leq 2 \cdot T(\frac{n}{2}) + cn$

$T(n) = O(n \log n)$

## Integer Multiplication

862341 · 979468




$z = x \cdot y$ O(1) for C, Java ints but might overflow

work to multiply 2 n-digit numbers $O(n^2)$

can we do better?

$x = x_1 \cdot 1000 + x_0$

$y = y_1 \cdot 1000 + y_0$

$xy = x_1 \cdot y_1 \cdot 10^6 + x_1 \cdot y_0 \cdot 10^3 + y_1 \cdot x_0 \cdot 1000 + x_0 \cdot y_0$

$T(n) \le 4 \cdot T(\frac{n}{2}) + cn$ ; $T(n)$ is $O(n^2)$

$= \underbrace{x_1 \cdot y_1}_{1\text{ mult}} \cdot \underbrace{10^6}_{\text{shift}} + \underbrace{x_1 \cdot y_1 \cdot 10^3}_{\text{reuse}} - 10^3 \underbrace{(x_1 - x_0)(y_1 - y_0)}_{1\text{ mult}} + 10^3 \underbrace{x_0 \cdot y_0}_{1\text{ mult}} + \underbrace{x_0 \cdot y_0}_{\text{reuse}}$

$T(n) \le 3 \cdot T(\frac{n}{2}) + cn$

$T(n)$ is ???

3 multiplications $3 \cdot T(\frac{n}{2})$
6 additions/subtractions $O(n)$
4 shifts $O(n)$

```
int m = s1.length() / 2;

String x1 = s1.substring(0, m);
String x0 = s1.substring(m);
String y1 = s2.substring(0, m);
String y0 = s2.substring(m);

BigInteger tenm = (new BigInteger("10")).pow(s1.length() - m);
BigInteger ten2m = tenm.multiply(tenm);
BigInteger z0 = multiply(x0, y0);
BigInteger z2 = multiply(x1, y1);
BigInteger z1 = multiply((new BigInteger(x1)).subtract(new BigInteger(x0)).toString(),
                         (new BigInteger(y1)).subtract(new BigInteger(y0)).toString());

return z2.multiply(ten2m).add(z2.multiply(tenm))
                         .subtract(z1.multiply(tenm))
                         .add(z0.multiply(tenm))
                         .add(z0);
```