COMP2119C In-class Exercise (4 questions)

Credits: Special thanks to Prof. Hubert Chan (COMP2119A) for sharing the questions.

2. (20 pts) True or False? No proofs are needed.

   (1) $n^{1.5}$ is $O(n \log n)$.
   (2) $2^{\sqrt{\log n}}$ is $O(n)$.
   (3) $n^{1.01} \cdot n^{0.99}$ is $o(n^2)$.
   (4) If $k \log k = \Theta(n)$ then $k = \Theta(\frac{n}{\log n})$.
   (5) If $f(n)$ is $O(n)$ and $g(n)$ is $o(n^2)$, then $\max(f(n), g(n))$ is $O(n^3)$.
   (6) If $f(n) + g(n)$ is $\Theta(g(n))$, then $f(n)$ is $\Theta(g(n))$.
   (7) If $f(n)$ is $O(n)$ and $g(n)$ is $\Omega(n)$, then $|g(n) - f(n)|$ is $\Omega(n)$.
   (8) If $T(n) = \begin{cases} 1, & n \leq 2; \\ T(n-1) + T(n-2), & n \geq 3, \end{cases}$ then $T(n)$ is $\Omega(n)$.
   (9) If $T(n) = \begin{cases} 5, & n = 1; \\ 4T(\lfloor \frac{n}{2} \rfloor) + \sqrt{n} \log n, & n \geq 2, \end{cases}$ then $T(n)$ is $O(\sqrt{n} \log n)$.
   (10) If $T(n) = \begin{cases} 10, & n \leq 139; \\ T(\lfloor \frac{1}{5} n \rfloor) + T(\lfloor \frac{7}{10} n \rfloor) + 10n, & n \geq 140, \end{cases}$ then $T(n)$ is $\Theta(n)$.

4. (15 pts) Answer the following questions about hashing.

   (1) (4 pts) If we have $60,000$ items to store in a hash table **H-Open** using open addressing with linear probing and we desire a load factor of 0.4, how many slots the hash table should contain?

   (2) (4 pts) If we have $60,000$ items to store in a hash table **H-Chaining** using chaining and we desire the expected number of comparisons in a successful search to be 3.5, how many slots the hash table should contain?

   (3) (7 pts) Analyse the memory usage for **H-Open** and **H-Chaining**. Suppose item's size is 10 bytes and pointer's size is 4 bytes.

5. (20 pts) There is an array `arr` of N integers (indexed from 0), and we are supposed to perform queries on `arr`. For each query, two indices l and r are given such that $0 \leq l \leq r < N$, which represents an interval of the array. You have to determine the index x such that $l \leq x \leq r$ and `arr`$[x]$ is the largest among `arr`$[l \dots r]$. If there are multiple indices x that satisfy the condition, return the smallest x among them.

   (1) (4 pts) Suppose the given array is $\{9, 5, 9, 3, 4, 2, 9\}$ with $N = 7$, determine the answer to the following queries:

   (i) $l = 0$ and $r = 6$;

   (ii) $l = 1$ and $r = 2$;

   (iii) $l = 3$ and $r = 3$;

   (iv) $l = 3$ and $r = 5$;

(2) (16 pts) Read the following `C++` code. Based on array `arr`, `pre_processing()` is executed once to determine an array named `pre_calc`, which is later used to handle all the queries. After that, whenever a query with l and r is given, `query(l,r)` is called to determine the answer. Fill in the blank so that the code correctly answers each query. (`Log2(x)` returns $\lfloor \log_2 x \rfloor$; `Pow2(x)` returns 2^x.)

```
int arr[N];   // Input array of size N
int pre_calc[N][Log2(N-1)+1];
// pre_calc[i][j] stores the answer for arr[i,...,i+Pow2(j)-1]

void pre_processing()
{
    for(int i = 0; i < N; ++i)
        pre_calc[i][0] = i;
    for(int idx = 1; ______(a)______ < N; ++idx)
        for(int i = 0; i < N; ++i)
            if(______(b)______){
                pre_calc[i][idx] = pre_calc[i+Pow2(idx-1)][idx-1];
            }else{
                pre_calc[i][idx] = pre_calc[i][idx-1];
            }
}


int query(int l, int r){
    int len = r - l;
    int llen = Log2(len);
    int blen = Pow2(llen);
    if(______(c)______){
        return pre_calc[l][llen];
    }else{
        return ______(d)______;
    }
}
```

6. (25 pts) In a video game, there are n rooms in a row, indexed from left to right as $0, 1, \ldots, n-1$. Each room, denoted by i, is assigned a competence value $c[i] > 0$.

   At the beginning of the game, the player is randomly assigned to one of the rooms and their competence level is set to the competence value of that room. The player can explore the remaining rooms, moving freely between consecutive rooms as long as their competence level is not smaller than the competence value of that room.

   Let $l[i]$ and $r[i]$ denote the leftmost and rightmost room, respectively, that a player can travel to if they start at the i-th room.

   (1) (5 pts) Suppose that there is a valuable treasure with value $v[i] > 0$ in each room. A player can collect the treasure in a room once they enter it. Let $t[i]$ denote the total value of treasures that a player can collect if they start in the i-th room. (Each treasure can be collected only once.)

   Suppose $l[0, \ldots, n-1]$ and $r[0, \ldots, n-1]$ are given. Provide an implementation (in pseudocode) to calculate $t[0, \ldots, n-1]$ in $O(n)$ time with at most $O(n)$ extra space.

   (2) (20 pts) Design an efficient algorithm to calculate $l[0, \ldots, n-1]$. What is the time and space complexity of your algorithm? Justify your answer.