Midterm Review

CS3230 AY21/22 Sem 2

Kingston Kuan
YANG Mingyang

1. True or False?

(a) $2^n = \Theta(2^{2n})$

(a) $2^n = \Theta(2^{2n})$

1. True or False?

- Suffices to show that it is impossible to find $c_1 > 0, n_0 > 0$ such that $0 \leq c_1 2^{2n} \leq 2^n$ for all $n \geq n_0$.

(a) $2^n = \Theta(2^{2n})$

1. True or False?

- Suffices to show that it is impossible to find $c_1 > 0, n_0 > 0$ such that $0 \leq c_1 2^{2n} \leq 2^n$ for all $n \geq n_0$.
- $c_1 \leq 2^{-n}$
- It is easy to see that for any c_1 that we fix, one can find a large enough n such that inequality does not hold.
- False.

1. True or False? Recall that lg denotes the logarithm with base 2.

(b) $\ln(n^2) = O(\lg n)$

(b) $\ln(n^2) = O(\lg n)$

1. True or False? Recall that lg denotes the logarithm with base 2.

- Recall logarithm-rebasing, $\log_b x = \dfrac{\log_a x}{\log_a b}$

(b) $\ln(n^2) = O(\lg n)$

1. True or False? Recall that lg denotes the logarithm with base 2.

- Recall logarithm-rebasing, $\log_b x = \dfrac{\log_a x}{\log_a b}$
- $\log_e n^2 = \dfrac{\log_2 n^2}{\log_2 e} = \dfrac{2\log_2 n}{\log_2 e}$
- Pick $c = \dfrac{2}{\log_2 e}$ for tight upper bound!
- True.

1. True or False? Recall that lg denotes the logarithm with base 2.

(c) $2^{\sqrt{\lg n}} = \omega(\lg n)$

(c) $2^{\sqrt{\lg n}} = \omega(\lg n)$

1. True or False? Recall that lg denotes the logarithm with base 2.

- If true, it means we can find $c > 0, n_0 > 0$ such that $0 \leq c\lg(n) < 2^{\sqrt{\lg(n)}}$ for all $n \geq n_0$.

(c) $2^{\sqrt{\lg n}} = \omega(\lg n)$

1. True or False? Recall that lg denotes the logarithm with base 2.

- If true, it means we can find $c > 0, n_0 > 0$ such that $0 \leq c\lg(n) < 2^{\sqrt{\lg(n)}}$ for all $n \geq n_0$.
- $\lg(c) + \lg\lg(n) < \sqrt{\lg(n)}$
- $\lg(c) < \sqrt{\lg(n)} - \lg\lg(n)$.
- Since $\sqrt{n}$ rises much faster than $\lg n$, one can reason that $\sqrt{\lg(n)}$ rises much faster than $\lg\lg(n)$ as well.
- This means that RHS increases VERY slowly on n, but is positive and unbounded anyway.
- This means that for $\forall c > 0$, $\exists n_0$ such that $\forall n \geq n_0$, inequality holds.
- True.

(c) $2^{\sqrt{\lg n}} = \omega(\lg n)$

1. True or False? Recall that lg denotes the logarithm with base 2.

Alternatively,

- $\sqrt{\lg(n)} - \lg\lg(n) = \sqrt{m} - \lg m$ where $m = \lg n$

(c) $2^{\sqrt{\lg n}} = \omega(\lg n)$

1. True or False? Recall that lg denotes the logarithm with base 2.

Alternatively,

- $\sqrt{\lg(n)} - \lg\lg(n) = \sqrt{m} - \lg m$ where $m = \lg n$
- $\sqrt{m} - \lg m > \frac{\sqrt{m}}{2}$ for $m \geq 400$. Why?
  - Set $f(m) = \frac{\sqrt{m}}{2} - \lg m$. Check that $f(m) \geq 0$ for $m = 400$. Also, $f'(m) = \frac{1}{4\sqrt{m}} - \frac{1}{m \ln 2} \geq 0$ for $m \geq 400$, so increasing. So, $f(m) \geq 0$ for $m \geq 400$.
- $\lim\limits_{n->\infty} \left(\frac{2^{\sqrt{\lg n}}}{\lg n} \right) = \lim\limits_{m \to \infty} 2^{\sqrt{m} - \lg m} \geq \lim_{m\to\infty} 2^{\frac{\sqrt{m}}{2}} = \infty$

1. True or False?

(d) $\sum_{i=1}^{n} \frac{n}{i} = o(n^2)$

(d) $\sum_{i=1}^{n} \frac{n}{i} = o(n^2)$

1. True or False?

Called the Harmonic Series, refer to Week 5 tutorial on Randomised Algorithms

- We know $\sum_{i=1}^{n} \frac{1}{i} = \Theta(\lg(n))$ (from previous tutorial)

(d) $\sum_{i=1}^{n} \frac{n}{i} = o(n^2)$

1. True or False?

- We know $\sum_{i=1}^{n} \frac{1}{i} = \Theta(\lg(n))$ (from previous tutorial)
- $\sum_{i=1}^{n} \frac{n}{i} = n \sum_{i=1}^{n} \frac{1}{i} = \Theta(n \lg(n)) = o(n^2)$
- True.

1. True or False?

(e) $\lg n = \Omega(1)$

(e) $\lg n = \Omega(1)$

1. True or False?

True. lg n is unbounded. For any choice of c, there surely exists some n such that c < lg n

2. Solve the following recurrences by providing tight asymptotic bounds.

(a) $T(n) = 3T(n/4) + \sqrt{n}$

(a) $T(n) = 3T(n/4) + \sqrt{n}$

2. Solve the following recurrences by providing tight asymptotic bounds.

Summary: Master Theorem

$$T(n) = aT(n/b) + \Theta(f(n))$$

Case 1: $f(n) = O(n^{\log_b a - \epsilon})$ ← If ε=0, it is case 2.

$T(n) = \Theta(n^{\log_b a})$

Case 2: $f(n) = \Theta(n^{\log_b a} \log^k n)$

$T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$

Case 3: $f(n) = \Omega(n^{\log_b a + \epsilon})$

$af(n/b) \leq cf(n), c < 1$

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

(a) $T(n) = 3T(n/4) + \sqrt{n}$

2. Solve the following recurrences by providing tight asymptotic bounds.

$$T(n) = 3T(n/4) + \sqrt{n}$$
$$a = 3, b = 4, n^{\log_4 3} = n^{1.262}, f(n) = n^{1/2}$$

(a) $T(n) = 3T(n/4) + \sqrt{n}$

2. Solve the following recurrences by providing tight asymptotic bounds.

Master Theorem is a good fit

$$T(n) = 3T(n/4) + \sqrt{n}$$

$$a = 3, b = 4, n^{\log_4 3} = n^{0.792}, f(n) = n^{1/2} = O(n^{\overbrace{\log_4 3 - \underbrace{(\log_4 3 - 0.5)}_{\varepsilon>0}}^{1/2}})$$

Case 1 satisfied!

Therefore $\Theta(n^{\log_4 3})$.

2. Solve the following recurrences by providing tight asymptotic bounds.

(b) $T(n) = T(n^{1/5}) + \lg n$

(b) $T(n) = T(n^{1/5}) + \lg n$

2. Solve the following recurrences by providing tight asymptotic bounds.

Master Theorem cannot be used

Unroll the recursion!

$$T(n^{1/5}) + \lg(n) = T(n^{1/25}) + \frac{\lg(n)}{5} + \lg(n)$$

$$= T(n^{1/125}) + \frac{\lg(n)}{25} + \frac{\lg(n)}{5} + \lg(n)$$

(b) $T(n) = T(n^{1/5}) + \lg n$

2. Solve the following recurrences by providing tight asymptotic bounds.

Master Theorem cannot be used

Recall infinite GP

$$\sum_{i=0}^{n-1} ar^i < \sum_{i=0}^{\infty} ar^i = \frac{a(1-0)}{(1-r)} = \frac{a}{(1-r)}$$

Unroll the recursion!

$$T(n^{1/5}) + \lg(n) = T(n^{1/25}) + \frac{\lg(n)}{5} + \lg(n)$$

$$= T(n^{1/125}) + \frac{\lg(n)}{25} + \frac{\lg(n)}{5} + \lg(n)$$

$< T(n^{small}) + \sum_{i=0}^{\infty} \frac{\lg(n)}{5^i} = \frac{5\lg(n)}{4}$, we assume base case $T(n^{small}) = \Theta(1)$

Therefore $\Theta(\lg(n))$.

Don't be over-reliant on Master Theorem!

2. Solve the following recurrences by providing tight asymptotic bounds.

(c) $T(n) = T(0.7n) + T(0.2n) + n$

(c) $T(n) = T(0.7n) + T(0.2n) + n$

2. Solve the following recurrences by providing tight asymptotic bounds.

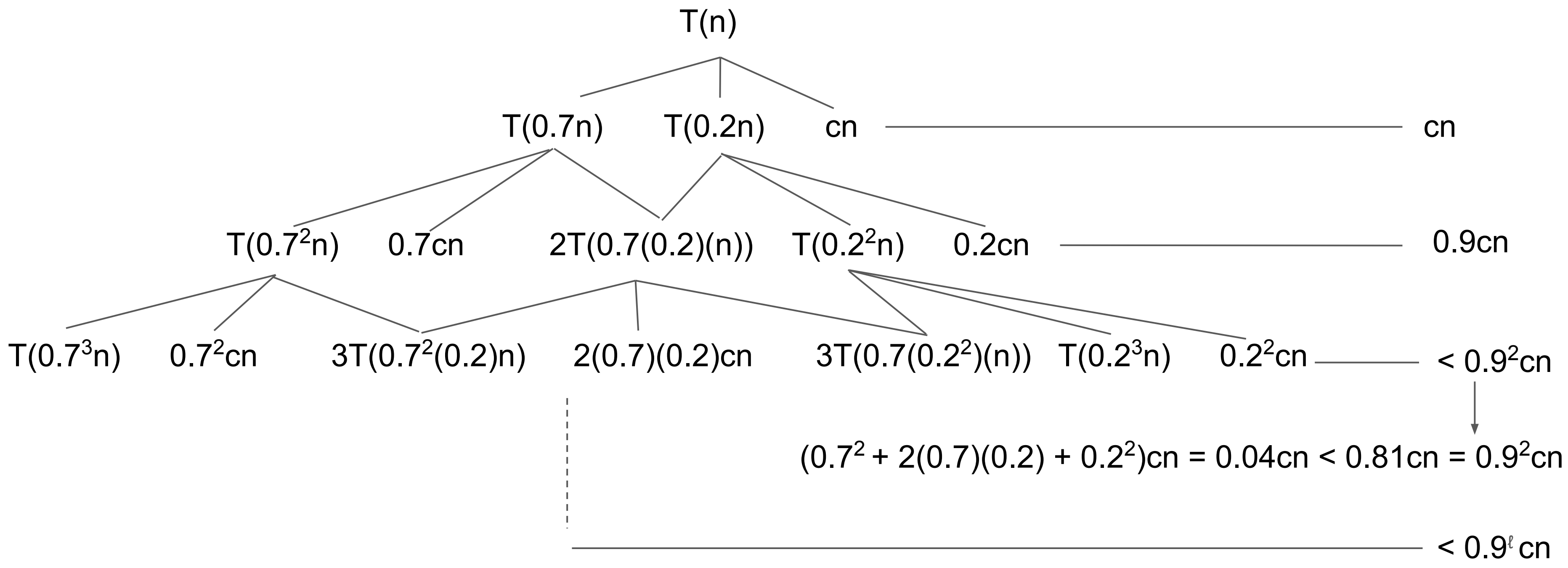
Intuition: In the first level, we need to do at least O(n) work, so we have a quick lower bound. However, note that at the same time (0.2 + 0.7) < 1, so we can guess that it is O(n).

To verify our intuition, we can use a recursion tree. Let c be such that T(n) ≤ T(0.7n) + T(0.2n) + cn for n larger than a constant.

(c) $T(n) = T(0.7n) + T(0.2n) + n$

2. Solve the following recurrences by providing tight asymptotic bounds.

By drawing the recursion tree, we have at most O(log n) levels, and at the ℓ'th level, at most $(0.9)^\ell$ cn work is being done.



(c) $T(n) = T(0.7n) + T(0.2n) + n$

2. Solve the following recurrences by providing tight asymptotic bounds.

$$T(n) \le \sum_{\ell=0}^{O(\log n)} 0.9^{\ell} cn \le cn \sum_{\ell=0}^{\infty} 0.9^{\ell} = O(n)$$

Recall infinite GP

$$\sum_{i=0}^{n-1} ar^i < \sum_{i=0}^{\infty} ar^i$$

$$= \frac{a(1-0)}{(1-r)}$$

$$= \frac{a}{(1-r)}$$

3. Prove that any deterministic algorithm for computing B must query all four input bits.

Let $B : \{0, 1\}^4 \to \{0, 1\}$ be defined by $B(x_1, x_2, x_3, x_4) = (x_1 \text{ and } x_2) \text{ or } (x_3 \text{ and } x_4)$. Prove that any deterministic algorithm for computing B must query all four input bits.

3. Prove that any deterministic algorithm for computing B must query all four input bits.

$B(x_1, x_2, x_3, x_4) = (x_1 \text{ and } x_2) \text{ or } (x_3 \text{ and } x_4)$

Suppose there exists some deterministic algorithm for computing B that skips 1 bit, say x_1.

3. Prove that any deterministic algorithm for computing B must query all four input bits.

$B(x_1, x_2, x_3, x_4) = (x_1 \text{ and } x_2) \text{ or } (x_3 \text{ and } x_4)$

Suppose there exists some deterministic algorithm for computing B that skips 1 bit, say x_1.

Such an algorithm will output the same answer for 1, 1, 0, 0 and 0, 1, 0, 0, since it does not query x_1.

However, B(1, 1, 0, 0) = 1 and B(0, 1, 0, 0) = 0, meaning an adversary can cause this algorithm to get 1 out of the 2 cases wrong.

Same argument applies for x_2, x_3, x_4 by symmetry.

Q4) Show that at least $2n$-1 comparisons are needed to merge two sorted arrays $A = [A_1, A_2, . . . , A_n]$ and $B = [B_1, B_2, . . . , B_n]$ into one sorted array by any comparison-based algorithm.

(**Hint**: Recall that your goal is to come up with two pairs of inputs (A, B) and (A', B') that have different mergings but which cannot be distinguished by an algorithm making at most $2n$-2 comparisons. Take $A = [1, 3, 5, . . . , 2n\ 1]$ and $B = [2, 4, . . . , 2n]$. Define A' and B' based on how the algorithm acts on A and B.)

- Take $A = [1,3,5,\dots,2n-1], B = [2,4,6,\dots,2n]$
- Suppose $\mathcal{M}$ indeed outputs the correct sorted array $[1,2,\dots,2n-1,2n]$ where $\mathcal{M}$ makes$< 2n-1$ comparisons.
- There must exist 2 consecutive elements in the sorted array where they were never compared, and 1 out of the 2 elements come from A, the other from B.
- Suppose "3" which is in A and "4" which is in B were never compared.
- Set $A' = A$, $B' = [2,2.99,6,\dots 2n]$, where 4 is replaced by 2.99. We choose 2.99 because B is still a sorted array as a result, but the combined and previously-sorted array will no longer be sorted.
- We know that "3" in A and "2.99" in B will also not be compared.
- $\mathcal{M}$ will thus output $[1,2,3,2.99,5,,\dots,2n-1,2n]$...
- Therefore it must use 2n-1 comparisons or more.

5. Let $\mathcal{H}$ be a universal family of hash functions mapping a universe $\mathcal{U}$ to $\{1, \ldots, M\}$. Let x and y be two different elements of $\mathcal{U}$. Are the following always true or not?

(a)

$$\Pr_{h \in \mathcal{H}}[h(x) = 1] \leq \frac{1}{M}$$

(b)

$$\Pr_{h \in \mathcal{H}}[h(x) = h(y) = 1] \leq \frac{1}{M^2}$$

Solution:

5.1.1 a

This is false. Note that if this condition were true, i.e., for all $i \in [M]$, $\Pr[h(x) = i] \leq 1/M$, then because all the probabilities still have to sum up to 1, each $\Pr[h(x) = i]$ has to equal $1/M$.

Counterexample:

- Let $\mathcal{U} = \{1, 2, 3\}$
- Let $M = 2, \mathcal{H} = \{(1, 1, 2), (1, 2, 1), (2, 1, 1)\}$
- $\Pr[h(x) = h(y)] = 1/3 \leq 1/M$ for all $x \neq y$. So, $\mathcal{H}$ is universal.
- However, $\Pr[h(x) = 1] = 2/3 > 1/M$ for each x.

5.1.2 b

Again, false. Use the same counterexample as before: We calculate $\Pr[h(x) = h(y) = 1]$. Fix $x = 1$ and $y = 2$. If a randomly chosen $h \in \mathcal{H}$ is chosen, it has $1/3$ chance of being the first one where $h(1) = 1$ and $h(2) = 1$. Therefore, $\Pr[h(x) = h(y) = 1] = 1/3 > 1/M^2$

6. Suppose you are throwing n balls into two bins, labeled A and B. Each ball goes into bin A with probability $1/2$ and bin B with probability $1/2$, and the balls are thrown independently. Let N_A be the total number of balls in bin A after all n balls have been thrown.

(a) Let X_i be the indicator random variable that equals 1 when the i-th ball falls in bin A and equals 0 otherwise. What is $\mathbb{E}[X_i]$? What is $\mathbb{E}[X_i^2]$? What is $\mathbb{E}[X_i X_j]$ for $i \neq j$?

(b) Compute $\mathbb{E}[N_A]$ and $\mathbb{E}[N_A^2]$. (**Hint**: Write N_A in terms of the indicator random variables $X_1, \ldots, X_n$.)

Solution:

5.2.1 a

1. $\mathbb{E}(X_i) = \frac{1}{2}0 + \frac{1}{2}1 = \frac{1}{2}$
2. $\mathbb{E}(X_i^2) = \frac{1}{2}0^2 + \frac{1}{2}1^2 = \frac{1}{2}$
3. X_i and X_j is independent, so $\mathbb{E}(X_i X_j) = \mathbb{E}(X_i)\mathbb{E}(X_j) = \frac{1}{4}$

5.2.2 b

1. $N_A = \sum_{i=1}^{n} X_i$
2. $N_A^2 = (\sum_{i=1}^{n} X_i)^2 = \sum_{i=1}^{n} X_i^2 + \sum_{i,j\in\{1..n\}, i\neq j} X_i X_j$
3. $\mathbb{E}(N_A) = \sum_{i=1}^{n} \mathbb{E}(X_i) = n/2$
4. $\mathbb{E}(N_A^2) = \mathbb{E}((\sum_{i=1}^{n} X_i)^2) = \sum_{i=1}^{n} \mathbb{E}(X_i^2) + \sum_{i,j\in\{1..n\}, i\neq j} \mathbb{E}(X_i X_j) = n(1/2) + (n^2 - n)(1/4) = \frac{n^2+n}{4}$