CS3230 Assignment 3

Zhuang Jianning
A0214561M

1. a) $T(n) = 3T(\frac{n}{2}) + n^2$

Using the Master theorem where $a = 3$, $b = 2$ and $f(n) = n^2$

Case 3 applies since $f(n) = \Omega(n^{\log_2 3 + \varepsilon})$, where $\varepsilon \approx 0.4$

$\log_2 3 \approx 1.585$

and $3(\frac{n}{2})^2 = \frac{3n^2}{4} \le cn^2$ for $c = \frac{3}{4}$

$\therefore T(n) = \Theta(n^2)$

b) $T(n) = T(\frac{n}{4}) + T(\frac{3n}{4}) + 2n$

Using the Akra-Bazzi Theorem for recurrences in the form

$T(n) = \sum_{i=1}^{k} a_i T(b_i n) + g(n)$

we find a real number p such that $\sum_{i=1}^{k} a_i (b_i^p) = 1$

Then $T(n) = \Theta\left(n^p + n^p \int_1^n \frac{g(u)}{u^{p+1}} du\right)$

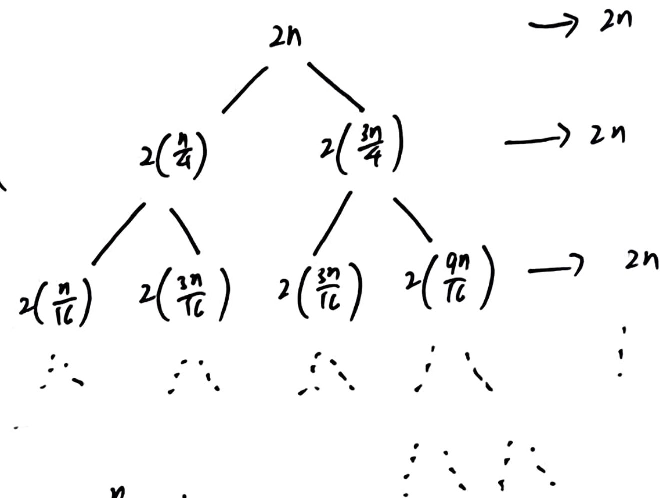
$1 \cdot (\frac{1}{4}^p) + 1 \cdot (\frac{3}{4}^p) = \frac{1}{4}^p + \frac{3}{4}^p \Rightarrow p = 1$, $g(n) = 2n$

$T(n) = \Theta\left(n^1 + n^1 \int_1^n \frac{2u}{u^2} du\right)$

$= \Theta(n + n[2\ln n - 0])$

$= \Theta(n \log n)$

OR use recursion tree & substitution method (next page) b, a.t) $T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{3n}{4}\right) + 2n$



cost at each level of the tree is 2n while the tree is still a complete binary tree

min. height of tree where each level still adds to 2n can be found by following leftmost child at each node => lower bound of algorithm

shortest simple path $n \rightarrow \frac{n}{4} \rightarrow \frac{n}{16} \rightarrow \cdots \frac{n}{4^i} = 1$

$i = \log_4 n$

lower bound on cost of algorithm is $2n(\log_4 n + 1) \geq \frac{2}{\log 4} n \log n$

$= \Omega(n \log n)$

upper bound by following rightmost child has longest simple path of length $\log_{4/3} n$

guess upper bound of $O(n \log n)$ => show $T(n) \leq c\, n \log n$ for $n \geq n_0$

$n_0 = 2 \quad T(2) = 9 \quad c = \frac{9}{2}$

$T(n) \leq c\left(\frac{n}{4}\right)\log\left(\frac{n}{4}\right) + c\left(\frac{3n}{4}\right)\log\left(\frac{3n}{4}\right) + 2n$

$= c\left(\frac{n}{4}\right)\log n - c\left(\frac{n}{4}\right)\log 4 + c\left(\frac{3n}{4}\right)\log n - c\left(\frac{3n}{4}\right)\log\frac{4}{3} + 2n$

$= cn\log n - cn\left(\frac{\log 4}{4} + \frac{3}{4}\log\frac{4}{3}\right) + 2n$

$= cn\log n - cn\left(2 - \frac{3}{4}\log 3\right) + 2n$

$\leq cn\log n$ where $c \geq \frac{2}{2 - \frac{3}{4}\log 3}$

$\therefore \quad T(n) = \Theta(n \log n)$ //

c) $T(n) = T(\frac{n}{2}) + T(\sqrt{n}) + n$

since $T(n) \le 2T(\frac{n}{2}) + n$ when $n > 4$, we guess $T(n) = O(n\log n)$

and $T(n) \ge n$, we guess $T(n) = \Omega(n)$

$T(n) = O(n\log n) \Rightarrow$ show $T(n) \le cn\log n$ for $n \ge n_0$

Let $n_0 = 2$, $c = \frac{q}{2}$

Assume $T(2) = q \Rightarrow$ Base case $(n=2)$: $T(2) = q \le c(2\log 2)$
$\le q$

By strong induction, assume $T(k) \le ck\log k$ for $n > k \ge 2$

$$
\begin{aligned}
T(n) &= T(\tfrac{n}{2}) + T(\sqrt{n}) + n \\
&\le \frac{cn}{2}\log\frac{n}{2} + c\sqrt{n}\log\sqrt{n} + n \\
&\le \frac{cn}{2}\log\frac{n}{2} + \frac{cn}{2}\log\frac{n}{2} + n \qquad (n \ge 4) \\
&= cn\log n - cn\log 2 + n \\
&= cn\log n - n(c-1) \\
&\le cn\log n \qquad (c \ge 1) \\
&= O(n\log n)
\end{aligned}
$$

$T(n) = \Omega(n) \Rightarrow$ show $T(n) \ge cn$ for $n \ge n_0$

Let $n_0 = 1$, $c \le q \le 2$

Assume $T(1) = q \Rightarrow$ Base case $(n=1)$: $T(1) = q \ge c(1)$
$= q$

By strong induction, assume $T(k) \ge ck$ for $n > k \ge 1$

$$
\begin{aligned}
T(n) &= T(\tfrac{n}{2}) + T(\sqrt{n}) + n \\
&\ge \frac{cn}{2} + c\sqrt{n} + n \\
&\ge n(\tfrac{c}{2}+1) + c\sqrt{n} \qquad (c \le 2) \\
&\ge cn \\
&= \Omega(n)
\end{aligned}
$$

$n(\frac{c}{2}+1) \ge cn$

$\frac{c}{2}+1 \ge c$

$2 \ge c$

try for tighter upper bound

Trying for a tighter upper bound $T(n) = O(n) \Rightarrow$ show $T(n) \le cn$ for $n \ge n_0$

let $n_0 = 1$, $c = q > 2$

Assume $T(1) = q \Rightarrow$ Base case $(n=1)$: $T(1) = q \le c(1)$
$\le q$

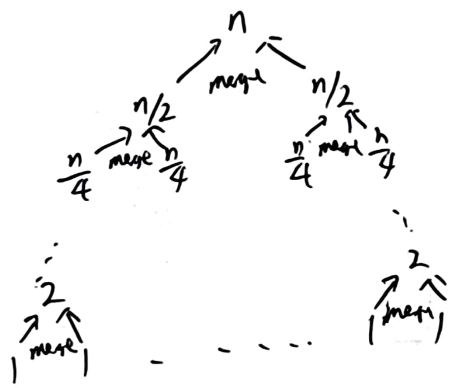
By strong induction, assume $T(k) \le ck$ for $n > k \ge 1$

$$T(n) = T\left(\frac{n}{2}\right) + T(\sqrt{n}) + n$$
$$\le \frac{cn}{2} + c\sqrt{n} + n$$
$$\le cn \qquad \left(\begin{array}{l} n \ge \frac{4c^2}{c^2 - 4c + 4} \\ c > 2 \end{array} \right)$$
$$= O(n)$$

$$\frac{cn}{2} + c\sqrt{n} + n \le cn$$
$$c\sqrt{n} + n \le \frac{cn}{2}$$
$$c\sqrt{n} \le n\left(\frac{c}{2} - 1\right)$$
$$c \le \sqrt{n}\left(\frac{c}{2} - 1\right)$$
$$\sqrt{n} \ge \frac{2c}{c-2}$$
$$n \ge \frac{4c^2}{c^2 - 4c + 4}$$

$\therefore T(n) = \Theta(n)$

2. Mergesort divides an array into 2 equal halves, recursively sorts the 2 halves and merges them into one sorted array.



Maximum comparisons:

Given 2 sorted arrays of length $\frac{n}{2}$ each, maximum number of comparisons is $n-1$ (lecture 1)

$$\text{max \# comparisons} = n-1 + 2\left(\frac{n}{2} - 1\right) + 4\left(\frac{n}{4} - 1\right) + \dots \frac{n}{2}(2-1)$$
$$= n\log n - n + 1$$

Minimum comparisons:

Given 2 sorted arrays of length $\frac{n}{2}$ each, minimum number of comparisons is $\frac{n}{2}$ when an array has elements less than the first element of the other array.

(proof using adversary argument in next page)

$$\text{min \# comparisons} = \frac{n}{2} + 2\left(\frac{n}{4}\right) + 4\left(\frac{n}{8}\right) + \dots$$
$$= \frac{n}{2}\log n$$

Proof for minimum case:

Let M be an algorithm that correctly merges using $< \frac{n}{2}$ comparisons.

There is at least 1 element in the left array which has not been compared to the current first element in the right array.

Let the left array L have all $\frac{n}{2}$ elements smaller than first element of right array and L' have first $\frac{n}{2}-1$ elements smaller than right array but last element greater.

Since M cannot differentiate between 2 inputs, at least one of the merged output is wrong

3. use a divide and conquer strategy to look for a valley

$A[0 .. n-1]$

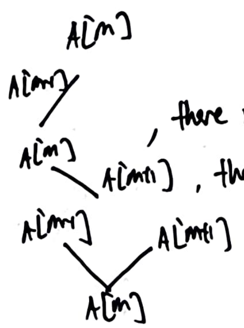
```
VALLEY (A, l, r)
    if (r-l ≤ 1)
        return min (A[l], A[r])
    m = l + ⌊(r-l)/2⌋
    if A[m] > A[m-1]
        return VALLEY ( A, l, m-1 )
    elseif A[m] > A[m+1]
        return VALLEY ( A, m+1, r )
    else
        return A[m]
```

Running time

$$
\begin{aligned}
T(n) &= T\left(\left\lfloor \tfrac{n}{2} \right\rfloor\right) + \theta(1) \\
&\le T\left(\tfrac{n}{2}\right) + c \\
&\le T\left(\tfrac{n}{4}\right) + c + c \\
&\le \underbrace{c + c + \cdots c}_{\log n} \\
&\le c \log n \\
&= O(\log n)
\end{aligned}
$$

correctness

look at midpoint m



If $A[m] > A[m-1]$, there must be a valley in the left half $0 .. m-1$

If $A[m] > A[m+1]$, there must be a valley in the right half $m+1 \cdots n-1$

else $A[m]$ is a peak

For base and edge cases, when there are ≤ 2 elements, the minimum must be a valley since either ends must have larger elements or are beyond the array eg