Sarawak Campus

Assignment Cover Sheet

(for individual and group assignments)

This cover sheet is to be attached to all assignments, both hard copy and electronic format

ASSIGNMENT DETAILS

Unit Code	COS10003	Unit Title	COMPUTER AND LOGIC ESSENTIALS
Tutorial/Lab Group	1	Lecturer/Tutor Name	YAKUB SEBASTIAN
Assignment Title	**Understanding Search and Sort Algorithm**		
Due date		Date Received	

DECLARATION

For both individual and group assignments, in the case of assignment submission on behalf of another student, it is assumed that permission has been given. The University takes no responsibility for any loss, damage, theft, or alteration of the assignment.

To be completed if this is an individual assignment

I declare that this assignment is my individual work. I have not worked collaboratively, nor have I copied from any other student's work or from any other source/s, except where due acknowledgment is made explicitly in the text, nor has any part been written for me by another person.

Student Details	Student ID Number	Student Name	Student Signature
Student 1			

To be completed if this is a group assignment

We declare that this is a group assignment and that no part of this submission has been copied from any other student's work or from any other source except where due acknowledgment is made explicitly in the text, nor has any part been written for us by another person.

Student Details	Student ID Number(s)	Student Name(s)	Student Signature (s)
Student 1	100069056	CHUNG CHONG SIEN	
Student 2	100066303	KUEH HANG ZEN	
Student 3	100069564	STELLA BINTI LAURANCE	
Student 4	100069085	YVONNE WONG SING YING	
Student 5			

MARKER'S COMMENTS

Total Mark	Marker's Signature	Date

EXTENSION CERTIFICATE

This assignment has been given an extension by

Unit Convenor

Extended due date

Date Received

Binary Search Algorithm

A Persian Mathematician, Al-Khwārizmī, had created "algorism" from Medival Latin and Greek word "arithmos". Then the term "algorithm" replace those words in late 19th century. Many computer programs contain algorithms that give computer instructions to process all the data by performing calculation but it has to be stated in a precise way and steps. It has been shown in many researches that algorithm are really useful to solve complex computing problem (Rashedi & Saryazdi 2010). In computer binary search algorithm is a searching tools that help to positioned the value in a specific array. Each time the algorithms will compare the value in the middle to do the comparison. After the comparison, it starts to eliminate other array. The process is been done repeatedly until it finds the targeted value. If all the numbers that array holds are arranged in ascending order, the array will be eliminated half by half until no element is left.

Example:

2	17	19	32	44	59	65	78	99	100	234
[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]

Now, we're finding the number of 65. We need to find where the number is. The search space in the array above is 1 to 11. We choose the value in the middle in the 6th array. The value is 59 which is smaller than 65. It means that search space of 1 to 5 is smaller than 65 too. We will eliminate them and the search space is narrowed into 7 to 11.

65	78	99	100	234
[7]	[8]	[9]	[10]	[11]

We repeated the process by finding the median value again which is 99 but it is greater than the targeted value, 65. So we eliminate the search space 9 to 11 again because we know that by comparison with the median value the value of 9 to 11 are bigger than 65.

65	78
[7]	[8]

Now we have two ways to identify the targeted value which are how we choose the median value of the even element or chop off 78 and left only one element. Both help us to know the location of the targeted value which is 7.

From all the statement above we also can say that binary search only use a very little time to run which will never more than O (log N) to do the comparison and find the targeted value. Before binary search, logarithms are used but it takes a lot of time. Oppositely, compared with binary search, binary search can shorter the time and steps for us to find specific element or information through it.

Pseudo code

Pseudo code is an organised English language that can portray calculations. It lets the programmer to think logically while concentrating on the designing the algorithm so that the3 programmer would not be distracted easily. The pseudo code must be completed.

The binary search can be coded in several ways such as recursion ways**.** *Recursion ways* is a process of defining a problem or solution of a problem in a more simple way by breaking down all the complex input.

```
// initially called with low = 0, high = N - 1
  BinarySearch_Right(A[0..N-1], value, low, high)
{
      // invariants: value >= A[i] for all i < low
                     value < A[i] for all i > high
      if (high < low)
    return low

      mid = low + ((high - low) / 2)
// THIS IS AN IMPORTANT STEP TO AVOID BUGS

      if (A[mid] > value)
          return BinarySearch_Right (A, value, low, mid-1)
      else
          return BinarySearch_Right (A, value, mid+1, high)

  }
```

Another way is *Iterative Pseudo code.* According to mathematician, iteration refers to a process that the functions are used repeatedly; using the first inserted input become the input of second . The input can be used continuously.

```
BinarySearch_Right (A [0..N-1], value) {

       low = 0
       high = N - 1

       while (low <= high) {
           // invariants: value >= A[i] for all i < low

                          value < A[i] for all i > high

           mid = low +((high - low) / 2)
            // THIS IS AN IMPORTANT STEP TO AVOID BUGS

           if (A[mid] > value)
               high = mid - 1
           else
               low = mid + 1
       }
       return low
   }
```

Sorting Algorithm- Bucket Sort

Sorting means to arrange data in a computer or orders a list of object. According to Adamchik 2009,If the number of objects is small enough to fits into the main memory, sorting is called *internal sorting*. If the number of objects is so large that some of them reside on external storage during the sort, it is called *external sorting*. In this report, we will focus on Bucket Sort, an internal sorting algorithm.

Bucket sort is mainly useful when input is uniformly distributed over a range. How it works is pretty simple. Lets says that we have a list of number in an array which was arrange randomly as shown in the table below.Bucket sort runs in linear time on the average. It assumes that the input is generated by a random process that distributes elements uniformly over the interval [0, 1).

0	1	2	3	4	5
12	6	10	22	30	14

Now, we need to create buckets and put these number into the buckets.Let’s name the array as arr[].

The total number of elements in arr[]= 6

So, we can write N=6

The max and min values are 30 and 6 respectively

To sort the elements we will take 10 buckets marked from 0 to 9

We can represent this bucket as b

B

0
1
2
3
4
5
6
7
8
9

Now we need to find a divider which will be used to put the elements in a bucket

Divider=ceil((max+1)/bucket))

=ceil(30+1)/10)

=ceil(3.1)

=4

If the index is denoted for array arr[]

Then to put the element arr[i] in the correct bucket we will use the following formula:

B[j]= arr[i]

Where j= floor(arr[i]/divider)

B[j] = Arr[0]

J = floor(12/4)

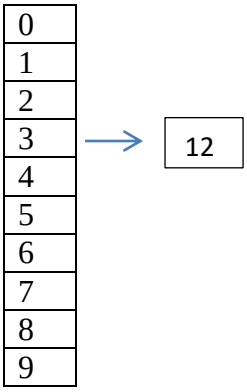
= floor(3)

= 3

Therefore, B[3]=arr[0]

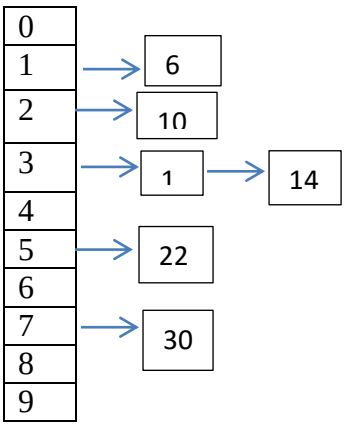
This means that the value of arr[0] will be put in bucket B[3]. The example is shown below:

B



The sorting will continue to Increment I by 1 and stop its process when all the elements in the array[I] are being filtered.

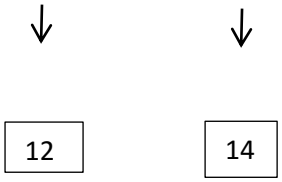
B



After we sort the elements to its respective buckets, we need to ensure that the number of elements in each bucket are arrange accordingly. In order to do this we will be using the insertion sort.Below are the examples of how these things are done:

Starting from the first bucket, we have 0 elements. Therefore we skip to the next bucket which is B[1]. B[1] only has one element which is 6. If there is only one element in a bucket, we cannot make a comparison so we did nothing to the element and move to the next bucket. If we found a bucket with 2 elements and more, we need to make the comparison between them like the one in B[3].

Example:



Is 12>14?

The answer is NO.

Therefore we don't have to swap their position.

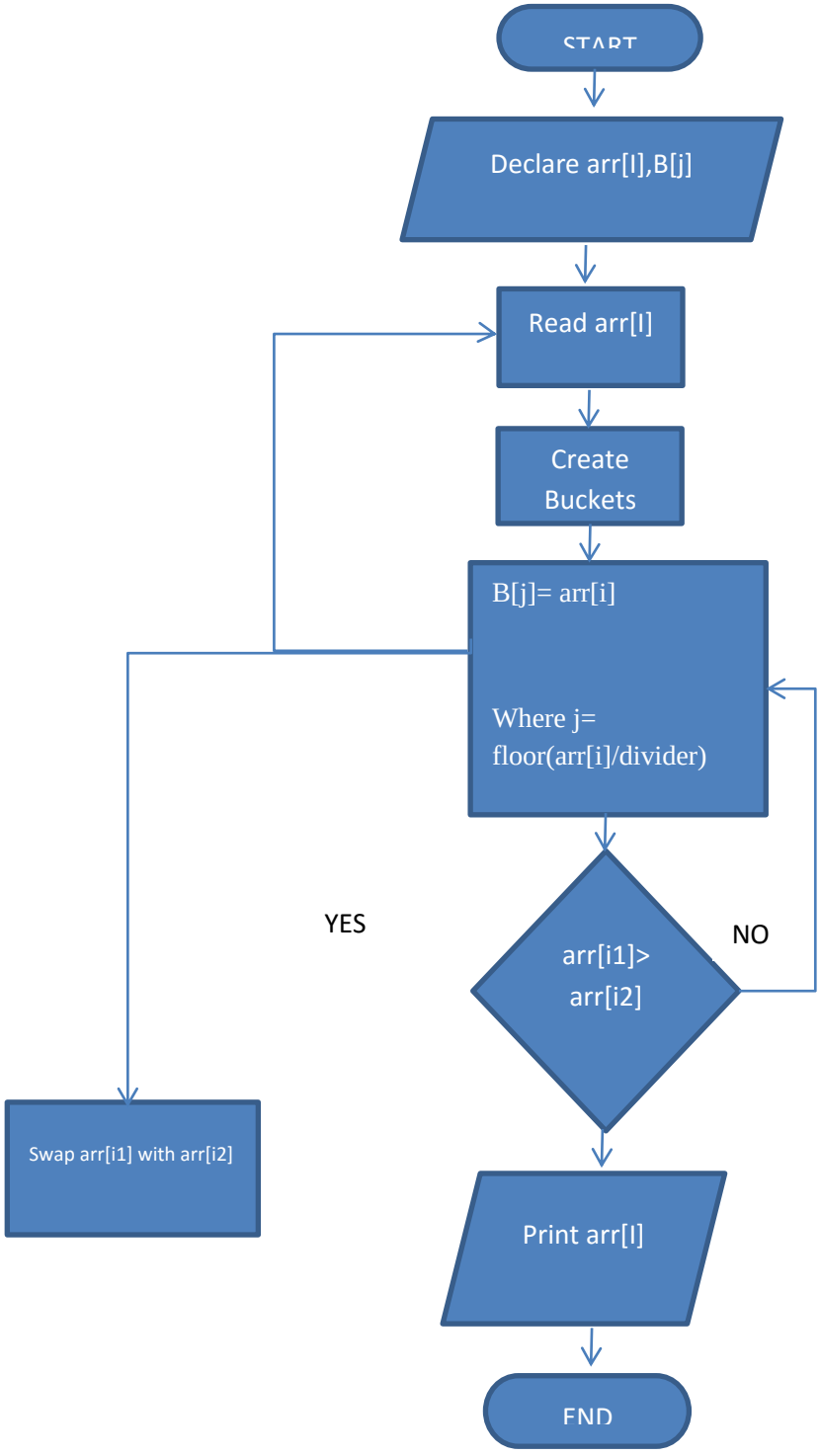
Swapping of the elements position are crucial to their conditions.

Last but not least, rearrange the elements in the bucket to the array.

0	1	2	3	4	5
6	10	12	14	22	30

Flowchart

Bucket sort can be represented using flowchart where we want to find the ascending order of a number.



Pseudocode

```
// Function to sort arr[] of size n using bucket sort
void bucketSort(float arr[], int n)
{
   // 1) Create n empty buckets
   vector<float> b[n];

   // 2) Put array elements in different buckets
   for (int i=0; i<n; i++)
   {
      int bi = n*arr[i]; // Index in bucket
      b[bi].push_back(arr[i]);
   }

   // 3) Sort individual buckets
   for (int i=0; i<n; i++)
      sort(b[i].begin(), b[i].end());

   // 4) Concatenate all buckets into arr[]
   int index = 0;
   for (int i = 0; i < n; i++)
       for (int j = 0; j < b[i].size(); j++)
         arr[index++] = b[i][j];
}
```

```
/* Driver program to test above funtion */
int main()
{
   float arr[] = {0.897, 0.565, 0.656, 0.1234, 0.665, 0.3434};
   int n = sizeof(arr)/sizeof(arr[0]);
   bucketSort(arr, n);

   cout << "Sorted array is \n";
   for (int i=0; i<n; i++)
     cout << arr[i] << " ";
   return 0;
}
```

Strength and Weaknesses of the algorithm

The fact that bucket sort degenerates to counting sort if each bucket has size 1 makes it to be seen as a generalization of counting sort . The variable bucket size of bucket sort allows it to use O(*n*) memory instead of O(*M*) memory, where *M* is the number of distinct values; in exchange, it gives up counting sort's O(*n* + *M*) worst-case behavior.

Bucket sort with two buckets is an effective version of quicksort where the pivot value always acts to be a middle value of the value range. While this choice is effective for uniformly distributed inputs, other means of choosing the pivot in quicksort such as randomly selected pivots make it more resistant to clustering in the input distribution.

References

Geek for Geeks 2014,*Bucket Sort*, Geeks for Geeks, viewed 11 November 2016,<http://www.geeksforgeeks.org/bucket-sort-2/>.

Adamchik, VS 2009,*Sorting*, Carnegie Mellon University, viewed 11 November 2016, <https://www.cs.cmu.edu/~adamchik/15-121/lectures/Sorting%20Algorithms/sorting.html>.

Topcoder (2016) Binary search – topcoder. https://www.topcoder.com/community/data-science/data-science-tutorials/binary-search/ ,viewed 18 November 2016.

Stoimen (2013) Stoimen's web log, http://www.stoimen.com/blog/2013/01/02/computer-algorithms-bucket-sort/ ,viewed18 November 2016).