# Algorithms Assignment

**Q4 –**

O(log(N)) Is the big O for binary search, I am using this algorithm to search in the main list for a specific student by surname. Binary search is O(log(N)) because it relies on divide and conquer strategy to fins a specific value in the main list. Similarly to the mergesort the data is divided in low, middle and high positions. Using a search key (the surname) once the search key becomes the middle element it is returned(the index is returned), if search key is on the left subarray the high index of the middle element is subtracted by 1 if it is in the right subarray the lowest index of the middle element is added 1, and keeps doing this recursively until the search key is the middle element.

Start program

SIZE = 10

MOD = 4

MAXST = 42

struct all surnames

{

char surnameslist[SIZE]

}

struct names

{

char firstname[SIZE]

char surname[SIZE]

}

struct modules

{

char code[SIZE]

int type

int maximum

int current

struct names students[MAXST]

}

start main

struct modules mods[MOD]= {{"DT265A", 0, 13, 0, {' ', ' '}},

{"DT265C", 0, 9, 0, {' ', ' '}},

{"DT265B", 1, 14, 0, {' ', ' '}},

{"DT8900", 1, 6, 0, {' ', ' '}}}

struct allsurnames surnames[MAXST];

int choice;

int end = 1;

char searchkey[SIZE];

DO

print {Menu 1.join 2.leave 3.Sorted Surnames 4.Display data 5.FULL-TIME students 6.what is your module 7.exit }

input choice

switch (choice)

case 1

join(mods)

break

case 2

leave(mods)

break

case 3

sorted\_surnames(surnames,mods)

break

case 4

display(mods)

break

case 5

PRINT "FULL-TIME students"

linear\_search(mods)

break

case 6

PRINT "enter you surname"

INPUT searchkey

int find = -1

int result

FOR i = 0 , i < MOD , i = i +1

IF (mods[i].current > 0 )

result = binary\_search(mods[i].students,mods[i].current, searchkey)

IF (result NOT -1)

find = i

break

END IF

END IF

END FOR

IF (find NOT -1)

PRINT "Name has been found"

END IF

ELSE

PRINT "studentnot found"

END ELSE

break

case 7

end = -1

break

default

PRINT "choose number form the menu"

break

END SWITCH

END DO WHILE (end == 1 )

END MAIN

START join (struct modules POINTER modsf)

int modch = 0

char mchoice[SIZE]

int find = 0

char firstname[SIZE]

char surname[SIZE]

INPUT mchoice, firstname, surname

FOR i= 0, i < MOD ,i= i+1

modch = compare(modsf[i].code , mchoice)

IF (modch == 0)

find = 1

IF (modsf[i].current < modsf[i].maximum)

modsf[i].current = modsf[i].current + 1

COPY (firstname INTO modsf[i].students[modsf[i].current - 1].firstname)

COPY (surname INTO modsf[i].students[modsf[i].current - 1].surname)

PRINT "you have been added to the module"

END IF

ELSE

PRINT "module is full"

END ELSE

break

END IF

END FOR

IF (find == 0 )

PRINT "choose an exsisting module"

END IF

END FUNCTION JOIN

START leave(struct modules POINTER modsf)

int modch = 0

int namexist = 0

char mchoice[SIZE]

int find = 0

char surname[SIZE]

INPUT mchoice,surname

FOR (i= 0 , i < MOD, i = i +1)

modch = COMPARE (mchoice TO modsf[i].code)

IF (modch = 0)

find = 1

FOR j=0, j <modsf[i].current, j = j+1

nameexist = COMPARE(surname TO modsf[i].students[j].surname)

IF (nameexist = 0 )

modsf[i].current = modsf[i].current - 1

FOR k = 0 , k < modsf[i].current, k = k +1

COPY (modsf[i].students[k + 1].surname INTO modsf[i].students[k].surname )

END FOR

PRINT "you have exited the module"

BREAK

END IF

END FOR

END IF

END FOR

IF find = 0

PRINT "please chose an existing module"

END IF

END FUNCTION leave

START binary\_search(struct names student[], int n, char searchkey[])

int low= 0

int high = n -1

int middle

WHILE (low <= high)

middle=(low + high)/2;

int cmp = COMPARE (searchkey, student[middle].surname )

IF (cmp == 0)

RETURN middle

END IF

ELSE IF (cmp < 0)

high = middle + 1

END ELSE IF

ELSE IF (cmp > 0)

low = middle - 1

END ELSE IF

END WHILE

RETURN -1

END FUNCTION