# CSCI 670: Advanced Analysis of Algorithms
# $2^{\text{nd}}$ Assignment

August 2021

**Problem 1.** SAT problem is a decision problem asking whether the given logical formula is satisfiable by any boolean assignment or not. In general, SAT is NP-complete. A CNF (conjunctive normal form) is a logical expression consisting of conjunctions of clauses where each clause is a disjunction of literals. A literal denotes a variable or its negation. The following formula is an example of CNF.

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_5) \wedge (x_2 \vee x_4 \vee x_5)$$

We call a formula is $k$-CNF if each clause contains exactly $k$ literals. Prove that for any $k$-CNF formula with $m$ clauses, there is a boolean assignment such that it satisfies at least $m(1-2^{-k})$ of clauses.

**Solution 1.** Assign uniformly uniformly random boolean assignment, i.e. set each variable to true with probability $1/2$ independently. Let $X_i$ be the random variable denoting the $i$-th clause is satisfies. Then,

$$\mathbb{E}\left[\sum_{i=1}^{m} X_i\right] = \sum_{i=1}^{m} \mathbb{E}[X_i] = m(1-2^{-k}).$$

By probabilistic method, there exists a boolean assignment satisfying at least $m(1-2^{-k})$. Otherwise, expectation can not have this value.

**Problem 2** (From Fall 2019)**.** Recall the algorithm for finding the median of a sequence in linear time. The algorithm implements a recursive function that finds the k-th order statistics.

**Algorithm 1** ORDER-STATISTICS($[a_1; \ldots; a_n], k$)

```
if n ≤ 5 then
    sort a in constant time and return the k-th element
else
    divide the array a into groups of size 5
    compute the median of each group, denote the median of the i-th group by mi
    m ← ORDER-STATISTICS([m1; ...; m_{n/5}]; n/10).
    A ← {a_i : a_i ≤ m}
    B ← {a_i : a_i > m}
    if |A| ≥ k then
        return ORDER-STATISTICS(A; k)
    else
        return ORDER-STATISTICS(B; k − |A|)
    end if
end if
```

Since $m$ is the median of medians there are $n/10$ medians smaller than $m$. For each of these medians there are two elements in their group that are smaller than them. Therefore, in total there are at least $3n/10$ elements smaller $m$ (i.e. $|A| \geq 3n/10$). The same way we can see that $|B| \geq 3n/10$. Therefore, the worst case complexity of our algorithm for an array of size $n$ can be written as:

$$T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + O(n)$$

Prove that $T(n) = O(n)$.

**Solution 2.** We use induction to prove that there exists $C > 0$ such that for each $n$ we have $T(n) \leq Cn$. First of all, the $O(n)$ term in the complexity can be bounded something like $30n$. Now let us take $C = 300$. In the base case $n = 1$ we have $T(n) = 1 \leq 300 \cdot 1$. Assuming that $T(k) \leq Ck$ for all $k < n$, let us prove that $T(n) \leq Cn$ :

$$\begin{aligned} T(n) &= T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + O(n) \\ &\leq C\frac{n}{5} + C\frac{7n}{10} + 30n \\ &\leq \left(\frac{9C}{10} + 30\right) n \\ &= \left(\frac{9C}{10} + \frac{C}{10}\right) n \\ &= Cn. \end{aligned}$$

**Problem 3** (From Fall 2017, Median of tree)**.** Prove that in every tree with $n$ nodes, there exists a node $v$ such that if $v$ is removed from the tree, each connected component has at most $\lfloor n/2 \rfloor$ nodes.

**Solution 3.** Here is the procedure to find the median of a tree. We start from an arbitrary vertex $v$ and iteratively move one of its neighbors until we find a median. For any neighbor $u$ of $v$, we define $C_{v\to u}$ as the vertices of the connected component containing $u$ when $v$ is removed and $D_{v\to u}$ as its cardinality. At each step, we choose the vertex $u$ such that $D_{v\to u} > n/2$. If such a vertex $u$ does not exist, then $v$ will be a median of the input tree.

Observe that at most one neighbor $u$ of $v$ has $D_{v\to u} > n/2$ since there can't be two distinct connected components each having size of at least $n/2$.

It remains to prove that procedure stops at a valid median. To prove this claim, we will use a monotonicity argument. Let's define $D^*_v = \max\{D_{v\to u} \ : \ (u, v) \in E\}$ and let $u$ be the vertex visited after $v$. We claim that $D^*_v > D^*_u$. Observe that $D^*_v = D_{v\to u}$ since $D_{v\to u} \geq n/2$. Then observe that for any neighbor $y$ of $u$ such that $y \neq v$, then observe that $C_{u\to y} \subsetneq C_v \to u$ and so $D_{u\to y} < C_v \to u$. Moreover, $D_{u\to v} = n - D_{v\to u}$ which is always less than $D_{v\to u} > n/2$.

How do you implement this procedure in linear time?

**Problem 4.** Let $T = (V, E)$ be a tree. Design an $O(n \log n)$ algorithm to find the number of length $k$ paths in the tree $T$.
Hint: You may use Problem 3.

**Solution 4.** We can design a divide and conquer algorithm for this problem.

Assume for a moment we can count the number of length $k$ paths passing through an arbitrary vertex $v$ efficiently. Then, consider the graph once we remove the vertex $v$. Notice that all remaining paths we want to count lie in smaller connected components. Then, we can count them by using the same procedure recursively. Instead of choosing an arbitrary vertex $v$, if we choose the median of the tree, then we ensure that subproblems have a size less than half of the original one. So, it remains to design a way to count the number of paths

passing through $v$ efficiently. Notice that we can count them by computing the number of paths starting from $v$ for each length $\ell \leq k$ (How?).

**Algorithm 2** Count length $k$ paths

1: Find the median of the tree, say $v$.
2: Consider the connected components (subtrees) obtained by removing $v$.
3: Count the length $k$ paths in each of the connected components.
4: Count the length $k$ paths passing through vertex $v$.

Analyze the runtime. Notice that depth of the recursion takes $O(\log n)$ steps at most. In each level of recursion algorithm performs linear number of operations.

**Problem 5.** Assume you are at the top of a building with $n$ floors. You have a few amount of identical eggs and you want to find the highest floor $t$ such that if you drop one of the eggs from there but it doesn't break. You can experiment whether an egg gets broken by testing from any floor. Of course you can not use a broken egg anymore. Be careful!

(a) Assuming that you have single egg, design an algorithm that finds answer with linear amount of experiments. Prove that your algorithm is the only one that works.

(b) Can you find the answer faster if you have k eggs instead of one? Design an algorithm to solve the problem using $O(k \cdot n^{1/k})$ number of experiment in the worst case.

**Solution 5.** (a) You have to check the floors one-by-one: Drop the egg from the first floor. If it doesn't break, go to the second floor; and so on and so forth.

To prove that it's the only algorithm that works, suppose that you drop the only egg from floor $f+1$ while you've not checked floor $f$ yet. If the egg breaks, you can't figure out what would happen if you dropped it from floor $f$.

(b) Let $m$ be an integer less than $n$. Drop the first egg from floors $i \cdot m$ for $1 \leq i \leq \lfloor n/m \rfloor$. If it breaks at floor $j \cdot m$, you know that $(j-1)m \leq t < jm$. If it doesn't break even from floor $\lfloor n/m \rfloor m$, then $\lfloor n/m \rfloor m \leq t \leq n$. In either case, at most $m$ consecutive floors remain as candidates.

Assuming that $T(n,k)$ is the optimal answer (for $n$ floors $k$ eggs), we have the following: $T(n,k) \leq n/m + T(m, k-1)$. For $m = n^{(k-1)/k}$,

$$T(n,k) \leq n^{1/k} + T\left(n^{(k-1)/k}, k-1\right)$$

By induction on $k$, it's easy to show that $T(n,k) \in O\left(k \cdot n^{1/k}\right)$.

**Problem 6.** Let $a_1, \ldots a_n \in \mathbb{R}$ be a sequence of reals. We say a pair of indices $(i,j)$ is an inversion if $i < j$ and $a_i > a_j$. Design a $O(n \log n)$ algorithm to find number of inversions of given sequence $a$.

**Solution 6.** Consider the following algorithm.

**Algorithm 3** Count Inversions

1: Split the sequence into two subsequences from the middle.
2: Recursively count number of inversions in both subsequence seperately.
3: Count the number of inversions across two subsequences.

Notice that the algorithm is very similar to Merge Sort Algorithm. The analysis of the algorithm is left to the reader. One should notice that in order to achieve $O(n \log n)$ runtime, Step 3 of the algorithm must be implemented in the linear time (How?).

**Problem 7** (From Fall 2019)**.** Consider two subsets $A$ and $B$ of $\{1, 2, \ldots, n\}$. The direct sum of A and B is defined as $A + B = \{a + b \ : \ a \in A, b \in B\}$. Design an $O(n \log n)$ algorithm to find the direct sum of $A$ and $B$.

**Solution 7.** Let us represent $A$ and $B$ as binary arrays $a_i$ and $b_i (i = 1..n)$ correspondingly, with $a_i = 1$ iff $i \in A$ and $b_i = 1$ iff $i \in B$. Consider the polynomials $a(x) = a_1 x + a_2 x^2 + \cdots + a_n x^n$ and $b(x) = b_1 x + b_2 x^2 + \cdots + b_n x^n$. Let $c(x) = c_1 x + c_2 x^2 + \cdots + c_{2n} x^{2n}$ be the product of $a(x)$ and $b(x)$. It is easy to see that $c_i \geq 1$ if and only if $i \in A + B$. Therefore, to find $A + B$ we just need to find the nonzero coefficients of $c(x)$. Computing the coefficients of $c(x)$ can be done in $O(n \log n)$ time using the discrete FFT algorithm.

**Problem 8.** We say that a collection of sets $H$ shatters a set $C$ if $H \cap C := \{h \cap C \ : \ h \in H\}$ equals to the power set of $C$. The VC dimension of the collection $H$ is the cardinality of the largest set $C$ that can be shattered by $H$. Let $H$ be subsets of the real line $\mathbb{R}$ formed by the union of $k$ intervals.

(a) Determine the VC dimension of $H$ when $k = 2$.

(b) Determine the VC dimension of $H$ for arbitrary $k > 2$.

**Solution 8.** (a) One can see that sets that consist of a union of 2 intervals can shatter any 4 points (How?). We will discuss that they can not shatter 5 points. Let $C = \{1, 3, 5, 7, 9\}$ be 5 points on the real axis. We claim that for $k = 2$, $H \cap C \not\supseteq \{1, 5, 9\}$. Assume for a contradiction for let $h \in H$ be a set such that $h \cap C = \{1, 5, 9\}$. We know that $h$ consists of two intervals. So either, 1 and 5 are in the same interval or 5 and 9. Then, either 3 belongs to intersection or 7. This gives us a contradiction. Therefore, VC-dimension of $H$ is 4 when $k = 2$.

(b) Similarly, prove that VC-dimension of $H$ is $2 \cdot k$ for any $k$.