1 Merge Sort

1.1 Pseudocode

Algorithm 1 Merge Sort

```
procedure MERGESORT(A, p, r)
    if p < r then
        q ← ⌊(p + r)/2⌋
        MERGESORT(A, p, q)
        MERGESORT(A, q + 1, r)
        MERGE(A, (p, q), (q + 1, r))
    end if
end procedure

procedure MERGE(A, (p, q), (q + 1, r))
    copy (A, p, r) into (B, p, r)
    i ← p
    j ← q + 1
    k ← p
    while i ≤ q and j ≤ r do
        if B[i] ≤ B[j] then
            A[k] ← B[i]
            i ← i + 1
        else
            A[k] ← B[j]
            j ← j + 1
        end if
        k ← k + 1
    end while
    if i > q then
        copy (B, j, r) into (A, k, r)
    else
        copy (B, i, q) into (A, k, r)
    end if
end procedure
```

1.2 Analysis of Merge

Equal Size

Best case, we only do n comparisons (this is when $A_n < B_1$ or $B_n < A_1$). Worst case is $2n - 1$ comparisons (this is when A_n and B_n are the two largest elements).

The average case is $2n - 2 + \frac{2}{n+1}$.

Differing Sizes

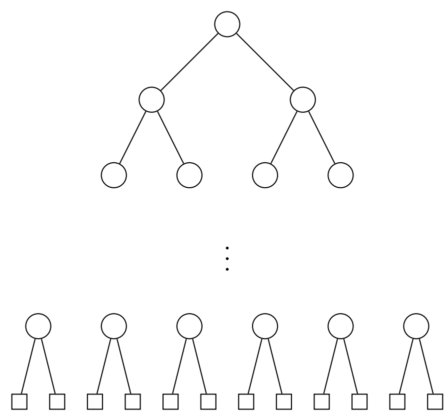
Let subarray A be of size m and subarray B be of size n where $m \leq n$. Best case is m comparisons (this is when $A_m < B_1$). Worst case is $m + n - 1$ (this is when A_m and B_n are the two largest elements). When m is much smaller than n there are better algorithms we can use. For $m = 1$ binary search gives $\approx \lg n$.

Notes

Merging is *not* in place. It can be implemented in place but those algorithms are not practical.

1.3 Analysis of Mergesort Comparisons

We will analyze mergesort using the tree method, along with the assumption that n is a power of 2.

Problem Size	Tree	Comparisons
n		$n-1$
$\frac{n}{2^1}$		$2\left(\frac{n}{2}-1\right)$
$\frac{n}{2^2}$		$4\left(\frac{n}{4}-1\right)$
$\vdots$	$\vdots$	$\vdots$
2		n
1		$0n$

We can notice that each level does exactly $2^k(\frac{n}{2^k} - 1)$ comparisons where k is the level of that branch, it follows then that since the height of the tree is $\lg n$, the total number of comparisons is

$$\begin{aligned}
\sum_{i=0}^{\lg n-1} 2^i \left(\frac{n}{2^i} - 1\right) &= \sum_{i=0}^{\lg n-1} \left(n - 2^i\right) \\
&= \sum_{i=0}^{\lg n-1} n - \sum_{i=0}^{\lg n-1} 2^i \\
&= (n \lg n) - \left(2^{\lg n-1+1} - 1\right) \\
&= (n \lg n) - (n-1) \\
&= n \lg n - n + 1
\end{aligned}$$

As a recurrence,

$$\begin{aligned}
T(n) &= T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + (n-1) \\
&= 2T\left(\frac{n}{2}\right) + n - 1, \quad T(0) = T(1) = 0
\end{aligned}$$