Module 5 Problem Set

- Student Name: Gaole Dai
- Student ID: S01435587

Problem 1

$\mathcal{O}(n\log(n))$

For 2-way merge sort, the complexity equation is $T(n) = 2T(\frac{n}{2}) + \mathcal{O}(n)$.

Similarly, 3-way merge sort, the equation is $T(n) = 3T(\frac{n}{3}) + \mathcal{O}(n)$. According to the master's theory, $a = 3, b = 3, k = 1, p = 0$ and $3 = 3^1, 0 > -1$.

The complexity is then $\mathcal{O}(n\log(n))$.

Problem 2

$\mathcal{O}(n\log(n))$

Alternating Recursive Pattern:

Top Level: Even Divide $T_e(n) = 2T_e(\frac{n}{2}) + \mathcal{O}(n)$.

Alternating Recursive Pattern:

Alt Level: Bad divide

$T_b(n) = T_b(n-1) + T(1) + \mathcal{O}(n)$

$= T_b(n-1) + \mathcal{O}(n)$

Put these together:

$T_e(n) = 2T_b(\frac{n}{2}) + \mathcal{O}(n)$

$T_b(n) = T_b(n-1) + \mathcal{O}(n)$

Then implies:

$T_e(n) = 2T_e(\frac{n}{2} - 1) + \mathcal{O}(n)$

$T_b(\frac{n}{2}) = T_e(\frac{n}{2} - 1) + \mathcal{O}(\frac{n}{2})$

$T_e(n) = 2T_e(\frac{n}{2} - 1) + \mathcal{O}(\frac{n}{2}) + \mathcal{O}(n)$

$T_e(n) = 2T_e(\frac{n}{2} - 1) + \mathcal{O}(n)$

Solve the equations, we could find upper bound:

$T_e(n) = 2T_e(\frac{n}{2} - 1) + \mathcal{O}(n)$

Note that,

$T_e(\frac{n}{2} - 1) \leq T_e(\frac{n}{2})$

So

$T_e(n) \le 2T_e(\frac{n}{2}) + \mathcal{O}(n)$

Therefore, by master's theorem

$T_e(n) = \mathcal{O}(n\log(n))$

Problem 3

Insertion Sort

The complexity of quicksort in worst case is $\mathcal{O}(n^2)$ when sorted or reverse sorted, and the expected complexity is $\mathcal{O}(n\log(n))$.

The complexity of Insertion sort in worst case is $\mathcal{O}(n^2)$, alternative complexity measure for insertion sort is $\mathcal{O}(\#inversions)$, so could get $\mathcal{O}(n)$ when $\#inversions$ is $\mathcal{O}(n)$.

Therefore, the insertion sort works better than quicksort.

Problem 4

$\mathcal{O}(k^2n)$

Induction:

Step 1: merge(1, 2), work = $2n$

Step 2: merge(1', 2'), work = $2n + n = 3n$

....

Step k-1: merge (1'''....', 2'''....'), work = $(k-1)n + n = kn$

....

Total work = $2n + 3n + 4n + ... + kn = (2 + 3 + 4 + ... + k)n = \frac{(2+k)(k-1)}{2}n = \mathcal{O}(k^2n)$

Problem 5

$\mathcal{O}(k\log(k)n)$

Idx	Phase	Work phase
0	Merge pairs	Merge pairs $\frac{k}{2} \times 2n = kn$
1	Merge $\frac{k}{2}$ $2n$	Merge Pairs $\frac{k}{4}$ pairs cost $4n = \frac{k}{4} \times 4n = kn$
2	Merge $\frac{k}{8}$ $8n$	Merge Pairs $\frac{k}{8}$ pairs cost $4n = \frac{k}{8} \times 8n = kn$
n	Total phase = $\log_2(k)$	For each phase, the cost is kn

We found that the work for every phase is kn, and the total number of phase is $\log_2(k)$.

Therefore, the total work = $\mathcal{O}(k\log(k)n)$