Daffodil International University

Summer 2020

Department of Computer Science and Engineering

Midterm Open Book Examination Answer Script

Full Marks: 25 Allowed Time: 4hrs (from: 2 p.m. to: 6 p.m.)

Date: Saturday 20 June 2020

Submission Date: Sat 20 June, 2020 by 10:00pm

General Information (must be filled by the student)

COURSE CODE: CSE214 **SECTION:** UC A **PROGRAM:CSE** ***DAY** **EVEN**

STUDENT ID: 173-15-928 **TIME STARTED:** 2 p.m **TIME ENDED:** 6 p.m

[Student must either TYPE or HAND WRITE the answers in this template; In case needed just write your detail on the paper using hand]

**** Plagiarism will be checked while you submit your response. You are advised to be honest during the open book exam.**

Mid Exam (Sum-2020)

Course Title: Algorithm
Course Code: CSE214
Section: UC-A
Student Name: Najib Mahmud
Student id: 173-15-928

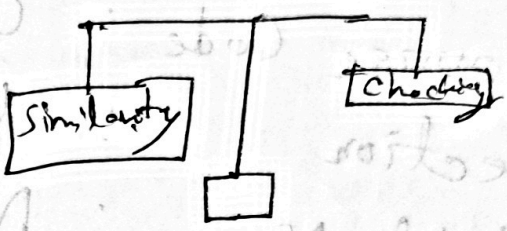
Ans. To The. Ques. No-1

I will create "Document Similarity Checker" report to cross check each report with one another. It will take the desired document that the user wants to check. After that it will match with the other it will match with the other reports and will tell it is similar or not.

Having across to some similarity report generated by the cross to semi semilarity crosscheck system. After the editor submit

a paper to cross check, he receives a report. This report look like this:

Sources → 31 %.
publications → 2%.
www.domain.com → 1%
etc.



There are two reasons this similarity obtain

(1) to quickly Plagiarized papers

(2) To check author is original

Ans. To. The. Ques. No → 2

(2) Algorithm:

Declare an array d [.....]

for i to n,

input (d[i]. " storing documents

" . user input

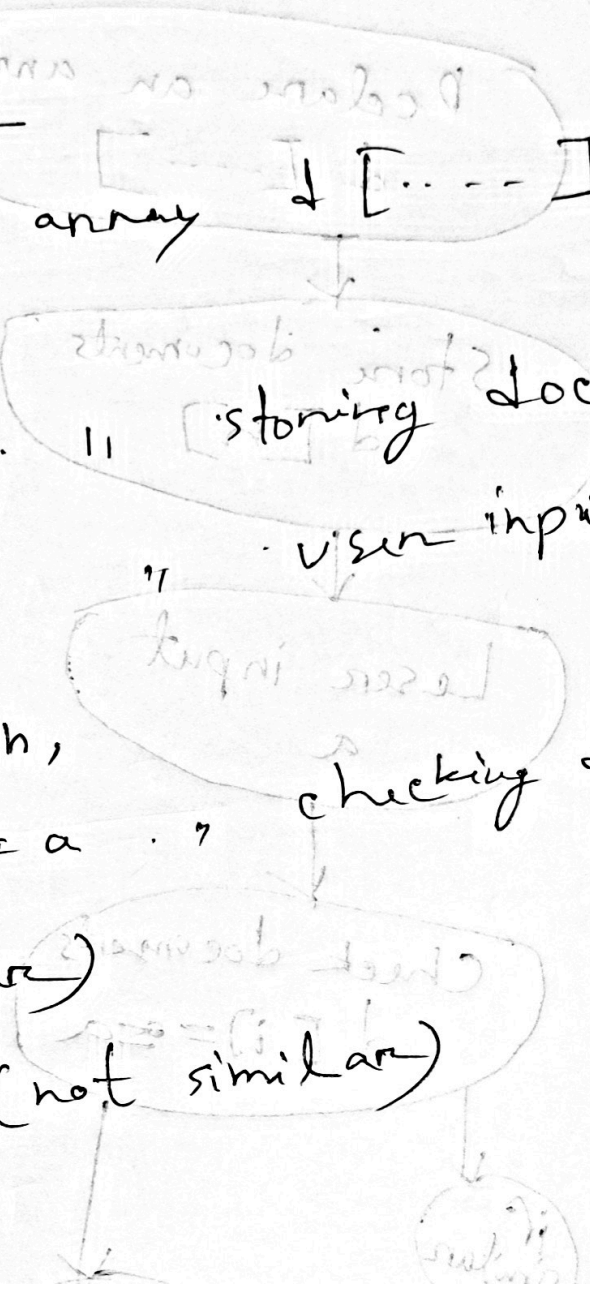
a

for 0 to n,

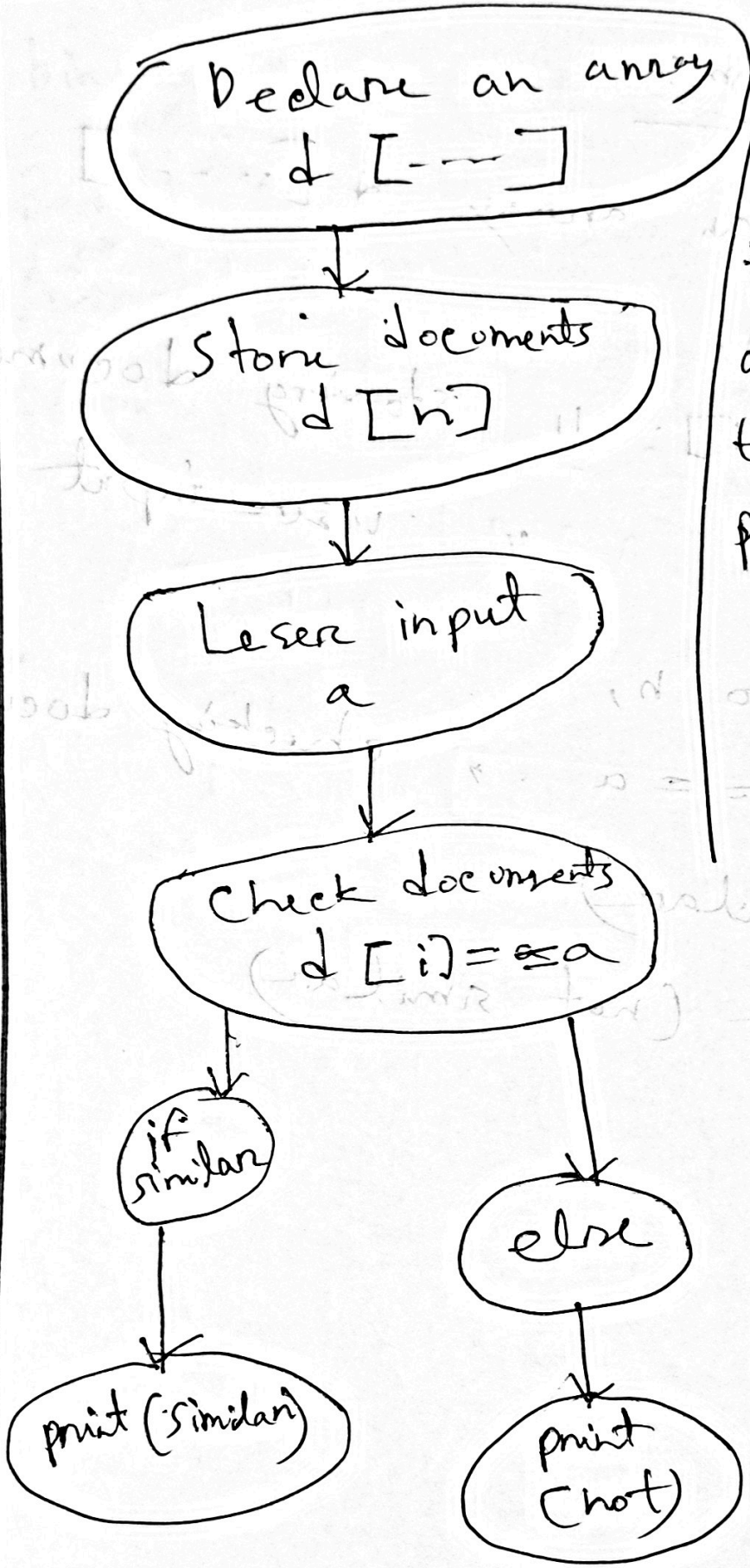
if .d [i] == a . " checking document

print (similar)

else print (not similar)



flow Chart:



I have used the Iterative approach here to solve the problem

Ans. To. The. Ques. No-3(i)

(3) (i)

I am using the Linear search to find how many words starting with 'H' or 'h' in this given MyDoc.txt document:

```
#include <stdio.h>
#include <string.h>
int main()
{ char arr[10000];
scanf("%[^\n]s", arr);
int size, i, count=0;
size = strlen(arr);
for(i=0; i<size; i++)
{ if(arr[i]==' ' && (arr[i+1]=='h'
                  || arr[i+1]=='H'))
    { ++count;
    }
}
```

```
printf("Total %d words start with 'H'
                    or 'h'.\n", count);

return 0;
}
```

Ans. To. The. Ques. No-3(ii)

The complexity of algorithm is $O(n)$

Ans. To The. Ques. No. 4

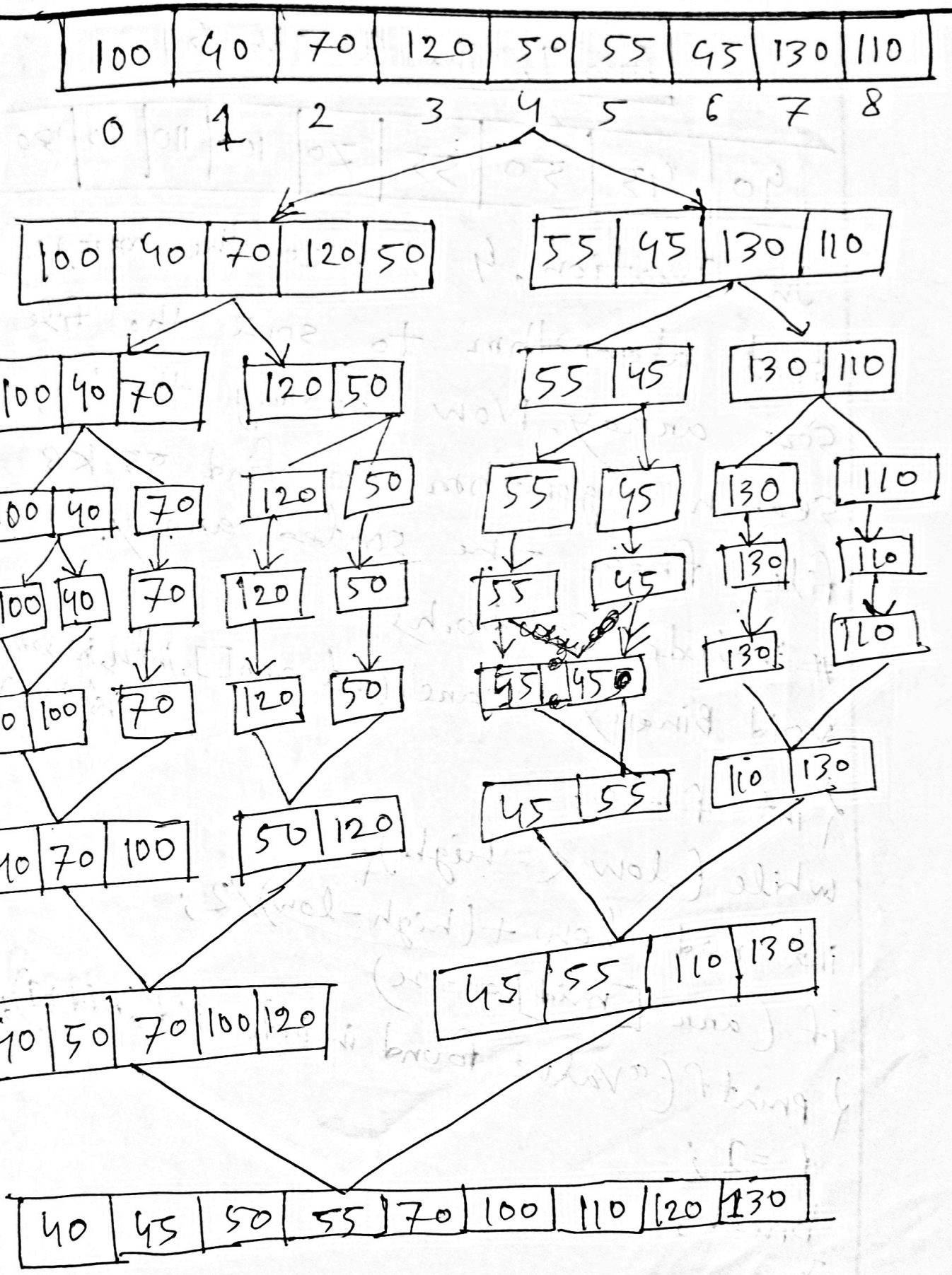
Item	0	1	2	3	4	5	6	7	8
Size	100 KB	40 KB	70 KB	120 KB	50 KB	55 KB	95 KB	130 KB	110 KB

I am using the Merge Sort algorithm to sort these files according to size. The merge sort algorithm follows the divide and conquer paradigm. Intuitively, it operates as follows:

Divide: Divide the n element sequence to be sorted into two subsequences of n/2 elements each.

Conquer: Sort the two subsequences recursively using merge sort.

Combine: Merge the two sorted subsequences to produce the sorted answer.


100 40 70 120 50 55 45 130 110
0 1 2 3 4 5 6 7 8
100 40 70 120 50
55 45 130 110
100 40 70
120 50
55 45
130 110
100 40
70
120
50
55
45
130
110
40 100
70
120
50
55
45
130
110
40 70 100
50 120
45 55
110 130
40 50 70 100 120
45 55 110 130
40 45 50 55 70 100 110 120 130

Ans. To. The. Ques. No-5

40	45	50	55	70	100	110	120	130

In Question 4, I used the merge sort algorithm to sort the file size array. Now choosing the Binary Search algorithm to find 55 KB file from the sorted array.

0	1	2	3	4	5	6	7	8
40	45	50	55	70	100	110	120	130

Start — End — Skip

key = 55

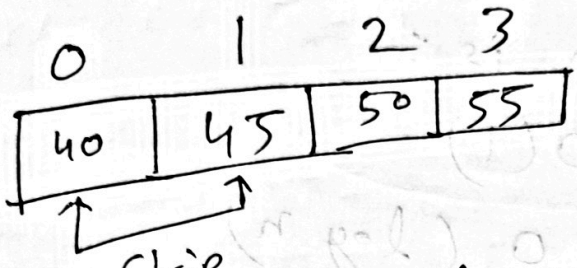
$$mid = \frac{(start + end)}{2} = \frac{0+8}{2} = 4$$

$55 < A[mid]$

$55 < A[4]$

Now, End = Mid - 1

Now, the array will be

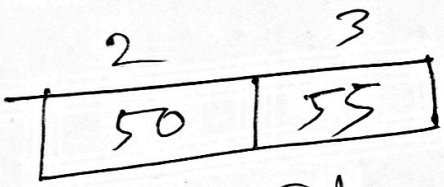


Again, $mid = \frac{start + End}{2} = \frac{0+3}{2} = 1$

$55 > A[mid]$

$55 > A[1]$

Now, start will be $= mid + 1$



$Mid = \frac{start + End}{2} = \frac{2+3}{2} = 2$

$55 > A[mid]$

$55 > A[2]$

Now, start will be $= mid + 1$

On this point start, end and middle are pointing the same value $A[3]$

And $A[3] = 55$,

So, the value will be found at the index $A[3]$

The time complexity of the binary search is:

Best case : O(1)

Average case : O (log n)

Worst case : O (log n)