Report(2017DS_Prog1)

1. Pseudo Code:

1..**While** input_file! =Null

2. **Do**

If(We obtain equals to “#”)

Then infor_end=1;

else (We obtain !=”#”)

if(infor_end equals to 1)

then save the row to Data2D;

else save the row to direction;

ENDIF;

ENDIF;

Then save to Data2D

ENDWHILE

3. **FOR** j **IN** 0 **TO** direction.size()

DO (read two values of every row in direction, and

determine the max value, which is called max_push)

FOR i **IN** 0 **TO** max_push

DO

(push data to temp_queue except two data which is compared);

ENDFOR

ENDFOR

4. **DO**(Save chosen data to compar_a and compar_b matrix)

FOR iterator **in** compar_b.begin() **TO** compar_b.end()

IF(the chosen data in compar_b equals to The first data of compare_a)

then (the data push front to compar_a);

else the data push back to compar_b;

ENDIF

ENDFOR

5. **FOR** i **IN** 0 **TO** max_push

DO(erase the begin of Data2D);

ENDFOR

6. **IF**(exchange equals to 0)

Then(Push compar_a and then push temp_queue);

Else (Push temp_queue and then push compar_a);

ENDIF

2. Approach Works:

Explain my approach to solve the problem in detail.

We can follow these steps following to upon pseudo code.

1. Read “input_file.txt”.

2.Divide the input data and disparate save to Data 2D and direction (Both is 2-D daynamic Matrix.). We completed this step by the delimiter “#”.

3.Read Instruction and send data to temp_queue

4.Compare and exchange deque1 and deque2 in figure of PDF file.

5.Pop the from 0th to $(\max\_push)^{th}$ of the Data2D.

6.Determine which data will sort with priority.

One case is that temp_queue is earlier than compar_a.

Another case is that compar_a is ealier than temp_queue.

3. Time Complexity:

We must consider the two-dimension dynamic array. The

number of direction is (n-1) and we operate at most n rows in a loop. SO, the max number of operating is n*(n-1). we can infer the time complexity of this program is O(n^2).

4. What we learn in this problem:

1. Divide the big problem to many small tasks.
2. Be familiar with two-dimension dynamic arrays
3. Handle with queue and stack function.
4. Practice reading and writing files.