Introduction to Algorithm
Handwritting 1

2024.09.17 - 2024.10.01

Kai-Chiang Wu

- **The problem numbers with '*' are the ones we think are tricky. Try your best on those.**
- **Please compile all your write-ups (photos or scanned copies are acceptable; ensure that the electronic files are clear and easy to read) into a single PDF file, and submit it via E3.**
- **Name this PDF file as {Your-student-ID}_hand1.pdf. For example, 123456789_hand1.pdf.**

1. (6 points) Prove that $n^2 + n \log n + 3 \in O(n^2)$
2. (6 points) Prove or disprove that $f(n) \in O(n^n) \iff f(n) \in O\Big((n+1)^{(n+1)}\Big)$?
3. ($3 \times 3 = 9$ points) Please write down the process of *Insertion Sort*, *Selection Sort*, and *Merge Sort* on the array $\{1, 4, 2, 8, 5, 7, 6, 3\}$
4. Use the Master Theorem to give the tight asymptotic bounds.
   (a). (3 points) $T(n) = 2T(n/2) + O(1)$
   (b). (3 points) $T(n) = 2T(n/4) + O(n^2)$
   (c). (3 points) $T(n) = 3T(n/\sqrt{2}) + O(n^4)$

5. There are 2 algorithms for counting the factors.

```c
int Count_factors(int n){
    int ans=0;
    for (int i=1;i<=n;i++){
        for (int j=1;j*j<=i;j++){
            if (i%j==0) ans+=2;
            if (j*j==i) ans--;
        }
    }
    return ans;
}
```

Listing 1: Algorithm *simp*

```c
int Count_factors(int n){
    int ans=0;
    for (int i=1;i<=n;i++){
        for (int j=1;j<=n/i;j++){
            ans++;
        }
    }
    return ans;
}
```

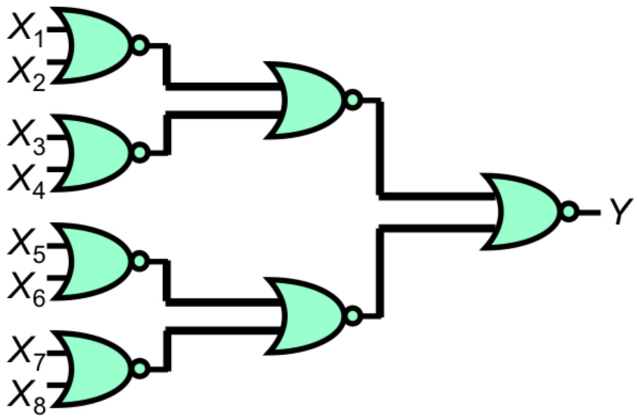
Listing 2: Algorithm *har*

(a). ($2\times 2 = 4$ points) What are the time complexities of *simp* and *har*, respectively? Please briefly explain your answer.

(b). (3 points) Let $f(n)$ be the number of the factors of n. For example, $f(4) = 3$, $f(10) = 4$. Please show that $f(n) \in O(n^{\frac{1}{3}})$.

(c*). (4 points)Do you think $\sum_{k=1}^{n} k^{\frac{1}{3}} \in O(n \log n)$ hold? You have to explain your answer.

6. Since this course is "***Introduction to Algorithm***", let's introduce a randomized algorithm!
   Here is an implementation of a NOR circuit:



You can see that this implementation looks like a full binary tree with 2^n inputs. Each input x_i is either True or False. Our target is to get the output value y.

(a). (3 points) Please design an algorithm with time complexity $O(2^n)$ to solve the problem. Make sure that your description is read-friendly.

Now, here is a randomized algorithm. Let's call it *Cool*!

Algorithm 1 Cool

```
Boolean COOL(x_1, x_2, ..., x_{2^n})
if 2^n==1 then
    return x_1
end if
Throw a fair coin.
if Show Head then
    return (COOL(x_1, ..., x_{2^{n-1}})==1)? 0 : !COOL(x_{2^{n-1}+1}, ..., x_{2^n})
else
    return (COOL(x_{2^{n-1}+1}, ..., x_{2^n})==1)? 0 : !COOL(x_1, ..., x_{2^{n-1}})
end if
```

(b*). (6 points) Please show that **the** *Cool* **algorithm is correct** and prove **the expected time complexity is** $O(3^{\frac{n}{2}})$ when $n \geq 2$. (Hint: For correctness, observing the truth table of NOR operation; For complexity, trying to use mathematical induction.)

After you answer the question. It's easy to imply that the time complexity with n inputs is $O(n^{0.793})$.