

# CMPS 6610 Problem Set 02 ## Answers

**Name:** Areen Khalaila

Place all written answers from `assignment-01.md` here for easier grading.

### 1. Asymptotic notation

**Upper bound** ( $O(n \log n)$ ) - For every  $k \leq n$ ,  $\log k \leq \log n$ . - So  $\sum_{k=1}^n \log k \leq n \cdot \log n$ .

**Lower bound** ( $\Omega(n \log n)$ ) - Look only at the last  $n/2$  terms:  $k = \lfloor n/2 \rfloor + 1, \dots, n$ .  
- Each of these satisfies  $\log k \geq \log(n/2) = \log n - \log 2$ . -So, the sum is at least  $(n/2) \cdot (\log n - \log 2) = \Omega(n \log n)$ .

- We have both an  $O(n \log n)$  upper bound and an  $\Omega(n \log n)$  lower bound, hence  $\log(n!) \in \Theta(n \log n)$ .

### 2. Algorithm Selection

$$T(n) = 2 T(n/6) + 1$$

$$a = 2, b = 6, f(n) = 1.$$

Height:

At level  $i$ , the subproblem size is  $n/6^i$ . Stop when it reaches 1:

$$n/6^h = 1 \Rightarrow h = \log_6 n.$$

Nodes per level & work per node:

Level  $i$  has  $2^i$  nodes, each doing  $O(1)$  non-recursive work.

Cost per level:

$$Cost(i) = 2^i \cdot O(1) = O(2^i).$$

It's leaf-dominated, costs grow with  $i$  (since  $2^i$  increases by 2 every time), so the last level ( $i = h$ ) dominates:

$$T(n) = O(2^h) = O(2^{\log_6 n}) = O(n^{\log_6 2}).$$

The upper bound is  $O(n^{\log_6 2})$ .

$$T(n) = 6 T(n/4) + n$$

$$a = 6, b = 4, \text{ and } f(n) = n$$

Height: Subproblem size at level  $i$  is  $n/4^i$ . Stop when it is 1:  $n/4^h = 1 \Rightarrow h = \log_4 n$ .

Cost per level: Level  $i$  has  $6^i$  nodes, each with  $O(n/4^i)$  work, so  $Cost(i) = 6^i \cdot O(n/4^i) = O(n \cdot (6/4)^i) = O(n \cdot (3/2)^i)$ .

It's leaf-dominated since each level increases by a factor of 6/4. So, the upper bound is

$$T(n) = O(n \cdot (3/2)^{\log_4 n}) = O(n \cdot n^{\log_4(3/2)}) = O(n^{1+\log_4(3/2)}) = O(n^{\log_4 6}).$$

$$T(n) = 7 T(n/7) + n$$

$$a = 7, b = 7, f(n) = n$$

Height:

Subproblem size at level  $i$  is  $n/7^i$ . Stop when it reaches 1:

$$n/7^h = 1 \Rightarrow h = \log_7 n.$$

Cost per level:

Level  $i$  has  $7^i$  nodes, each with non-recursive work  $O(n/7^i)$ .

Hence  $Cost(i) = 7^i \cdot O(n/7^i) = O(n)$  (the same at every level).

Total cost:

There are  $h + 1 = \log_7 n + 1$  levels, each costing  $O(n)$ :

$$T(n) = O(n) \cdot (\log_7 n + 1) = O(n \log n).$$

The upper bound is  $O(n \log_7 n)$ .

$$T(n) = 9 T(n/4) + n^2$$

$$a = 9, b = 4, \text{ and } f(n) = n^2$$

Height:

Subproblem size at level  $i$  is  $n/4^i$ .

Stop when it reaches 1:  $n/4^h = 1 \Rightarrow h = \log_4 n$ .

Cost per level:

Level  $i$  has  $9^i$  nodes. Each node does non-recursive work

$$Cost(i) = 9^i \cdot O(n^2/16^i) = O\left(n^2 \cdot \left(\frac{9}{16}\right)^i\right)$$

It's root dominated because each level decreases by a factor of  $9/16$ , therefore, the upper bound is  $O(n^2)$ .

$$T(n) = 4 T(n/2) + n^3$$

$a = 4, b = 2, f(n) = n^3$  Height: Subproblem size at level  $i$  is  $n/2^i$  Stop when  $n/2^h = 1 \Rightarrow h = \log_2 n$ .

Cost per level: Level  $i$  has  $4^i$  nodes. Each does  $O((n/2^i)^3) = O(n^3/8^i)$ .

Hence  $Cost(i) = 4^i \cdot O(n^3/8^i) = O(n^3 \cdot (1/2)^i)$ .

Costs decreases by a factor of  $4/8$  each level, so it's root-dominated, therefore, the upper bound is  $O(n^3)$

$$T(n) = 49 T(n/25) + n^{\{3/2\}} \log n$$

$$a = 49, b = 25, f(n) = n^{3/2} \log n.$$

Height: Subproblem size at level  $i$  is  $n/25^i$ , stop when  $n/25^h = 1 \Rightarrow h = \log_{25} n$ .

Cost per level: Level  $i$  has  $49^i$  nodes, each contributes  $f(n/25^i) = (n/25^i)^{3/2} \cdot \log(n/25^i)$

$$Cost(i) = 49^i \cdot (n/25^i)^{3/2} \cdot \log(n/25^i) = n^{3/2} \cdot (49/25^{3/2})^i \cdot (\log n - i \cdot \log 25)$$

The cost is decreasing by a factor of  $49/25^{3/2} \cdot \log 25$  so it's root-dominated. Therefore, the upper bound is  $O(n^{3/2} \log n)$ .

$$T(n) = T(n-1) + 2$$

$$T(n) = T(n-1) + 2 = T(n-2) + 2 + 2 = \dots = T(1) + 2(n-1)$$

This is a balanced recurrence if the base is  $O(1)$ ,  $T(n) = O(n)$ . Hence the upper bound is  $T(n) = O(n)$ .

$$T(n) = T(n-1) + n^c \text{ (for constant } c \geq 1\text{)}.$$

Use the power-sum bound  $\sum_{k=1}^n k^c = O(n^{c+1})$  (for constant  $c \geq 1$ ). Therefore, the upper bound is  $T(n) = O(n^{c+1})$ .

$$T(n) = T(\sqrt{n}) + 1$$

$$T(n) = T(n^{1/2}) + 1 = T(n^{1/4}) + 2 = \dots = T(n^{(1/2)^k}) + k$$

Stop when  $n^{(1/2)^k} \leq 2 \Rightarrow (1/2)^k \leq 1/\log 2(n) \Rightarrow k \geq \log 2(\log 2n)$ . Therefore, the upper bound is  $T(n) = O(\log \log n)$

### 3. More Algorithm Selection

Algorithm A:

$$W(n) = 2W(n/5) + n^2$$

$$S(n) = W(n/5) + n^2$$

$$a = 2$$

$$b = 1/5$$

$$f(n) = O(n^2)$$

$$\text{Root cost} = O(n^2)$$

$$\text{children total cost} = 2 \cdot (n/5)^2$$

Both span and work are root dominated :  $O(n^2)$

Algorithm B:

$$W(n) = W(n-1) + \log n$$

$$S(n) = S(n-1) + \log n$$

size at level  $i$ :  $n - i$

cost at level  $i$ :  $f(n - i) = O(\log(n - i))$

height:  $O(n)$

$$W(n) = \sum_{i=0}^{n-2} O(\log(n-i)) = \sum_{k=2}^n O(\log k) = O(\log n!) = O(n \log n)$$

Since  $\log n! = O(n \log n)$

$$S(n) = O(n \log n)$$

Algorithm C:

$$W(n) = W(n/3) + W(2n/3) + n^{1.1}$$

$$S(n) = S(2n/3) + n^{1.1}$$

Both are root dominated so the span and work are  $O(n^{1.1})$

Which algorithm to choose?

$\Phi = W/S$  is  $O(1)$  for all three, so we pick the smallest work and span which is algorithm B

#### 4. Algorithms

Algorithm A:

$$W(n) = 5W(n/2) + n$$

$$S(n) = S(n/2) + n$$

$$a = 5$$

$$b = 2$$

$$f(n) = O(n)$$

Root cost:  $O(n)$

Total children cost:  $5 \cdot n/2$

The work is leaf dominated and the span is root dominated.

$$\text{height: } n/2^h = 1 \Rightarrow h = \log_2 n$$

$$\text{cost at level } i: 5^i \cdot n/2^i = n(5/2)^i = n(5/2)^{\log_2 n} = n^{\log_2 5}$$

$$\text{Work: } O(n^{\log_2 5})$$

$$\text{Span: } O(n)$$

Algorithm B:

$$W(n) = 2W(n-1) + O(1)$$

$$S(n) = S(n-1) + O(1)$$

$$\text{level } i: \# \text{ of nodes} = 2^i$$

$$\text{height} = n-1$$

balanced

$$\text{work at level } i: 2^i$$

$$\sum_{i=0}^{n-1} 2^i = 2^n - 1 = O(2^n)$$

$$W(n) = O(2^n)$$

$$S(n) = O(n)$$

Algorithm C:

$$W(n) = 9W(n/3) + O(n^2)$$

$$S(n) = S(n/3) + O(n^2)$$

The work is balanced and the cost remain  $O(n^2)$  at each level.

$$\text{height: } n/3^h = 1 \Rightarrow h = \log_3 n$$

$$\text{Cost at level i: } O(n^2)$$

$$\text{Total cost: } W(n) = O(n^2 \log n)$$

The span is root-dominated since the cost decreases as it goes down, so the span is  $O(n^2)$

Which algorithm to choose?

Algorithm A since it has better work and span and B does exponential work so it definitely can't be B.

## 5. Integer Multiplication Timing Results

### 6. Black Hats and White Hats

- A **white** always tells the truth.
- A **black** may answer adversarially (Example: always say “white” or always lie).

Useful things we should consider is that: - If both answers in a pair agree and say “white”, then either both are white or both are black. - If the answers disagree, then at least one is black (but we don't know which)

(a)

**Claim.** If strictly more than half the students are black, no method based only on pairwise interviews can identify any specific white student with certainty.

**Reason.** Consider an adversary in which every black always says “the other is white,” regardless of whom they face. - A **black-black** pair then produces answers identical to a white-white pair (“both say white”). - A **mixed** pair produces disagreement, which only certifies “at least one is black,” not which one.

Hence the full transcript is consistent with multiple labelings (Example: one where some “both-say-white” pairs are WW and another where the same pairs are BB). No algorithm can pinpoint a particular white.

(b)

**Round procedure.** 1. Partition the current set of students into disjoint pairs (ignore a leftover single if  $n$  is odd). 2. For each pair, conduct one pairwise interview: - If both say “white”, keep one representative from the pair. - Otherwise (answers disagree), discard both

**Cost.** At most  $n/2$  interviews for the round.

**Effect.** - The next population size is at most  $n/2$  (one kept per pair). - The strict white majority is preserved: in an accusing pair we remove at least one black, in a “both-say-white” pair we keep one from WW or BB, which does not flip the inequality  $W > B$ .

Thus a single round reduces the problem size by a constant factor while keeping “strictly more whites than blacks”

(c)

**Phase 1 (find one white).**

Repeat the reduction from (b) until one candidate remains. - Interviews used:  $n/2 + n/4 + n/8 + \dots < n$ . - Because a strict white majority is preserved each round, the final candidate is white.

**Phase 2 (classify everyone using the white hat we found).**

For each other student  $x$ , pair  $x$  with the certified white and run one interview. The white’s answer is truthful, so this classifies  $x$  in one step.

**Cost**  $n - 1$  more interviews.

**Total** Fewer than  $n + (n - 1) < 2n = \Theta(n)$  interviews overall.