Homework #1

Problem. 5

(a) Complexity (10 points)

Prove or disprove each of the following statements:

1. $\ln n = O(\sqrt{n})$ (1pt)
2. $928 \log_{830} n = \Theta(\ln n)$ (1pt)
3. $\Sigma_{i=1}^{n} i^3 = \Theta(n^4),\ n \in \mathbb{Z}^+$(1pt)
4. $n^{\pi}(\log n)^{112} = O(n^3\sqrt{n})$ (2pt)
5. $\lfloor \log \log n \rfloor ! = O(n)$ (2pt)

6(a). $\ln(n!) = \Theta(n \ln n),\ n \in \mathbb{Z}^+$(2pt)

6(b). prove $\ln(n!) - n \ln n + n = \Theta(\ln n)$ (1pt)

Please note that you can't use 6(b) to prove 6(a) without your proof of 6(b).

We say that $f(n) = o(g(n))$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$. That is, if n is big enough $f(n) \leq c \times g(n)$must hold, for the speed of growth of $g(n) > f(n)$.

Conisder $f(n) = O(g(n))$, There exists N such that for all $n \geq N$, $f(n) \leq c \times g(n)$, for c is a constant. Similarly, consider$f(n) = \Omega(g(n))$, There exists N such that for all $n \geq N$, $f(n) \geq c \times g(n)$, for c is a constant.

From definition above, we can conclude that if $f(n) = o(g(n))$, than $f(n) = O(g(n))$, and if $g(n) = o(f(n))$, than $f(n) = \Omega(g(n))$.

1.

$f(n) = \ln n, g(n) = \sqrt{n}$

$\lim_{n\to\infty} \frac{\ln n}{\sqrt{n}} \overset{\text{L'H}}{=} \lim_{n\to\infty} \frac{\frac{1}{n}}{\frac{1}{2\sqrt{n}}} = 0$

Then, $\ln n = O(\sqrt{n})$ is true.

2.

$f(n) = 928 \log_{830} n, g(n) = \ln n$

$\lim_{n\to\infty} \frac{928\log_{830}n}{\ln n} \overset{\text{L'H}}{=} \lim_{n\to\infty} \frac{\frac{928}{\ln 830}\frac{1}{n}}{\frac{1}{n}} = \frac{928}{\ln 830}$

That is, we can find a large c such that $\forall n$ is large, $f(n) \leq c \times g(n)$.

Similarly, we can find a small c such that $\forall n$ is large, $f(n) \geq c \times g(n)$.

Then, $f(n) = O(g(n)) = \Omega(g(n)) = \Theta(g(n))$, so $928\log_{830}n = \Theta(\ln n)$is true.

3.

$f(n) = \sum_{i=1}^{n} i^3 = [\frac{1}{2}(n \times (n-1))]^2, g(n) = n^4$

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{1}{4}$

That is, we can find a large c such that $\forall n$ is large, $f(n) \leq c \times g(n)$.

Similarly, we can find a small c such that $\forall n$ is large, $f(n) \geq c \times g(n)$.

Then, $f(n) = O(g(n)) = \Omega(g(n)) = \Theta(g(n))$, so $\sum_{i=1}^{n} i^3 = \Theta(n^4)$is true.

4.

$f(n) = n^{\pi}(\log n)^{112}, g(n) = n^3\sqrt{n}$

$\lim_{n\to\infty} \frac{n^{\pi}(\log n)^{112}}{n^3\sqrt{n}} = \lim_{n\to\infty} \frac{(\log n)^{112}}{n^{112}} \frac{1}{n^{3.5-\pi}} = 1 \times 0 = 0$

Then, $n^{\pi}(\log n)^{112} = O(n^3\sqrt{n})$ is true.

5.

$f(n) = \lfloor \log\log n \rfloor !, g(n) = n$

$\log(n!) = \log n + \log n - 1 + ... + \log 1 \leq n\log n$

$\log(\log\log n!) \leq \log\log n \times \log\log\log n \leq (\log\log n)^2$

$\lim_{n\to\infty} \frac{(\log\log n)^2}{\log n} \overset{\text{L'H}}{=} \lim_{n\to\infty} \frac{2(\ln\ln n)\frac{1}{\ln n}\frac{1}{n}}{\frac{1}{n}} \overset{\text{L'H}}{=} \lim_{n\to\infty} \frac{2\frac{1}{\ln n}\frac{1}{n}}{\frac{1}{n}} = 0$

By Squeeze Thm, $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$. Thus, $f(n) = O(g(n))$.

6.

(a)

$f(n) = \ln(n!), g(n) = n\ln n$

$\ln(n!) = \ln n + \ln(n-1) + ... + \ln 1 \leq n\ln n \leq c \times n\ln n$

By definition, $f(n) = O(g(n))$

$\ln(n!) = \ln[(\frac{n}{e})^n\sqrt{2\pi n}] \geq n\ln(\frac{n}{e}) \geq c \times n\ln n$, for n large.

By definition, $f(n) = \Omega(g(n))$

Thus, $f(n) = \Theta(g(n))$

(b)

$\ln(n!) - n\ln n + n = \ln(n! \times (\frac{e}{n})^n) \overset{\text{Stirling's formula}}{\sim} \ln[(\frac{n}{e})^n\sqrt{(2\pi n)} \times (\frac{e}{n})^n] \leq c \times \ln n$

By definition, $\ln(n!) - n\ln n + n = O(\ln n)$.

Similarly, $ln[(\frac{n}{e})^n\sqrt{(2\pi n)} \times (\frac{e}{n})^n] = \ln\sqrt{2\pi n} \geq c \times \ln n$

By definition, $\ln(n!) - n\ln n + n = \Omega(\ln n)$

Thus, $\ln(n!) - n\ln n + n = \Theta(\ln n)$

(b) Fill-in (9 points)

Please fill in the **numbers** according to their complexity in correct order.
Every pair $f(n) < g(n)$ in your answer means $f(n) = O(g(n))$.
You don't need to prove your answer in (b)!

- Points $= 9 - 0.6 \times \{(i,j) \mid i < j, answer[i] > answer[j]\}$
- **If your answer is not a permutation of [1,6], you would get 0 point from (b)!**

1. $\sqrt{n^{2.048}}$
2. $n\log(\lfloor n \rfloor!)$
3. 2^n
4. $n\log n$
5. n^k, for a large constant $k \to \infty,\ k \in \mathbb{R}^+$
6. $(\log n)^{1.0001^n}$

___ < ___ < ___ < ___ < ___ < ___

Fun fact: Every choice that is smaller than 5. is said to be "polynomially bounded".

***Ans:* 4. < 1. < 2. < 5. < 3. < 6.**

(c) Recursion (6 points)

In this section, we assume $T(1) = 1$.

Please give the tightest upper bound for each subproblem and justify your answer with a brief proof.

1. $T(n) = 26T\left(\frac{n}{9}\right) + n^{1.5}$ (3pt)
2. $T(n) = 3n + T\left(\frac{2n}{3}\right) + T\left(\frac{2n}{9}\right)$ (3pt)

1.

By the master theorem, $T(n) = O(n^{1.5})$

2.

1. The root of the recursion tree has value $3n$
2. The root has two children, with values $2n$, $\frac{2}{3}n$. The total value of all children is $\frac{8}{3}n$.
3. The root has four grandchildren, with values $\frac{4}{3}n$, $\frac{4}{9}n$, $\frac{4}{9}n$, $\frac{4}{27}n$. The total value of all grand children is $\frac{64}{27}n$.
4. The total value of k-th generation is $3 \times (\frac{8}{9})^k n$, which can be easily proved by induction.
5. $\text{The sum of values of all generation} = \lim_{m\to\infty} S(m) = \lim_{m\to\infty} \frac{3n[1-(\frac{8}{9})^m]}{1-\frac{8}{9}} = 27n$
6. We can conlucde that $T(n) = O(n)$.

Problem. 6

(a) (5 points)

For the following statements, you only need to answer **T**(True) or **F**(False). No proofs or reasons need to be provided.

Assume that r, s are **similar** strings with length k and $1 \leq i(\in \mathbb{N}) < k$. Define $r_1 = r[1..i], r_2 = r[(i+1)..k]$, and define s_1, s_2 accordingly.

1. If $r_1 = s_1$, then $r_2 = s_2$
2. If $r_1 \neq s_1$, then $r_2 = s_2$
3. If $r_1 = s_1$, then $r_2 \neq s_2$
4. If $r_1 = s_1$, then r_2, s_2 are **similar**
5. If $r_1 \neq s_1$, then r_2, s_2 are **similar**

1. False
2. True
3. False
4. True
5. True

(b) (5 points)

Note: according to policy, you need to prove the correctness and time complexity for (b).

Given n strings with length k, please design an $O(nk \log n)$ algorithm to divide them into groups, where

- each string in the same group are identical
- if two strings are in different group, then they are different.

You finally separated them and took some **gene** samples. For the following problems, you can assume that the n **gene**s are distinct.

We can group the string by sorting them by their ASCII code. Since each string that in the same group is next to each other, we can divide them into groups by this algorithm.

Divide

First, repeat Dividing a number of strings into half (like merge sort). It will take $\log n$ steps to do it, so the complexity of divide is $O(\log n)$. Then, we will get some sub arrays of strings that we should conquer.

Conquer

Assume that if s_1, s_2 is two strings. If $s_1[0...i-1] = s_2[0...i-1]$ and $s_1[i] < s_2[i]$(according to ASCII code), then s_1 is considered **bigger** than s_2. Sort the sub arrays of strings by this rule.

Combine

Now we need to combine sub arrays of strings into a output vector by compare their bigness. The complexity of comparing two strings is $O(k)$ since we need to run through all the indices of strings. Also, every strings are needed to be compared 0~1 time every round, so the complexity of comparing all strings is $O(n)$.Thus, the total complexity in every round is $O(nk)$.

Output

In the end, we first open a 2D vector. We check through the vector $O(n)$. If the string is new, then push back to the vector. Otherwise, add 1 to the index. Since we sort the vector by the method above, the indentical strings are next to each other. Thus, we just need to check the string next to the previous one every time. The complexity is $O(nk)$.

As a result, this algoritm have an $O(nk \log n)$ complexity.

(c) (5 points)

Design a **divide-and-conquer** algorithm to solve the **final task**. No correctness or complexity analysis needed. (Hint: **Problem (a) & (b)**)

This algorithm needs to run in $O(nk \log n \log k)$, although you don't need to prove it in this subproblem. Also, we recommend you to explain the algorithm in words. If you want to answer it in pseudo code, make sure it is correct, readable (comment or explain if needed), and **less than 35 lines**.

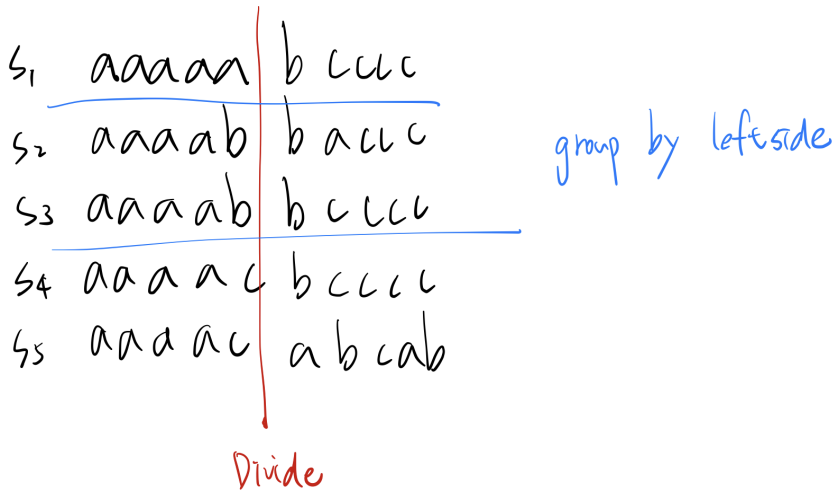
From Problem (a), if two strings r, s are considered similar, than if we divide the strings into half, at most one of the sub-strings have at most one index i s.t. $r_{sub}[i] \neq s_{sub}[i]$.

From Problem (b), we can group the n strings first $O(nk \log n)$, and then we have a vector with unique strings and the number of each string..

Let s_i be i-th string (unique), and n_i be the numbers of each string.

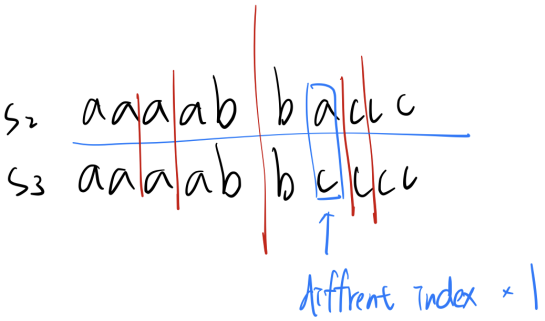
Divide

First, Divide the strings into half, and then group them by the substrings on the left side according to the problem (b) method ($O(n \log nk)$). For example,

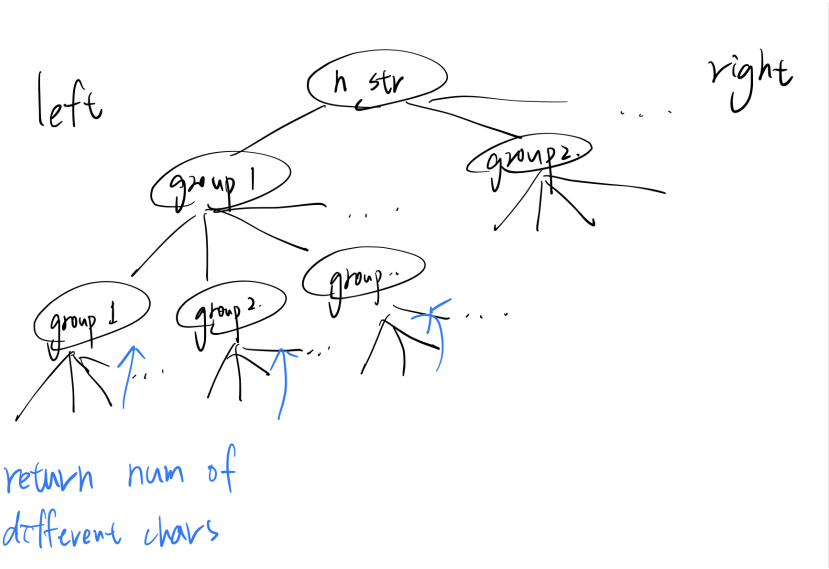


Also, if we've done cuting the right hand side, then we cut left hand side.

Then, Eventually when we cut them into some substrings with length of 1 or 2, we can compare them and return the number of different characters in the substrings to the parents node. Take s_1 and s_2 for example.



If a group have more than 1 different character in the group, then they won't be all similar. However, if they have less than 1 different character, the group are all similar.



The divide complexity will be $O(\log k)$, the total complexity of grouping and dividing is $O(nk \log n \log k)$.

Conquer

Now we know how to divide the strings. The way to conquer it is that according to we know every group is similar or not. If it's similar, we can get the number of similar pairs, which is $C_2^{\text{num of group's menber}}$ and then add to the answer. If the children nodes of this group are also similar, then we ignore them and just consider the parents groups. The complexity of it is $O(1)$.

Combine

Since each group will return children's the number of different characters, we can get the parents' number of different characters, which is the total of children's the number of different characters. The complexity is $O(1)$.

In the end, we will know all the group nodes that are similar. We can compute the answer by adding up their $C_2^{\text{num of group's menber}}$. Finally, we get the answer.

Thus, this algoritm have an $O(nk \log n \log k)$ complexity.

(d) (3 points)

Prove the correctness for **(c)**.

However, since you are very busy, you are not sure about how much time it will take. Therefore, you decide to analyze it first.

Base case

Suppose that we have 2 strings r and s whose length is 2. Divide them and then we got two groups(let them be g_1 and g_2), which both contain two strings with length of 1.

If g_1 is similar and g_2 is identical, then they are similar.

If g_1 is indentical and g_2 is similar, then they are similar.

If $g_1 g_2$ both indentical, then they are similar.

If both $g_1 g_2$ have ≥ 1 different character, then they are not similar.

Inductive step

Suppose that we have n strings. s_i deonte i-th string. Add another string s_{n+1} to the strings set, which is also similar to s_n. That is, s_{n+1} have at most 1 index different to s_k for some $k \leq n$, let it be t if exist. Now, when we divide the strings set into half, the s_{n+1}, s_k will eventually be in a same group because they are similar to each other. In this case, these groups will thus have $C_2^{\text{num of group menber }+1}$ pairs. Thus, the number of k that satisfying the assumption will be added to the answer.

(e) (2 points)

Let $f(x) = x \log x$, and $a, b > 1$, prove that:

$$f(a) + f(b) \leq f(a+b)$$

$$f(a) + f(b) \leq f(a+b)$$
$$a \log a + b \log b \leq (a+b)\log(a+b) = a\log(a+b) + b\log(a+b)$$
$$\because a \log a \leq a \log(a+b) \text{ and } b \log b \leq b \log(a+b) \text{ obviously true.}$$
$$\text{Thus, } f(a) + f(b) \leq f(a+b) \text{ is true.}$$

(f) (5 points)

Prove that the algorithm in **(c)** runs in $O(nk \log n \log k)$.

Let $T(n)$ denote the running time of this algorithm.

$$T(n,k) = \begin{cases} O(1), \text{if } k = 1, \text{otherwise} \\ 2T(n, \frac{k}{2}) + O(nk \log n) \end{cases}$$

Use induction to prove $T(n,k) < c_1 \cdot nk \log n \log k - c_2 nk \log n$

Base case

$n = 2, k = 2$

$$\begin{aligned} T(2,2) &= 2T(2,1) + O(4 \log 2) \\ &\leq O(1) + c \cdot 4 \log 2 \\ &\leq c \cdot 4 \log^2 2 \end{aligned}$$

Inductive step

$$\begin{aligned} T(n,k) &= 2T(n, \tfrac{k}{2}) + O(nk \log k) \\ &\leq 2\left[c_1 n \tfrac{k}{2} \log n \log \tfrac{k}{2} - c_2 n \tfrac{k}{2} \log n\right) + c_3 nk \log n \\ &\leq c_1 nk \log n \log \tfrac{k}{2} - c_2 nk \log n + c_3 nk \log n \\ &\leq c_1 n \cdot k \cdot \log n \log k \end{aligned}$$

(Choose $c_2 \geq c_3$ & $\log k > \log \frac{k}{2}$)

Thus, $T(n,k) = O(n \cdot k \cdot \log n \log k)$

Thus, $T(n,k) = O(nk \log k \log n)$

(g) (bonus 5 points)

Prove that the algorithm in **(c)** runs in $O(nk \log n)$.

$$T(n,k) = \begin{cases} O(1), \text{if } k = 1, \text{otherwise} \\ 2T(n, \frac{k}{2}) + O(nk \log n) \end{cases}$$