Date 18/09/23
Page 1

ELL 781 Mid-term Solutions

1. $\boxed{(a), (c)}$: Can be shown using Master theorem or recursion tree, observing that the number of levels of recursion is logarithmic in n, and at each level time taken is polynomial.
(b), (d) involve exponential growth of recursive calls, similar to Towers of Hanoi
(e) can be expanded as

$$T(n) = n\left(\frac{n}{2}\right)\left(\frac{n}{4}\right)\cdots\left(\frac{n}{2^{\log_2 n}}\right)$$

$$= 2^{\log_2 n} \cdot 2^{\log_2 n - 1} \cdots 2^1 \cdot 2^0$$

$$= 2^{\frac{(\log n)(\log n + 1)}{2}} = 2^{(\log n)^2/2} \cdot 2^{(\log n)/2}$$

$$= (\sqrt{n})^{\log n + 1}$$

2. $\boxed{(c)}$ $^7C_2 = 21$, so only 2 edges missing from complete graph. If both these 'missing' edges share a common [degree-4] endpoint, then the rest of the graph is a 6-clique needing 6 colours. Otherwise, there must be 2 pairs of nodes which are unlinked and distinct and both pairs can be given one colour each, hence 5-coloring possible.

3. Definition I : Yes, choose $c=1$, $T(n) \geq n^2 \ \forall n$.
Definition II : Yes, choose $c=1$, $n_0=1$.

4. (a) Suppose $T(n)$ is $\Omega(n \log n)$, i.e. we can take $T(k) \geq a\, k \log k + b$ [Set $b = T(1)$] for $k < n$, and try induction:

$$T(n) = \overbrace{T(2n/5)}^{cn} + \overbrace{T(3n/5)}^{cn} + cn$$

$$\geq a\frac{2n}{5}\log\frac{2n}{5} + b + a\frac{3n}{5}\log\frac{3n}{5} + b + cn$$

$$\geq \frac{2an}{5}(\log 2n - \log 5) + b + \frac{3an}{5}(\log 3n - \log 5) + b + c$$

$$\geq \frac{2an}{5}\left(\log n + \log\frac{2}{5}\right) + \frac{3an}{5}\left(\log n + \log\frac{3}{5}\right) + 2b + c$$

$$\geq an\log n + \left[\frac{2a}{5}\log\frac{2}{5} + \frac{3a}{5}\log\frac{3}{5} + c\right]n + 2b$$

We need to show this $\geq an\log n + b$; since $b > 0$ and $n > 0$, it will suffice if $\left[\frac{2a}{5}\log\frac{2}{5} + \frac{3a}{5}\log\frac{3}{5} + c\right] \geq 0$

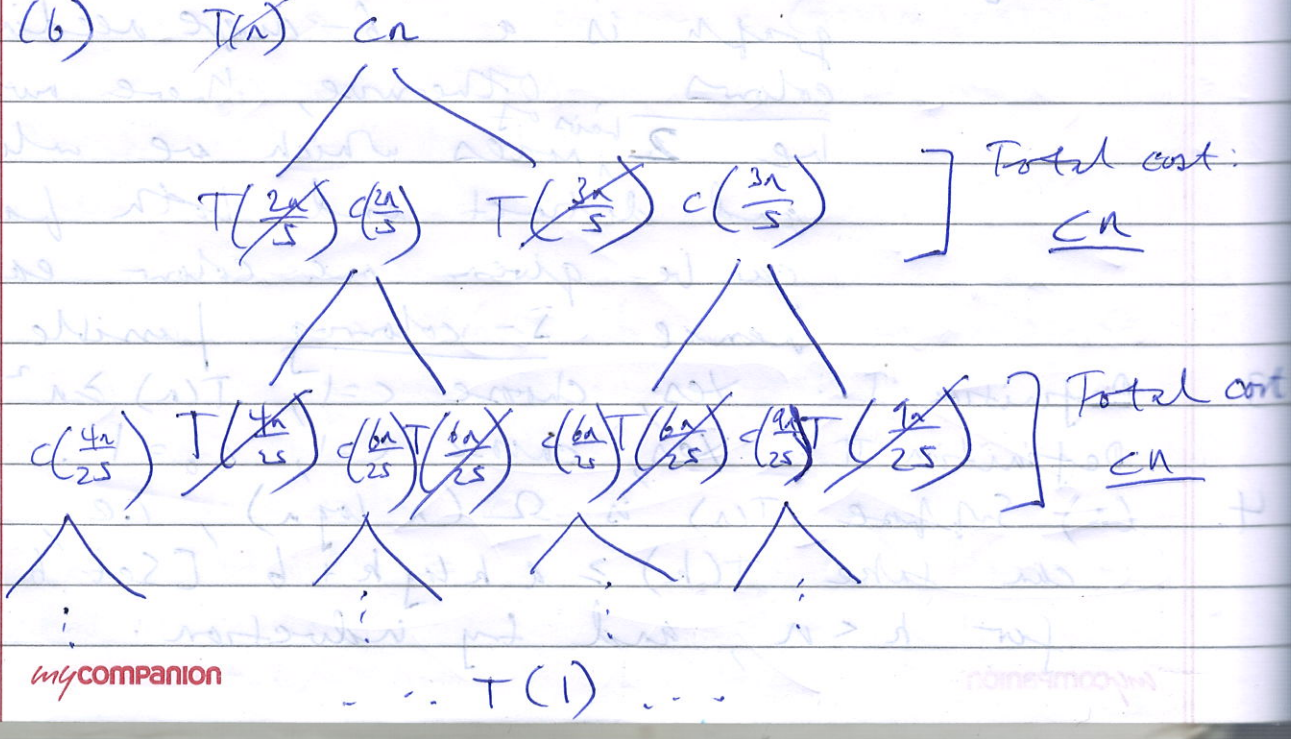
i.e., $a\left(\frac{2}{5}\log\frac{2}{5} + \frac{3}{5}\log\frac{3}{5}\right) + c \geq 0$

So we can choose

$$a \leq \frac{c}{\left(\frac{2}{5}\log\frac{5}{2} + \frac{3}{5}\log\frac{5}{3}\right)}$$

[RHS is positive]

4. (b)



Depth of tree: we need $\left(\frac{2}{5}\right)^i n = 1$, for shortest (leftmost) branch to reach base case, where i is its depth. So:

$$i = \log_{2/5}\frac{1}{n} = \log_{5/2} n$$

And each level has cost cn.
Hence total cost, even going via shortest branch, is $\underline{\Omega(n \log n)}$.

5. (a) 3 1 6 4 9 7 8 2 5 [3 comparisons]

[1 comparison to check for break] swap

2 1 6 4 9 7 8 3 5 [10 comparisons]

[1 comparison]

(b) 15 comparisons, 1 swap
(c) 2 − 1 = 1
(d) After recursive calls have returned:

1 3 4 6 9 | 2 5 7 8

Final merge:
1 2 3 4 5 6 7 8 9
$(1<2)\ (2<3)\ (3<5)\ (4<5)\ (5<6)\ (6<7)\ (7<9)\ (8<9)$

Total comparisons = $\underline{8}$

Checks on x, y cursors in main while loop:
$9 \times 2 = \underline{18}$
Checks on x, y in final two while loops: $2+1 = \underline{3}$

⇒ Sum total comparisons are 29.
If we include the for loop to copy values back to main array: 10 more
⇒ Total comparisons in merge() fn. = 39

(e) Quicksort does appear more efficient in terms of no. of comparisons, as expected. It has 1 additional swap operation, but probably still faster. And note that mergesort may also have greater memory access costs: due to merging into a new array, and then moving elements back to original array. So overall, based on this comparison of the top-level merge/partition calls, quicksort does seem faster. However the number of levels of recursion will generally be greater in quicksort, due to imbalance.

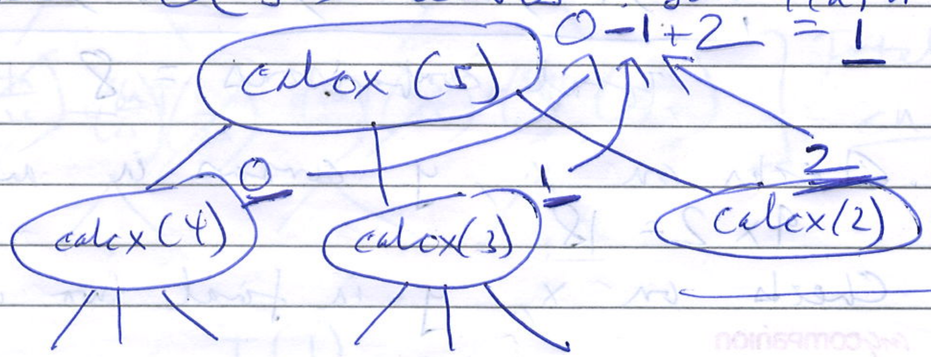
6. (a)

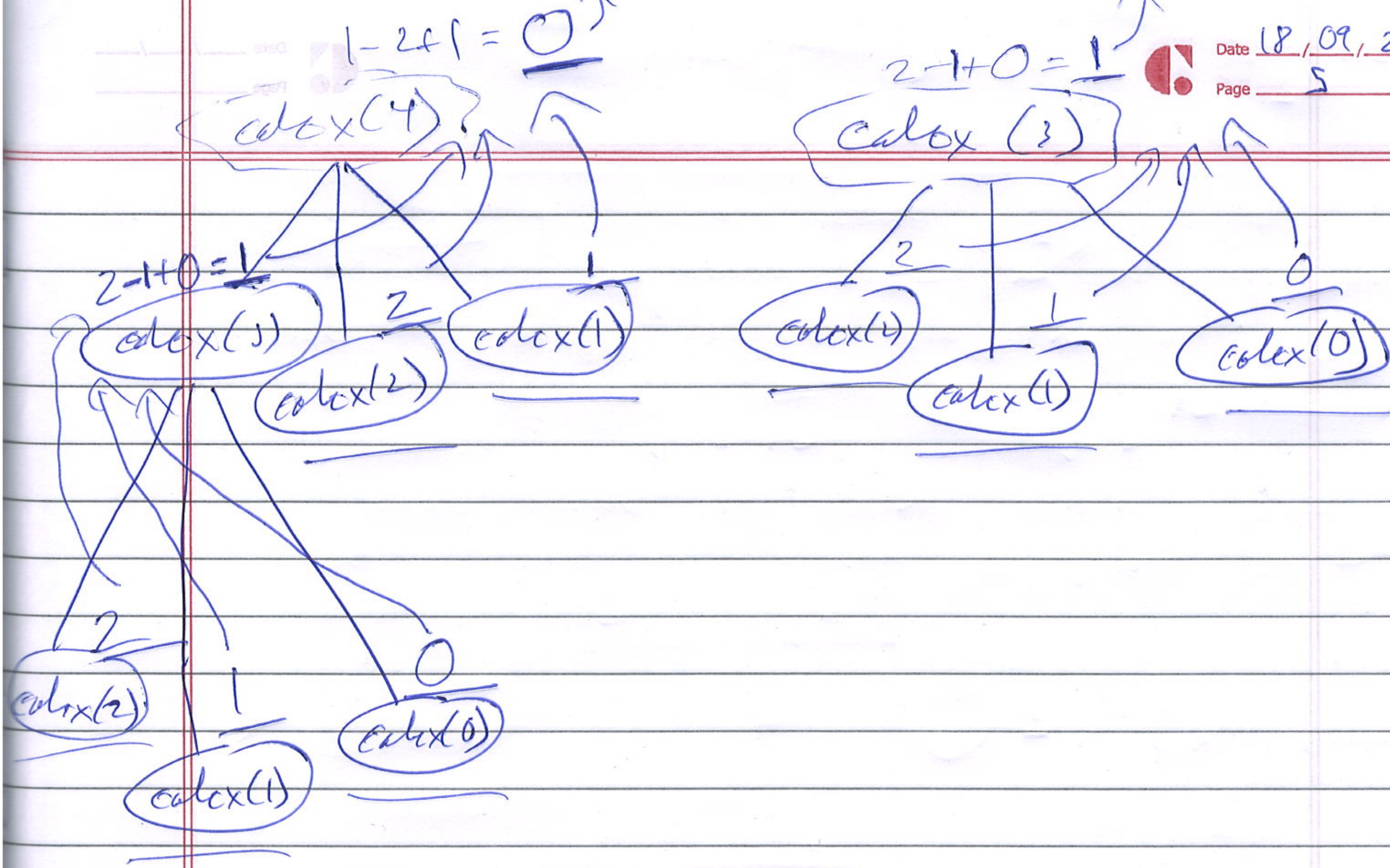
```
int calcx (int n) {
    if (n <= 2)
        return n;
    return calcx(n-1) - calcx(n-2) + calcx(n-3)
}
```

$$T(n) = \begin{cases} T(n-1) + T(n-2) + T(n-3) + d, & n \geq 3 \\ c, & n < 3 \end{cases}$$

Since prob. size is reducing linearly, recurrence tree has $O(n)$ depth, and hence $O(3^n)$ leaves. So $T(n)$ is $O(3^n$

(b)





6. (c) No; repeated computation of subproblems (here, for n=3).