Homework 1

Thomas Kim tsk389

David Munoz dam2989

CS331 Algorithms and Complexity

Problem Q1(a). Sort the following list of functions in ascending order of growth rates.

1. $f_7(n) = \sqrt{n}$
2. $f_4(n) = n(logn)^3$
3. $f_5(n) = n^4$
4. $f_6(n) = 2^{2^{log(logn)}}$
5. $f_3(n) = n^{logn}$
6. $f_2(n) = 2^{n^3}$
7. $f_1(n) = 2^{2^n}$

Problem Q1(b). State whether Alice or Bob have provided correct algorithms. Also, either prove the algorithm to be right, or show how it is wrong.
The algorithms in question have been reproduced below with numbered lines for convenience:

Alice's Algorithm

1: Initially $sum = 0$ and $count = 0$
2: **while** the input stream is not empty **do**
3: Read the integer i
4: Add i to sum
5: Report average as $sum/count$
6: Increase $count$ by 1
7: **end while**

Bob's Algorithm

1: Initially $average = 0$ and $count = 0$
2: **while** the input stream is not empty **do**
3: Read the integer i
4: $average := (average \times count + i)/(count + 1)$
5: Increase $count$ by 1
6: $Reportaverage$
7: **end while**

Solution:
Alice's algorithm is incorrect

Proof. Suppose the set of numbers taken as input to Alice's algorithm $S = \{1\}$
By line 5 of Alice's algorithm, $count = 0, sum = 1, \frac{sum}{count} = \frac{1}{0}$.
The answer is undefined when it should be 1.

□

Bob's algorithm is correct

Proof. Let $Average(S_n)$ be defined as the mean of the subset of S containing the first n elements. Let S_n be defined as the nth element of the set S. Let S be a set of N integers.
By the definition of mean of set S, $Average(S_N) = \frac{\sum_{i=1}^{N} S_i}{N}$.
Base case:
$Average(S_1) = \frac{\sum_{i=1}^{1} S_i}{1} = S_1$
By line 4 of Bob's algorithm, $Average(S_1) = 0 \times 0 + S_1/(0+1) = S_1$
By simple observation, Bob's algorithm is correct when computing the average of a set consisting of 1 number.
Induction hypothesis:
By line 4 of Bob's algorithm, $Average(S_{n+1}) = Average(S_n) * count + i)/(count + 1)$
By line 5 of Bob's algorithm, when processing integer S_{n+1}, $count = n$
$Average(S_{n+1}) = (Average(S_n) * n) + i)/(n+1)$
$Average(S_{n+1}) = (\frac{\sum_{i=1}^{n} S_i}{n} * n + S_{n+1})/(n+1)$
$Average(S_{n+1}) = ((\sum_{i=1}^{n} S_i) + S_{n+1})/(n+1)$
$\{((\sum_{i=1}^{n} S_i) + S_{n+1}) = (\sum_{i=1}^{n+1} S_i)\}$
$Average(S_{n+1}) = (\sum_{i=1}^{n+1} S_i)/(n+1)$
This matches line 4 of Bob's Algorithm.

□

Problem Q1(c). Prove the following using induction for every $a \neq 1$

$$\sum_{i=0}^{n-1} a^i = \frac{1-a^n}{1-a}$$

Proof. **Base case:** $n = 0$

$$\begin{aligned} \sum_{i=0}^{0} a^i &= 1 \\ &= \frac{1-a^1}{1-a} \\ &= \frac{(1-a^1)}{1-a} \end{aligned}$$

Induction Hypothesis:

$$\begin{aligned}\sum_{i=0}^{n} a^i &= \sum_{i=0}^{n-1} + a^n \\ &= \frac{1-a^n}{1-a} + a^n \\ &= \frac{1-a^n}{1-a} + \frac{(1-a)a^n}{1-a} \\ &= \frac{1-a^n+(a^n-a^{n+1})}{1-a} \\ &= \frac{1-a^n+a^n-a^{n+1}}{1-a} \\ &= \frac{1-a^{n+1}}{1-a}\end{aligned}$$

□

Problem Q2(a). Show that it is always possible to guess a number between 1 and 1 million using 20 questions.

Proof. Suppose the number in question is N. Suppose we arranged the numbers into a binary tree T where one node n's left and right children are given to be $n + (n/2)$ and $n - (n/2)$ respectively
Suppose the root of the tree is given to be $500,000$
Execute binary search on the tree T, where the current node is A.
Each question is "Is $N > A$?", and the algorithm follows the left child branch when the response is "no" and the right child when the response is "yes"
The complexity of binary search is $O(log_2(n))$
$log_2(1,000,000) \leq 20 \implies$ it is possible to guess the number using 20 or more questions.

□

Problem Q2(b). Show that it possible to guess a number 1-N given 3 "strikes" (questions answered no) and $O(n^{\frac{1}{3}})$ questions.

Phase 1: $g_1 = 0$, Guessing number $1 < n < N$
Define a mapping $f(x) \to x^{\frac{1}{3}}$
for $g_1 = 0 \ldots N^{\frac{1}{3}}$ **do**
    Ask "Is $f(n) > g_1$?"
    if Answer = "No" **then**
        Save the current value of g_1 and go to Phase 2
    end if
end for
Phase 2: $g_2 = 0$
Define a mapping $g(x) \to (x - (g_1 - 1)^3)$
for $g_2 = 0 \ldots \sqrt{3g_1^2 - 3g_1 + 1}$ **do**
    Ask "Is $(g(f(n)) > g_2$?"
    if Answer = "No" **then**

14: Save the current value of g_2 and go to Phase 3
15: **end if**
16: **end for**
17: Phase 3: $g_3 = 0$
18: Define a mapping $h(x) \rightarrow (x - (g_2 - 1)^2)$
19: **for** $g_3 = 0 \dots 2g_2 - 1$ **do**
20: Ask "Is $h(g(x)) > g_3$?"
21: **if** Answer = "No" **then**
22: Guess $h^{-1}(g^{-1}(x))$
23: **end if**
24: **end for**

Lemma 1a: At the beginning of phase 2, up to $N^{\frac{1}{3}}$ questions will have been asked.

Proof. By line TODO, phase 1 is guaranteed to terminate after the loop finishes, which finishes in $N^{\frac{1}{3}}$ iterations.
By line TODO, during each iteration of the loop in phase 1, precisely 1 question is asked.
$\implies$ no more than $N^{\frac{1}{3}}$ questions can be asked during phase 1.
By line TODO and line TODO, phase 1 can only directly precede phase 2.
$\implies$ Phase 2 immediately follows phase 1, so no questions can be asked between phase 1 and phase 2
$\implies$ At the beginning of phase 2, up to $N^{\frac{1}{3}}$ questions can have been asked.

□

Lemma 1b: At the beginning of phase 2, precisely 1 strike will be used.

Proof. By line TODO, if 1 strike is used during phase 1, the algorithm will immediately go to phase 2.
Suppose no strikes are used during phase 1.
$\implies n^{\frac{1}{3}} > N^{\frac{1}{3}}$ by lines 2 and 3, since $g_1 = N^{\frac{1}{3}}$ during the last iteration of the loop.
$\implies n > N$, which means n is not in the range 1-N.
Therefore, no more than 1 strikes can be used during phase 1, and at least 1 strike must be used during phase 1
Phase 1 always leads immediately to phase 2, from the proof of lemma 1.
$\therefore$ precisely 1 strike will be used at the beginning of phase 2.

□

Lemma 2: At the beginning of phase 2, the possible values of n will be bounded by $(g_1 - 1)^3 < n \leq (g_1)^3$

Proof. By lemma 1b, precisely 1 strike must have been used during phase 1.
$\implies$ line 5 was executed precisely 1 time.
Suppose $g_1 = \alpha$ when line 5 was executed.
During the previous iteration of the loop, line 5 must not have executed, because otherwise the loop would terminate.
During the previous iteration of the loop, $g_1 = \alpha - 1$.
Since line TODO did not execute during the previous iteration, $\alpha - 1 < n^{\frac{1}{3}}$ by line TODO and 4.
Since line TODO executed during this iteration, $\alpha \geq n^{frac13}$ by line TODO and 4.
$\implies (\alpha - 1)^3 < n \wedge (\alpha)^3 \geq n \implies (\alpha - 1)^3 < n \leq (\alpha)^3$

By line TODO of this proof, $g_1 = \alpha \implies (g_1 - 1)^3 < n \leq (g_1)^3$

□

Lemma 3a: During phase 2, up to $\sqrt{3N^{\frac{2}{3}} - 3N^{\frac{1}{3}} + 1}$ questions will be asked

Proof. By **Lemma 2** and the definition of $g(x)$, $0 < g(n) < \sqrt{3g_1^{\frac{2}{3}} - 3g_1^{\frac{1}{3}} + 1}$
By line TODO of the algorithm, g_1 is bounded by $N^{\frac{1}{3}}$
By line TODO of the algorithm, the loop iterates $\sqrt{3g_1^{\frac{2}{3}} - 3g_1^{\frac{1}{3}} + 1}$ times
$\therefore \sqrt{3g_1^{\frac{2}{3}} - 3g_1^{\frac{1}{3}} + 1}$ is bounded by $\sqrt{3N^{\frac{2}{3}} - 3N^{\frac{1}{3}} + 1}$

□

Lemma 3b: At the beginning of phase 3, precisely 2 strikes will be used.

Proof. By line TODO, if 1 strike is used during phase 2, the algorithm will immediately go to phase 3.
Suppose no strikes are used during phase 2.
This means during the last iteration, $g_2 = (g_1)^3 - (g_1 - 1)^3 \wedge f(n) > g_2$
By **lemma 2**, $(g_1 - 1)^3 < n \leq (g_1)^3 \implies 0 < n - (g_1 - 1)^3 < (g_1)^3 - (g_1 - 1)^3$
By line TODO, the algorithm iterates $(g_1)^3 - (g_1 - 1)^3$ times, iterating through $0 < g_2 \leq (g_1)^3 - (g_1 - 1)^3$
By line TODO of this proof, $0 < f(n) < (g_1)^3 - (g_1 - 1)^3$
$\therefore$ if $f(n) > g_2$ during the last iteration, $f(n) > (g_1)^3 - (g_1 - 1)^3 \implies n > (g_1)^3$
$\implies$ at least 1 strike must be used during phase 2.
By line TODO of this proof, and **lemma 1b**, precisely 2 strikes must be used by the end of phase 2
Since phase 2 always immediately leads to phase 3, precisely 2 strikes have been used at the beginning of phase 3.

□

Lemma 4: At the beginning of phase 3, the possible values of $g(h(n))$ will be bounded by $0 < g(h(n)) \leq 2(\sqrt{3(N^{\frac{2}{3}} - N^{\frac{1}{3}}) + 1}) - 1$

Proof. At the beginning of phase 3, the possible vales of $h(n)$ will be bounded by $(g_2 - 1)^2 < h(n) \leq (g_2)^2$
Subtract $(g_2 - 1)^2$ from all sides of the inequality.
Note that $g_1 \leq N^{\frac{1}{3}} \wedge g_2 \leq \sqrt{(g_1)^3 - (g_1 - 1)^3}$
$0 < g(h(n)) \leq 2(\sqrt{3(N^{\frac{2}{3}} - N^{\frac{1}{3}}) + 1}) - 1$

□

Lemma 5: $g^{-1}(x)$ and $h^{-1}(x)$ are well defined and bijective.

Proof. $g^{-1}(x) = (x + (g_1 - 1)^3)$
$h^{-1}(x) = (x + (g_2 - 1)^2)$
Both are linear, and are therefore bijective.

□

Proof of correctness:

Proof. By **Lemma 4**, the possible values of n are bounded by $(\beta - 1)^3 \leq n \leq (\beta)^3$.
By line TODO! of the algorithm, Phase 3 is a linear search over this range By **Lemma 5**, $g_3 - 1 < n \wedge n \leq g_3 \implies n = g_3$ By line TODO! of the algorithm, the value guessed is $g_3 = n$. □

Proof of termination within $O(N^{\frac{1}{3}})$ time:

Proof. By lemma 1a Phase 1 uses $N^{\frac{1}{3}}$ time.
By lemma 2a Phase 2 uses $\sqrt{3N^{\frac{2}{3}} - 3N^{\frac{1}{3}} + 1}$ time.
By lemma 4 and the fact that phase 3 is linear search, Phase 3 uses $2(\sqrt{3(N^{\frac{2}{3}} - N^{\frac{1}{3}}) + 1}) - 1$ time.
No term grows faster than $N^{\frac{1}{3}}$

□

Proof of termination using precisely 3 strikes:

Proof. By **Lemma 3**, At the beginning of phase 3, 2 strikes have been used up.
By line TODO! of the algorithm, after receiving the 3^{rd} strike, the algorithm will halt.

□

Problem Q3. Modify the Gale-Shapley Algorithm to fulfill the following definition of stability, and prove correctness
Stability implies none of the following types of instability:

- Type 1 instability: There is a candidate X and a team Y , such that X is not assigned to Y, but X prefers Y more than his assigned team, and Y prefers X more than at least one of the candidates assigned to it.
- Type 2 instability: There exists a candidate X who is not assigned to any team, but there is a team Y with an empty spot, or a candidate X_0 assigned to it such that Y prefers X more than X_0.
- Type 3 instability: There exists a team with one or more empty spots, but there is an unassigned candidate.

Suppose the set of teams $G = \{T_1, T_2, \ldots, T_n\}$ where each team is a set of empty spots $T_i = \{T_{i,1}, T_{i,2}, \ldots, T_{i,m}\}$
Let the set of men M be the set of empty slots.
Let the set of women W be the set of candidates.
The preference list for each empty slot is the same as the preference list for its respective team.
For each candidate, expand each team T_i on their preference list into a list of slots $\{T_{i,j}\}$, ordered such that lower j are more preferred
Now simply run the modified G-S algorithm on these two sets M and W as shown below.

```
1: Pick one unmatched man x
2: if x has not yet proposed to a woman in his list then
3:     x proposes to the top woman y in his list who he has not yet proposed to
4: else
5:     mark x done
6:     goto 1
7: end if
8: if y is free then
9:     y temporarily accepts x
10: else if y is matched with x' ≠ x then
11:    if x <_y x' then
```

12: y accepts x temporarily, x' becomes free
13: **else**
14: y rejects x
15: **end if**
16: **end if**
17: Repeat until every man is marked done or is matched

Lemma 1: The G-S algorithm will either match every woman to a man or every man to a woman given 2 sets of unequal length

Proof. Suppose $|W| > |M|$.
The modified G-S algorithm will not terminate until every man is matched or has proposed to every woman on his list.
Suppose some man m is unmatched when the algorithm completes.
If m is unmatched, he must been rejected by every woman $w \in W$
Every woman $w \in W$ must already be matched to a man $m' : m' >_w m$
However, $|W| > |M|$, so there must be some woman who is not matched to a man.
$\therefore$ if $|W| > |M|$, every man must be matched by the end of the modified G-S algorithm.

Suppose $|M| > |W|$.
The modified G-S algorithm will not terminate until every man is matched or has proposed to every woman on his list.
Suppose some woman w is unmatched when the algorithm completes.
If w is unmatched, she must not have been proposed to by any man $m \in M$
If a woman has not been proposed to, every man must already be matched.
However, $|M| > |W|$, so there must exist some $m \in M$ which is unmatched.
$\therefore$ if $|M| > |W|$, every woman must be matched by the end of the modified G-S algorithm.

□

Fact 1: The set S_{G-S} resulting from running the G-S algorithm will never contain an unstable pair, where instability is defined as $(m, w) \in S_{G-S} : \exists m', w', m <_w m' \wedge w <_m w'$
Fact 2: G-S is man-optimal **Proof of Type 1 instability:**

Proof. Suppose there is candidate X and a team Y.
Suppose the slots of team Y are given to be $Y_1, Y_2, \ldots, Y_n$
Suppose there is type 1 instability, where X is not assigned to Y, but X prefers Y and Y prefers X.
This means, the pairs $(X, Y_i'), (X', Y_j)$ are part of the resulting matching.
Since X, X' are drawn from the set of men, and Y_i', Y_j are drawn from the set of women, this means $\exists (m, w) \in S_{G-S} : \exists m', w', m <_w m' \wedge w <_m w'$
However, G-S is guaranteed not to have such instabilities, therefore such a pairing cannot exist.

□

Proof of Type 2 instability:

Proof. There cannot exist a candidate who is not assigned to any team while there is an empty slot by **Lemma 1**.
G-S is man-optimal, so each man (slot) will get its best valid partner (candidate). Therefore, there cannot exist such an X_0 that the team prefers X over X_0.

□

Proof of Type 3 instability:

Proof. By **lemma 1**, the modified G-S algorithm must either match all women or all men.
Case 1: Suppose the modified G-S algorithm matches all men but not all women.
By definition, this means all of the slots are matched and there are some unassigned candidates.
By definition, this means no team exists with an empty slot.
Therefore, there cannot exist a team with an empty slot if there are some unassigned candidates.
Case 2: Suppose the modified G-S algorithm matches all women but not all men.
This means there exist no unmatched candidates and there are some empty slots.
By definition, this means some team exists with an empty slot.
Therefore, there cannot exist unmatched candidates if some team exists with an empty slot.
Case 3: Suppose all men and women are matched.
By definition, this means no team has an empty slot and there are not unassigned candidates.

□