CSC165H1 Problem Set 4

Wei CUI, Mengyao Li, Xingkun Yin

April 5, 2018

1 Binary representation and algorithm analysis

(a) **Proof:** Let $n \in \mathbb{N}$, and consider the running time of algorithm on an input integer n.

The initialization lines before the for loop take 1 step (i.e., have runtime independent of the input size).
For a fixed iteration of the outer loop, the inner loop at Line 13 iterates at most $\lfloor log_2(n) \rfloor + 1$ times, with each iteration costing a single step, the outer loop at Line 9 iterates n times with each iteration taking at most $\lfloor log_2(n) \rfloor + 1$ steps. Therefore, the total runtime of this algorithm is at most $1 + n(\lfloor log_2(n) \rfloor + 1)$, which is $\mathcal{O}(nlogn)$

(b) **Proof:** Let $n \in \mathbb{N}$, and consider the running time of algorithm on an input integer n.

The initialization lines before the for loop take 1 step (i.e., have runtime independent of the input size).
For a fixed iteration of the outer loop, the inner loop at Line 13 iterates at least 0 times. The outer loop at Line 9 iterates n times, with each iteration taking at least a single step. Therefore, the total runtime of this algorithm is at least $1 + n$, which is $\Omega(n)$

2 Worst-case and best-case algorithm analysis

(a) **Proof:** WTP: $\forall n \in \mathbb{N}, \forall L \in \mathcal{I}_{func,L}$, running time of executing $f(L) \leq n$
Let $n \in \mathbb{N}$, let $L \in \mathcal{I}_{func,L}$ and consider the running time of this algorithm on an input list L of length n.
The initialization lines before the while loop take 1 step (i.e., have runtime independent of the input size).
The while loop at Line 5 terminates when $i \leq 0$ and has two options in it, if $L[i]$ % 2 == 0 is always true, then the loop will iterate $\lceil log_2(n-1) \rceil$ times, since i is decreased by half of its previous value; if $L[i]$ % 2 == 0 is always false, then else block executes. Since i decreases

x each iteration, x is initialized with 1 and its value doesn't change during the while loop, thus the loop will execute $n-1$ times.
Therefore, by comparing these two options, the while loop iterates at most $n-1$ times, with each iteration taking a single step.
As a result, the total runtime of executing $f(L)$ is at most n, which is $\mathcal{O}(n)$

(b) **Proof:** WTP: $\forall n \in \mathbb{N}$, $\exists L \in \mathcal{I}_{func,L}$, running time of executing $f(L) \geq n$
Let $n \in \mathbb{N}$, pick a list L with the following property: $L[i] = 1$, for each $i \in \{0, 1, 2, ..., n-1\}$.
The initialization lines before the while loop take 1 step (i.e., have runtime independent of the input size).
Since all the elements in list L is 1, which is odd, therefore $L[i]$ % 2 == 0 is always false. Then the else block executes. Since the while loop terminates when $i \leq 0$, and i decreases x each iteration, x is initialized with 1 and its value doesn't change during the while loop, thus the loop will execute $n-1$ times, with each execution taking a single step. As a result, it takes a total of n steps, which is $\Omega(n)$

(c) **Proof:** WTP: $\forall n \in \mathbb{N}$, $\exists L \in \mathcal{I}_{func,L}$, running time of executing $f(L) \leq log(n)$
Let $n \in \mathbb{N}$, pick a list L with the following property: $L[i] = 2$, for each $i \in \{0, 1, 2, ..., n-1\}$.
The initialization lines before the while loop take 1 step (i.e., have runtime independent of the input size).
Since all the elements in list L is 2, which is even, therefore $L[i]$ % 2 == 0 is always true.Then the loop will always execute if block. Since by each iteration, i will be divided by 2 until i is less than or equal to 0. As a result, the loop will iterate $\lceil log_2(n-1) \rceil$ times, with each iteration taking a single step. Therefore, the total runtime of executing $f(L)$ is $1 + \lceil log_2(n-1) \rceil$, which is $\mathcal{O}(logn)$

(d) **Proof:** WTP: $\forall n \in \mathbb{N}$, $\forall L \in \mathcal{I}_{func,L}$, running time of executing $f(L) \geq log(n)$
Let $n \in \mathbb{N}$, let $L \in \mathcal{I}_{func,L}$ and consider the running time of this algorithm on an input list L of length n.

In this function, i is initially assigned with the value $n-1$ which is a relative large number and x initially equals to 1. Therefore for the best-case runtime of function, we want to run the if branch at Line 6 first. If the else statement at Line 9 is executed first, the function makes i minus x in each iteration, and x is initially equals to 1. In contrast, if the if statement is executed first, the variable i is divided by 2 and the value of x increases by 1. Dividing a large i by 2 always decreases the value of i by at least 1. Therefore, executing the else statement first is far less efficient for the first few iterations. However, after a few iterations, the value of x will be big enough, which makes running the else statement faster. As the result, for the best-case runtime of this function, we want the function to run the if branch

at Line 6 first. When the variable x is big enough, enter the else block at Line 9. Next, we will generalize this case and show that its runtime is $\Omega(logn)$.

Let $a \in \mathbb{N}$ be the number of times of executing the if statement at Line 6 during the while loop. Then $0 \leq a \leq \lceil log_2(n-1) \rceil$(which is the number of iterations we only run the if statement), also a must be greater than 0. Let $b \in \mathcal{Q}$, $0 \leq b \leq 1$ and $a = b \lceil log_2(n-1) \rceil$.

After executing a times of the if statement at Line 6 during the while loop, the variable x equals to $a+1$; and the value of the variable i will be $\left\lfloor \frac{n-1}{2^a} \right\rfloor$. The runtime of executing all the if statement during the while loop is a since executing the if statement in each iteration of the while loop takes constant time(a single step).

Next, we will enter the else block at Line 9. Since the while loop terminates when $i \leq 0$, therefore the function will execute $\left\lceil \frac{\left\lfloor \frac{n-1}{2^a} \right\rfloor}{a+1} \right\rceil$ times of the else statement. Since, in each iteration of the while loop, executing the else block costs a single step, therefore the runtime of executing all the else statement during the while loop is $\left\lceil \frac{\left\lfloor \frac{n-1}{2^a} \right\rfloor}{a+1} \right\rceil$.

The initialization lines before the while loop take 1 step(i.e., have runtime independent of the input size). As a result, the total running time of executing $f(L)$ is at least $\left\lceil \frac{\left\lfloor \frac{n-1}{2^a} \right\rfloor}{a+1} \right\rceil + a + 1$.

$$
\begin{aligned}
\left\lceil \frac{\left\lfloor \frac{n-1}{2^a} \right\rfloor}{a+1} \right\rceil + a + 1 &\geq \frac{\left\lfloor \frac{n-1}{2^a} \right\rfloor}{a+1} + a + 1 && \text{(by the definition of the ceiling function)} \\
&= \frac{\left\lfloor \frac{n-1}{2^{b \cdot \lceil log_2(n-1) \rceil}} \right\rfloor}{1 + b \lceil log_2(n-1) \rceil} + 1 + b \lceil log_2(n-1) \rceil && \text{(since } a = b \cdot \lceil log_2(n-1) \rceil\text{)} \\
&\geq \frac{\left\lfloor \frac{n-1}{2^{b[1+log_2(n-1)]}} \right\rfloor}{1 + b[1 + log_2(n-1)]} + 1 + b \cdot \lceil log_2(n-1) \rceil \\
&= \frac{\left\lfloor \frac{(n-1)^{1-b}}{2^b} \right\rfloor}{1 + b + b \cdot log_2(n-1)} + 1 + b \cdot \lceil log_2(n-1) \rceil \\
&\geq \frac{\frac{(n-1)^{1-b}}{2^b} - 1}{1 + b + b \cdot log_2(n-1)} + 1 + b \cdot \lceil log_2(n-1) \rceil \\
&= \frac{(n-1)^{1-b} - 2^b}{2^b + b \cdot 2^b + 2^b \cdot b \cdot log_2(n-1)} + 1 + b \cdot \lceil log_2(n-1) \rceil \\
&\in \Omega(logn) && \text{(since } b \text{ is a constant)}
\end{aligned}
$$

Therefore, $\forall n \in \mathbb{N}$, $\forall L \in \mathcal{I}_{func,L}$, running time of executing $f(L) \geq log(n)$
Therefore, $BC(n) \in \Omega(logn)$

3 Graph algorithm

(a) **Proof:** WTP: $WC(n) \in \Theta(2^n)$, which is: $\forall n \in \mathbb{N}$, $\forall M \in \mathcal{I}_{func,M}$, running time of executing $f(M) \leq 2^n$ and $\forall n \in \mathbb{N}$, $\exists M \in \mathcal{I}_{func,M}$, running time of executing $f(M) \geq 2^n$

First, we will prove that $WC(n) \in \mathcal{O}(2^n)$, which is: $\forall n \in \mathbb{N}$, $\forall M \in \mathcal{I}_{func,M}$, running time of executing $f(M) \leq 2^n$
Let $n \in \mathbb{N}$, let $M \in \mathcal{I}_{func,M}$ and consider the running time of this algorithm on an input adjacency matrix M of n by n.
The initialization lines before the for loop take 1 step (i.e., have runtime independent of the input size).
For a fixed iteration of the outer loop, the inner loop at Line 7 iterates n times with each iteration costing a single step. If $count == 0$ is true only when $i = n - 1$, the outer loop at Line 5 will iterate at most n times, with each iteration taking n steps, therefore the total steps of the loop 1 and loop 2 is at most n^2. Now let's consider the loop 3 at Line 14, when $i = n - 1$, the variable $found\_isolated$ is true, the if block at Line 13 will execute and the loop 3 at Line 14 will iterate 2^n times, with each iteration costing a single step.
As a result, the total runtime of executing $f(M)$ is at most $1 + n^2 + 2^n$, which is $\mathcal{O}(2^n)$

Next, we will prove that $WC(n) \in \Omega(2^n)$, which is: $\forall n \in \mathbb{N}$, $\exists M \in \mathcal{I}_{func,M}$, running time of executing $f(M) \geq 2^n$
Let $n \in \mathbb{N}$, and pick a adjacency matrix M with the following property: $M[i][j] = 0$, for each $i, j \in \{0, 1, 2, ..., n - 1\}$
The initialization lines before the for loop take 1 step (i.e., have runtime independent of the input size).
When $i = 0$, the inner loop at Line 7 will execute n times, with each execution taking a single step, after executing the second loop at Line 7, the variable *count* will equal to 0, since $M[0][j] = 0$, for each $j \in \{0, 1, 2, ..., n - 1\}$. Then the variable $found\_isolated$ will be assigned True and loop 1 at Line 5 will terminate after the first iteration because of the break statement at Line 11. Therefore, the total cost of loop 1 and loop 2 is n steps.
Since the variable $found\_isolated$ is True, then the if block at Line 13 will execute, and loop 3 at Line 14 will iterate 2^n times, with each iteration costing a single step, thus the loop 3 will take a totol of 2^n steps.
As a result, the total runtime of executing $f(M)$ is $1 + n + 2^n$, which is $\Omega(2^n)$

Finally, since $WC(n) \in \mathcal{O}(2^n) \wedge WC(n) \in \Omega(2^n)$, therefore we can conclude that $WC(n) \in \Theta(2^n)$.

(b) **Proof:** WTP: $BC(n) \in \Theta(n^2)$, which is: $\forall n \in \mathbb{N}$, $\exists M \in \mathcal{I}_{func,M}$, running time of executing $f(M) \leq n^2$ and $\forall n \in \mathbb{N}$, $\forall M \in \mathcal{I}_{func,M}$,running time of executing $f(M) \geq n^2$

First, we will prove that $BC(n) \in \mathcal{O}(n^2)$, which is: $\forall n \in \mathbb{N}$, $\exists M \in \mathcal{I}_{func,M}$,running time of executing $f(M) \leq n^2$
Let $n \in \mathbb{N}$, and pick a adjacency matrix M with the following property: except for the diagonal of the adjacency matrix, all the other entries are filled with 1.
The initialization lines before the for loop take 1 step (i.e., have runtime independent of the input size).
For a fixed iteration of the outer loop, the inner loop at Line 7 iterates n times and take a constant time with each iteration. Since $M[i][j] = 1$, for all $i, j \in \{0, 1, 2, ..., n-1\}$ (except for the diagonal of the adjacency matrix), the variable *count* will never be equal to 0, which means that the algorithm will never enter the if block at Line 9 and reach the break statement at Line 11. The outer loop at Line 5 thus will iterate n times, with each iteration taking n steps. Therefore after ending loop 1 at Line 5 and loop 2 at Line 7, the total runtime is n^2. Since the algorithm never reaches the if block at Line 9, the variable $found\_isolated$ is always False. As a result, we won't enter the if block at Line 13 and loop 3 will not run any times.
Thus, the overall running time of executing $f(M)$ is $1 + n^2$, which is $\mathcal{O}(n^2)$

Next, we will prove that $BC(n) \in \Omega(n^2)$, which is: $\forall n \in \mathbb{N}$, $\forall M \in \mathcal{I}_{func,M}$, running time of executing $f(M) \geq n^2$
Let $n \in \mathbb{N}$, let $M \in \mathcal{I}_{func,M}$ and consider the running time of this algorithm on an input adjacency matrix M of n by n.
The initialization lines before the for loop take 1 step (i.e., have runtime independent of the input size).
For a fixed iteration of the outer loop, the inner loop at Line 7 takes n iterations, with each iteration costing a single step. If the variable *count* never equals to 0, then we won't enter the if block at Line 9 and the variable $found\_isolated$ will always be False. Therefore, the outer loop at Line 5 will iterate n times, with each iteration taking n steps. Since the variable $found\_isolated$ is False, then we won't reach the if block at Line 13 and loop 3 at Line 14 will not run any times.
Therefore, the total running time of executing $f(M)$ is at least $1 + n^2$, which is $\Omega(n^2)$

Finally, since $BC(n) \in \mathcal{O}(n^2) \wedge BC(n) \in \Omega(n^2)$, then we can conclude that $BC(n) \in \Theta(n^2)$.

(c) Let $n \in \mathbb{N}$, formula for the number of adjacency matrices of size n-by-n that represent valid graphs is: $2^{\frac{n(n-1)}{2}}$

(d) **Proof by induction:** we define a predicate $P(n)$: The number of adjacency matrices of size n-by-n is: $2^{\frac{n(n-1)}{2}}$
WTP: $\forall n \in \mathbb{N}$, $P(n)$
Base Case: when $n = 0$

$$\begin{aligned} 2^{\frac{n(n-1)}{2}} &= 2^{\frac{0\cdot(-1)}{2}} && \text{(since } n = 0\text{)} \\ &= 2^0 \\ &= 1 \end{aligned}$$

By intuition, the only possible matrix is the empty matrix(a 0-by-0 matrix). Therefore, the formula works for when $n = 0$
Thus, $P(0)$ holds.

Inductive Step: Let $k \in \mathbb{N}$, and assume $P(k)$ is true, which is: the number of adjacency matrices of size k-by-k is: $2^{\frac{k(k-1)}{2}}$, WTS: $P(k+1)$ is true, which is: the number of adjacency matrices of size $(k+1)$-by-$(k+1)$ is: $2^{\frac{(k+1)k}{2}}$.
Let $G = (V, E)$ be a graph, and let $V = \{0, 1, 2, ..., n-1\}$ be the vertices of the graph. Let M be a two-dimensional n-by-n array containing 0's or 1's called adjacency matrix.

Let's consider a $(k+1)$-by-$(k+1)$ adjacency matrix. The first k columns and k rows making it a k-by-k adjacency matrix which represents the graph containing the first k vertices. By induction hypothesis, the number of adjacency matrices of size k-by-k is: $2^{\frac{k(k-1)}{2}}$. Since the adjacency matrix is symmetrical (an edge $\{i, j\}$ is equivalent to an edge $\{j, i\}$), and that no vertex can ever be adjacent to itself. This means that for all $i, j \in \{0, 1, 2, ..., n-1\}$, $M[i][j] == M[j][i]$, and that $M[i][i] = 0$. Therefore, we only have to consider k different entries in the last $(k+1)^{th}$ column (Since $M[k+1][k+1] = 0$), for each entry in the $(k+1)^{th}$ column, there are two possibilities (i.e., 0 or 1). Thus, there are a total of 2^k possibilities. So the number of adjacency matrices of size $(k+1)$-by-$(k+1)$ is:

$$\begin{aligned} 2^{\frac{k(k-1)}{2}} \cdot 2^k &= 2^{\frac{k(k-1)}{2}+k} && \text{(By induction hypothesis)} \\ &= 2^{\frac{k^2-k+2k}{2}} \\ &= 2^{\frac{k^2+k}{2}} \\ &= 2^{\frac{(k+1)k}{2}} \end{aligned}$$

Therefore, $P(k+1)$ is true
Therefore, the number of adjacency matrices of size n-by-n is: $2^{\frac{n(n-1)}{2}}$

(e) **Proof:** Let $n \in \mathbb{N}$, WTP: the number of n-by-n adjacency matrices that represent a graph with at least one isolated vertex is at most $n \cdot 2^{\frac{(n-1)(n-2)}{2}}$.

Let $G = (V, E)$ be a graph, and let $V = \{0, 1, 2, ..., n-1\}$ be the vertices of the graph. Let E be defined as: a set E of pair of objects, where each pair (v_1, v_2) consists of two

distinct vertices - i.e.,$v_1, v_2 \in V$ and $v_1 \neq v_2$ - and is called an edge of the graph.
Let M be a two-dimensional n-by-n array containing 0's or 1's called adjacency matrix.

Let's consider the set $V = \{0, 1, 2, ..., n-1\}$, let $i \in \mathbb{N}$ and $i \leq n-1$. We pick an arbitrary vertex v_i in the set V to be the isolated vertex. Since $\forall G = (V, E)$, $|E| \leq \frac{|V|(|V|-1)}{2}$ (cited from the course notes), there are at most $\frac{(n-1)(n-2)}{2}$ edges in the graph, which means that there are $\frac{(n-1)(n-2)}{2}$ entries in the adjacency matrix, with each entry containing two possibilities: 0 or 1 (since v_i is already chosen to be the isolated vertex, then the number of rest vertices are $n-1$). As a result, the number of n-by-n adjacency matrices that represent a graph with one particular isolated vertex v_i is at most $2^{\frac{(n-1)(n-2)}{2}}$. Since $|V| = n$, we can choose v_0 or v_1 or... v_{n-1} to be the particular isolated vertex (There are total n choices).
Therefore the number of n-by-n adjacency matrices that represent a graph with at least one isolated vertex is at most $n \cdot 2^{\frac{(n-1)(n-2)}{2}}$.

(f) **Proof:** WTP: $AC(n) \in \Theta(n^2)$, which is: $AC(n) \in \Omega(n^2) \wedge AC(n) \in \mathcal{O}(n^2)$
Let $n \in \mathbb{N}$, let $\mathcal{I}_n$ be the set of inputs with all valid adjacency matrices, and let M be any valid adjacency matrix.

First, we will show that: $AC(n) \in \Omega(n^2)$.
Since by question (b), we know that $BC(n) \in \Theta(n^2)$, meaning that $BC(n) \in \Omega(n^2)$, which is: $\forall n \in \mathbb{N}$, $\forall M \in \mathcal{I}_n$, running time of executing $f(M) \geq n^2$
Therefore,

$$\begin{aligned} AC(n) &= \frac{1}{|\mathcal{I}_n|} \sum_{M \in \mathcal{I}_n} \textit{running time of executing } f(M) \\ &\geq \frac{1}{|\mathcal{I}_n|} \cdot |\mathcal{I}_n| \cdot n^2 && \text{(since } BC(n) \in \Omega(n^2)\text{)} \\ &= n^2 \end{aligned}$$

Thus, $AC(n) \in \Omega(n^2)$

Next, we want to show that $AC(n) \in \mathcal{O}(n^2)$
There are two types of adjacency matrices, one type are the adjacency matrices that represent the graph containing isolated vertices and the other type are the adjacency matrices that represent the graph without isolated vertices.

Hence, let $\mathcal{I}_{with\_isolated\_vertices}$ ($|\mathcal{I}_{with\_isolated\_vertices}| < |\mathcal{I}_n|$) be the set of inputs with all adjacency matrices that represent the graph containing isolated vertices, for any $M \in \mathcal{I}_{with\_isolated\_vertices}$, let the runtime of executing $f(M)$ be denoted as $RT_1(n)$

Let $\mathcal{I}_{without\_isolated\_vertices}$ ($|\mathcal{I}_{without\_isolated\_vertices}| < |\mathcal{I}_n|$) be the set of inputs with all adjacency matrices that represent the graph without isolated vertices. For each $M \in \mathcal{I}_{without\_isolated\_vertices}$, let the running time of executing $f(M)$ be denoted as $RT_2(n)$.

By question (a) and (b), we know that $RT_1(n) \in \mathcal{O}(2^n)$(the worst-case), which is:
$\forall M \in \mathcal{I}_{with\_isolated\_vertices}$, $RT_1(n) \leq 2^n$;
and $RT_2(n) \in \mathcal{O}(n^2)$(the best-case), meaning that: $\forall M \in \mathcal{I}_{without\_isolated\_vertices}$, $RT_2(n) \leq n^2$

Therefore,

$$
\begin{aligned}
AC(n) &= \frac{1}{|\mathcal{I}_n|} \sum_{M \in \mathcal{I}_n} running\ time\ of\ executing\ f(M) \\
&= \frac{1}{|\mathcal{I}_n|} [\sum_{M \in \mathcal{I}_{with\_isolated\_vertices}} RT_1(n) + \sum_{M \in \mathcal{I}_{without\_isolated\_vertices}} RT_2(n)] \\
&\leq \frac{1}{|\mathcal{I}_n|} [|\mathcal{I}_{with\_isolated\_vertices}| \cdot 2^n + |\mathcal{I}_{without\_isolated\_vertices}| \cdot n^2] \\
&\leq \frac{1}{|\mathcal{I}_n|} [n \cdot 2^{\frac{(n-1)(n-2)}{2}} \cdot 2^n + |\mathcal{I}_{without\_isolated\_vertices}| \cdot n^2] \\
&\qquad\qquad \text{(since by question (e), } |\mathcal{I}_{with\_isolated\_vertices}| \leq n \cdot 2^{\frac{(n-1)(n-2)}{2}}) \\
&\leq \frac{1}{|\mathcal{I}_n|} [n \cdot 2^{\frac{(n-1)(n-2)}{2}} \cdot 2^n + |\mathcal{I}_n| \cdot n^2] \qquad \text{(since } |\mathcal{I}_{without\_isolated\_vertices)}| < |\mathcal{I}_n|) \\
&= \frac{1}{2^{\frac{n(n-1)}{2}}} \cdot n \cdot 2^{\frac{(n-1)(n-2)}{2}} \cdot 2^n + \frac{1}{|\mathcal{I}_n|} \cdot |\mathcal{I}_n| \cdot n^2 \quad \text{(since by question (c), } |\mathcal{I}_n| = 2^{\frac{n(n-1)}{2}}) \\
&= 2n + n^2
\end{aligned}
$$

Therefore, $AC(n) \in \mathcal{O}(n^2)$
Since $AC(n) \in \Omega(n^2) \wedge AC(n) \in \mathcal{O}(n^2)$, then we can conclude that $AC(n) \in \Theta(n^2)$.