Introduction to Algorithms

Lab2 – Routing

Time complexity:

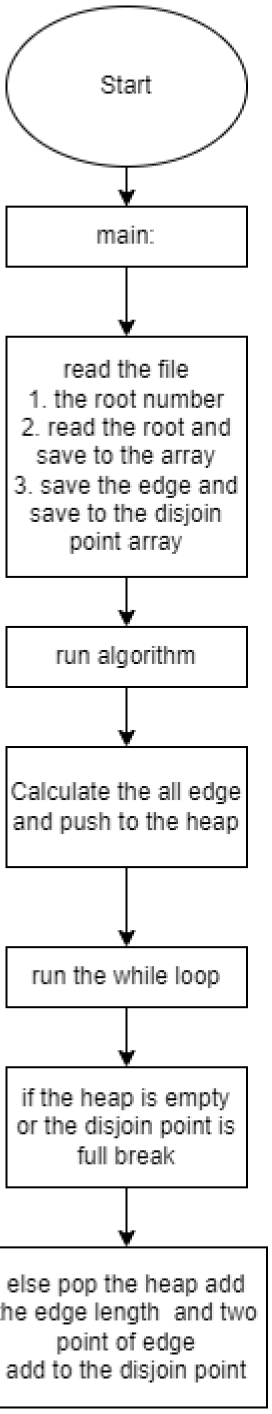
1. Read file
    - Step 1 Read Point $\text{Big O} = O(\text{m})$ m is number of the root of input
    - Step 2 Read Edge $\text{Big O} = O(\text{e})$ e is number of edge
2. Run algorithm
    - Calculate the all Edge $\text{Big O} = O(\text{n}\hat{}2)$ n is number of the root number
    - Run algorithm -> make heap $\text{Big O} = O(\text{n})$ n is a number of element
    - Pop Heap $\text{Big O} = O(\log(\text{n}))$ n is a number of element

Total time complexity Big $\text{Big O} = O(n^2 \log(\text{n}))$

About Design :

In this homework, I choose Kruskal's algorithm to be the base design to customize and optimize some cases, About the disjoin point part, I used the compression path method to speed up my code, In the order edge part, I used the heap replaces sorting to speed up code, it because make_heap algorithm run time is O(n) but sort algorithm run time is O(nlog(n)). My Test case generator and checker Lab2_checker (github.com) (This is my Github). I make some test cases, I find it if the ratio of nodes and edge number around to be 70%, My Code will run faster, in the 3000 nodes of the case, run time of around 0.2s. if it does not open the compiler optimize mode O3, the time will be around 1.2~1.7s.

Flow chat :


Start
main:
read the file
1. the root number
2. read the root and save to the array
3. save the edge and save to the disjoin point array
run algorithm
Calculate the all edge and push to the heap
run the while loop
if the heap is empty or the disjoin point is full break
else pop the heap add the edge length and two point of edge
add to the disjoin point