CSC165 Problem Set 5

Ruijian An & Make Zhang & Haoran Fang

Q1. Proving Correctness

(a) Use simple induction:

Base case: Let q = 1

Want to prove: $At\ the\ end\ of\ iteration\ 1\ of\ Main\ Loop, \forall i, j\ \in V, A[i][j] = 1 \Rightarrow PathLength(G, i, j, 2)$

Let $i, j\ \in V$. Assume $A[i][j] = 1$

Case 1: $A[i][j] = 1$ in the initial adjacent matrix A.

i.e. i, j are adjacent, so path between i, j has length 1.

So $PathLength(G, i, j, 2)$

Case 2: $A[i][j] = 1$ where $i = j$, so path between i, j has length 0.

So $PathLength(G, i, j, 2)$

Case 3: After 1 iteration, from line 13 we would know that $\exists k\ \in range(n),\ A[i][k] = 1 \wedge A[k][j] = 1.\ i.e.\ the\ path\ between\ i, j\ should\ look\ like:\ \ 'i - k - j'$.

So, path between i, j has length 2.

So $PathLength(G, i, j, 2)$

Induction step: Let $n\ \in \mathbb{N}$, assume $1 \leq n \leq \lceil logn \rceil - 1$

Assume $at\ the\ end\ of\ iteration\ n\ of\ Main\ Loop, \forall i, j\ \in V, A[i][j] = 1 \Rightarrow PathLength(G, i, j, 2^n)$

Consider after n+1 iterations:

Case 1: $A[i][j] = 1$ in the initial adjacent matrix A.

i.e. i, j are adjacent, so path between i, j has length 1.

So $PathLength(G, i, j, 2^{n+1})$

Case 2: $A[i][j] = 1$ where $i = j$, so path between i, j has length 0.

So $PathLength(G, i, j, 2^{n+1})$

Case 3: From line 13 we would know that $\exists k\ \in range(n),\ A[i][k] = 1 \wedge A[k][j] = 1$

$A[i][k] = 1\ and\ A[k][j] = 1\ after\ n\ iterations,$

So, by induction hypothesis, $PathLength(G, i, k, 2^n) \wedge PathLength(G, k, j, 2^n)$

By the fact $\forall G = (V, E), \forall u, v \in V, \forall d \in \mathbb{N}, PathLength(G, u, v, 2d) \Leftrightarrow\ (\exists w \in V, PathLength(G, u, w, d) \wedge PathLength(G, w, v, d))$, we can conclude that $PathLength(G, i, j, 2^{n+1})$

(b) Use simple induction:

Base case: Let q = 1
Want to prove: $At\ the\ end\ of\ iteration\ 1\ of\ Main\ Loop, \forall i, j\ \in V, A[i][j] = 0 \Rightarrow \neg PathLength(G, i, j, 2)$
Let $i, j\ \in V$. Assume $A[i][j] = 0$
After 1 iteration,

Case 1: i, j is not connected. Hence there is no path between i, j, so $\neg PathLength(G, i, j, 2)$

Case 2: i, j is connected. after 1 iteration, $A[i][j] = 0$, so there exists NO $k\ in\ range(n)\ such\ that\ there\ is\ a\ path\ looks\ like\ that\ 'i - k - j'$. (Or we can explain it by assuming if there exist a k that is adjacent with both i and j, after 1 iteration, i and j will be connected, which A[i][j] will equal to 1, not 0.)
However, i, j is connected, so the path is longer than 2, thus $\neg PathLength(G, i, j, 2)$

Induction step: Let $n\ \in \mathbb{N}$, assume $1 \leq n \leq \lceil logn \rceil - 1$

Assume $At\ the\ end\ of\ iteration\ n\ of\ Main\ Loop, \forall i, j\ \in V, A[i][j] = 0 \Rightarrow \neg PathLength(G, i, j, 2^n)$.

Let $i, j\ \in V.\ Assume\ A[i][j] = 0$
Consider after n+1 iterations:

$A[i][j] = 0 \Rightarrow A1[i][j] = 0$ (according to the code on line 20)
So, if $A1[i][j] = 0$,
then the if statement "if $A[i][k] == 1\ and\ A[k][j] == 1$" on line 14
is false for all k in range(n). So $\forall k\ \in range(n), A[i][k] == 0\ or\ A[k][j] == 0$
is true, hence $\forall k \in range(n), A[i][k] = 0\ or\ A[k][j] = 0\ after\ n\ iterations$
By induction hypothesis, $\neg PathLength(G, i, k, 2^n)\ or\ \neg PathLength(G, k, j, 2^n)$.
By the fact, $\forall G = (V, E), \forall u, v \in V, \forall d \in \mathbb{N}, PathLength(G, u, v, 2d) \Leftrightarrow (\exists w \in V, PathLength(G, u, w, d) \wedge PathLength(G, w, v, d))$
Then negate both sides, $\forall G = (V, E), \forall u, v \in V, \forall d \in \mathbb{N}, \neg PathLength(G, u, v, 2d) \Leftrightarrow (\forall w \in V, \neg PathLength(G, u, w, d) \vee \neg PathLength(G, w, v, d))$
So, $\neg PathLength(G, i, j, 2^{n+1})$

(c)

(1) If $connected$ returns $True$:

Then all_conn is $True$. (according to the code on line 30)

So, the $if\ statement\ 'if\ A[0][i] == 0' is\ always\ False$

i.e. $\forall i\ \in range(n)$, $A[0][i]$ == 1 is $True$.

By (a), $\forall i, PathLength(G, 0, i, 2^q)\ for\ all\ i \in range(n)\ after\ q\ iterations$.

So, 0 is connected to all vertices

So, for any two vertices i , j, there always exists a path between i and j which are connected by 0.

So, graph is connected.

(2) If $connected$ returns $False$:

i.e. the loop has not returned early and the main loop runs $ceil(logn)\ times$.

$so\ all\_conn$ is always $False$. (according to the code on line 30)

So, A[0][i] = 0 for some i in range (n) (According to the code on line 25-26)

By (b) $\neg PathLength\left(G, 0, i,\ 2^{ceil(logn)}\right)$ after $ceil(logn)$ iterations,

Case 1: there is no path between 0, i, so the graph is not connected.

Case 2: Suppose there is a path between 0, i.

Then the length of path $> 2^{ceil(logn)} > n$.

By the definition of path from notes,

If there are only n vertices in the graph,

the length of path should less than n, contradiction.

So, there is no path between 0, i, hence the graph is not connected.

Q2. Runtime analysis

(a) The code basically has 4 loop. The first loop is from line 5 to line 6. The second loop is from line 8 to line 15. The third loop is from line 18 to line 21. The forth loop from line 24 to line 28.

The other part of the program are all parts of the basic operations. So, this part can be all considered as 1 basic operation.

The first loop iterates n times
The second loop contains an nested inner loop. This nested loop is a Three-level nested loop and each level of loop iterate n times. So the inner loop from line 11 to line 15 totally runs n^3 times. The outer loop runs log(n) times. Totally the first loop runs n^3log(n) times.
The third loop contains an inner loop. The inner loop runs n times. The outer loop also runs n times. So, the loop from line 18 to line 21 totally runs n^2 times.
We only consider the worst case running time, which means the forth loop will NOT end early. So, the fourth loop runs n times.

This program totally runs n + n^3logn + n^2+ n + 1 times.

$$\text{n} + \text{n\^{}3logn} + \text{n\^{}2} + \text{n} + 1 \in O(n^3 logn)$$
$$WC(n) \in O(n^3 logn)$$

(b) Now we consider the Best Case. Suppose A[0][i] == 1 for all i (the code on line 26), then all_conn = True, so the Main loop will return early (the code on line 31), thus the Main Loop only runs 1 time.

The code basically has 4 loop. The first loop is from line 5 to line 6.
The second loop is from line 8 to line 15. The third loop is from line 18 to line 21. The forth loop from line 24 to line 28.

The other part of the program are all parts of the basic operations.
So, those parts can be all considered as runtime 1.

The first loop iterates n times
The second loop contains an nested inner loop. This nested loop is a Three-level nested loop and each level of loop iterate n times. So the inner loop from line 11 to line 15 totally runs n^3 times. The outer loop runs 1 times since the best case the loop will end after 1 iteration of the main loop. Therefore, the total runtime of the second loop runs n^3.
The third loop contains an inner loop. The inner loop runs n times. The outer loop also runs n times. So, the loop from line 18 to line 21 totally runs n^2 times.
In the fourth loop, it runs n times.
Then on line 30, all_conn = True, so the Main loop will return early (the code on line 31), thus the Main Loop only runs 1 time.
This program totally runs n + (n^3 + n^2 + n) * 1 times, which is $\Omega(n^3)$

$$so\ BC(n) \in \Omega(n^3)$$

(c) Let $n \in \mathbb{N}$. $Assume\ n > 1\ and\ n\ is\ the\ number\ of\ vertices\ of\ graph.$

Define the best-case input family for each n:
Assume this graph is a graph that all vertices are connected by edges.
Therefore, $\forall\ i \in V$, A[0][i] == 1
From line 30, we know that if all_conn is True and the main loop only run one time.
The details of analysis is same as part (b), and we could conclude that:

$$BC(n) \in O(n^3)$$