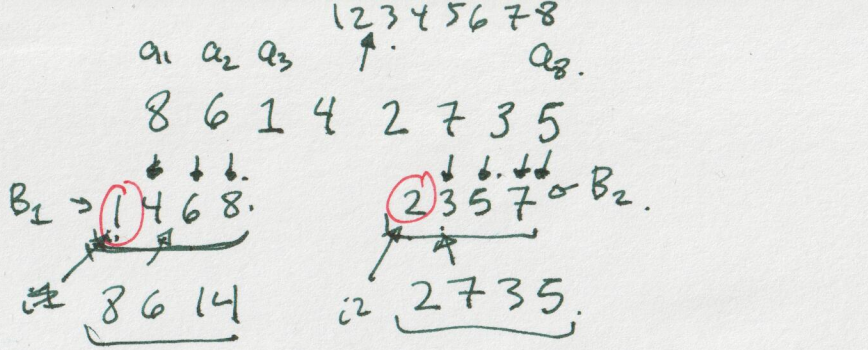
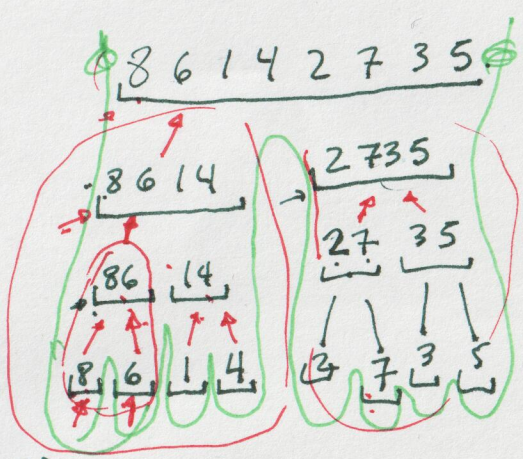
MergeSort: Divide-and-conquer!

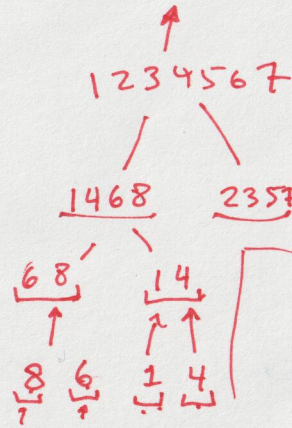
```
MergeSort(a_1, ..., a_n)        // unsorted array
  if n ≤ 1 return a_1 ... a_n.
  B_1 = MergeSort(a_1, ..., a_⌈n/2⌉)
  B_2 = MergeSort(a_⌈n/2⌉+1, ..., a_n)
  return Merge(B_1, B_2)
```



= 1 4 6 8 (4) 2 3 5 7 (4) ⇒ L (8)

L = 1 . 2 3 4 5 6 . 7 8 .






```
Merge(B_1, B_2)      // B_1 = B_1[1], B_1[2], ... B_1[len(B_1)].
                     // B_2 = B_2[1], B_2[2], ..., B_2[len(B_2)].
  L = new list.
  i1 = 1
  i2 = 1
  while len(L) < len(B_1) + len(B_2).
    if i1 > len(B_1).
      L.append(B_2[i2]).
      i2 = i2 + 1
    elif i2 > len(B_2)
      L.append(B_1[i1]).
      i1 = i1 + 1.
    if B_1[i1] < B_2[i2].
      L.append(B_1[i1]).
      i1 = i1 + 1.
    else
      L.append(B_2[i2]).
      i2 = i2 + 1.
```

1 2 3 4 5 6 7 8.

8 7 6 5 4 3 2 1.

5 6 7 8 1 2 3 4

Proof of Correctness:

By induction on n.

Base case $n \leq 1$, then obviously correct.

I.H. MergeSort correctly sorts any input array of length $K \in \{0, 1, \ldots n-1\}$. Prove that MergeSort correctly sorts any input of length $n \geq 2$.

$\lfloor \frac{n}{2} \rfloor \leq n-1$ for any $n \geq 2$,

therefore MergeSort($a_1 \ldots a_{\lfloor \frac{n}{2} \rfloor}$) ~~return~~ correctly sorts $a_1 \ldots a_{\lfloor \frac{n}{2} \rfloor}$ by the inductive hypothesis.

$n - \lfloor \frac{n}{2} \rfloor \leq n-1$ for any $n \geq 2$.

therefore MergeSort($a_{\lfloor \frac{n}{2} \rfloor + 1} \ldots a_n$) correctly sorts $a_{\lfloor \frac{n}{2} \rfloor + 1} \ldots a_n$.

- If Merge(B_1, B_2) correctly merges B_1 and B_2 then MergeSort($a_1 \ldots a_n$) correctly sorts $a_1, \ldots a_n$.

- Running Time Analysis.
- Number of comparisons done.
- Number of comparisons done by Merge(B_1, B_2) is variable.

→ but never more than $\text{len}(B_1) + \text{len}(B_2)$ [-1].

- Let $f(n)$ be the maximum number of comparisons done by MergeSort for an input of length n.

$$f(n) \leq \begin{cases} 0. & \text{if } n \leq 1. \\ f(\lfloor \frac{n}{2} \rfloor) + f(n - \lfloor \frac{n}{2} \rfloor) + n & \text{if } n \geq 2. \end{cases}$$

n	k	$f(n)$.
1.	0	0.
2	1	$f(1)+f(1)+2 = 2$.
4	2	$f(2)+f(2)+4 = 2+2+4 = 8$.
8	3	$f(4)+f(4)+8 = 8+8+8 = 24$.

Theorem: If $n = 2^k$ for some non-negative integer k then

- $f(n) = n \log_2 n$
- $= n \cdot k$.
- $= 2^k \cdot k$ ∴

Proof: By induction on k.

Base case $k=0$, $n = 2^0 = 1$. ✓

Now assume ~~$f(\ell)$~~

→ $f(2^\ell) = 2^\ell \cdot \ell$ for any $\ell \in \{0, \ldots, k-1\}$.

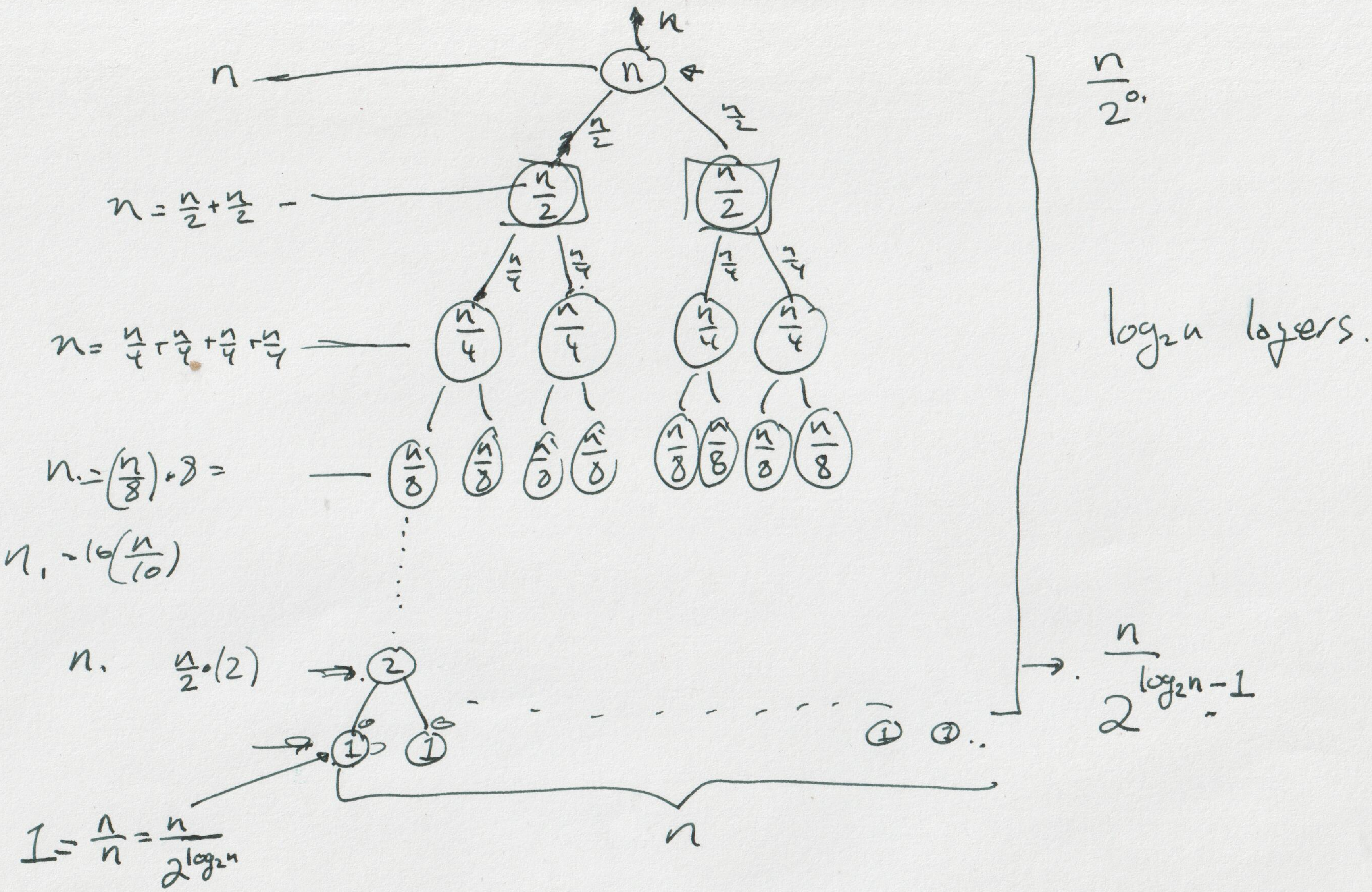
and prove $f(2^k) = 2^k \cdot k$.

$$f(2^k) = f(2^{k-1}) + f(2^{k-1}) + 2^k$$
$$= 2 \cdot f(2^{k-1}) + 2^k$$
$$= 2\left(2^{k-1}(k-1)\right) + 2^k = 2^k(k-1) + 2^k$$
$$= 2^k \cdot k - 2^k + 2^k$$
$$= 2^k \cdot k. \quad \text{QED.}$$

$f(n) = f(\frac{n}{2}) + f(\frac{n}{2}) + n.$

$\ell = k-1$

$f(2^{k-1})$ $2^{k-1}(k-1)$.


n
n
n = n/2 + n/2
n = n/4 + n/4 + n/4 + n/4
n = (n/8) · 8 =
n, = 10(n/10)
n. n/2 · (2)
1 = n/n = n/2^{log₂n}
n
n/2⁰
log₂n layers.
n/2^{log₂n − 1}