جامعة الزقازيق
كلية الحاسبات و المعلومات
الفرقة الثانية

المشروع البحثى
المقدم عن مقرر
تصميم و تحليل الخوارزميات

اسم الطالب	**Hassan Ramadan Ibrahim Abdellatif**
رقم الجلوس	**2123**
الايميل	*HassanRamadanEbrahim@gmail.com*
البيانات الاتية تملىء للطالب فقط في حالة التخلفات	
اسم الطالب	N/A
الفرقة المقيد فيها حاليا	2
القسم المقيد فيه حاليا	N/A
رقم الجلوس بالفرقة المقيد بها حاليا	N/A

يرجىء ملىء بيانات الطالب في الجدول الاتى

للعلم انه تنطبق على تقييم هذا المشروع القواعد التي اقرتها وزارة التعليم العالى و مجلس جامعة الزقازيق و مجلس كلية الحاسبات و المعلومات بجامعة الزقازيق وانه في حالة تطابق نسخ جزئى او كلى من اى مصدر يعد رسوبا للطالب. يرجى كتابة التقرير باللغة الأجنبية .

Bucket sort Algorithm

Abstract

Bucket Sort Algorithm is, in the best case, a non-comparison based sorting algorithm with linear time complexity. It is useful when the input is spread over a range.

It runs on the elements by splitting them into buckets and sorting the buckets using a separate sorting algorithm.

The best time complexity for sorting n items using Bucket Sort is O(n), the average time complexity is O(n + k) and the worst time complexity is O(n2). The complexity of the space for Bucket Sort is O(n+k).

The advantage of bucket sorting is that once the elements are divided into buckets, each bucket can be processed independently of the others. This means that you often need to sort out a lot smaller arrays as a follow-up step than the original array. It also means that you can sort all the buckets in parallel.

The downside is that if you get a bad distribution on the buckets, you may end up doing a huge amount of extra work for no benefit or minimal benefit.

Introduction

Suppose we have array-based input with n elements that need to be sorted using bucket sort.

Bucket Sort works in the following steps:

1. Create an array of size n. Each element of this array is used as a bucket for storing elements.
2. Insert elements into the buckets from the array. The elements are inserted according to the range of the bucket. It is done by iterating through all elements. And for each element, insert the input element in the bucket with index that equal to converting the multiplication of size and input element into an integer.
3. The elements of each bucket are sorted using any stable sorting algorithm like quick sort or even using bucket sort recursively.

4. The elements from each bucket are gathered. It is done by iterating through the bucket and inserting an individual element into the original array in each cycle. The element from the bucket is erased once it is copied into the original array.

The main advantage of this algorithm in comparison with other sorting algorithms that it is much faster than all sorting algorithms such as Merge Sort, Heap Sort, or Quick Sort -when the input is uniformly distributed because it runs in linear time complexity O(n)- because these algorithms guarantee the best case time complexity of O(n*log(n)).

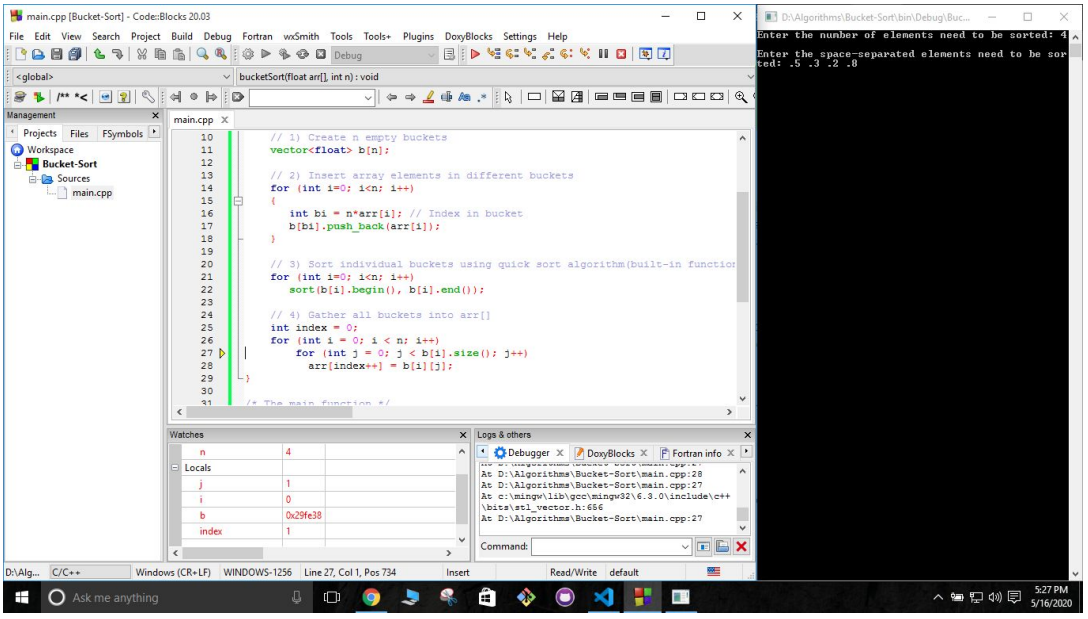
Results

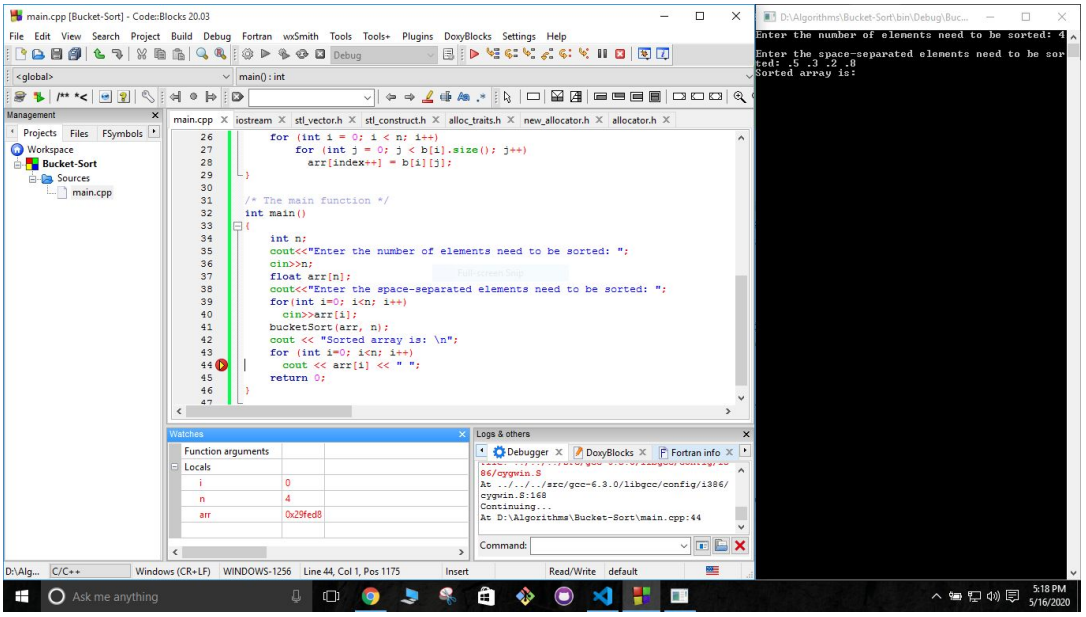
The code

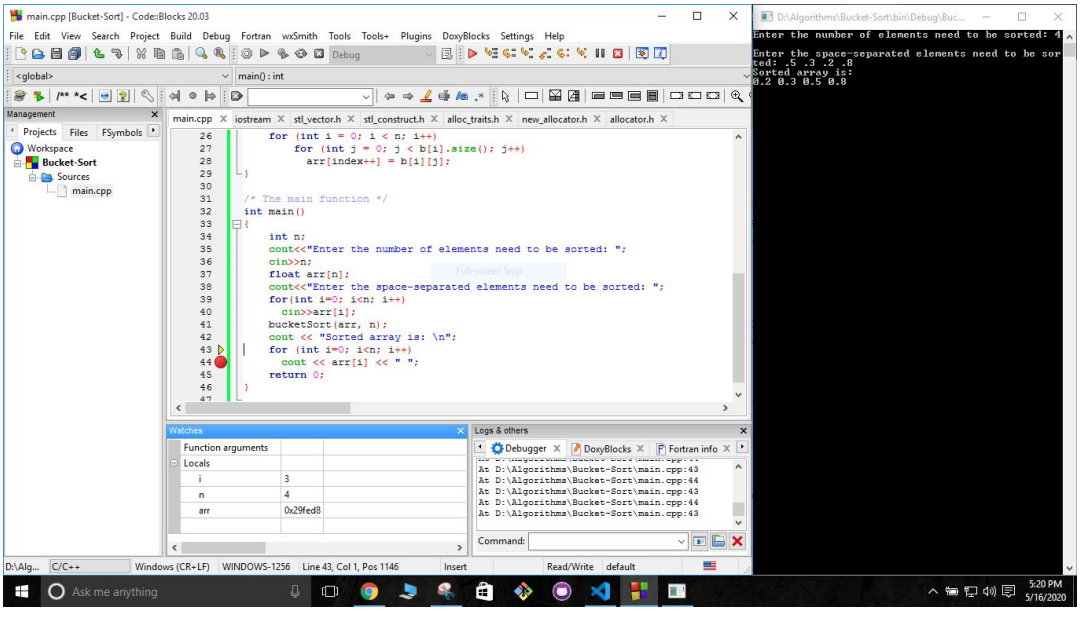
```
// C++ program to sort an array using bucket sort
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
//Function to return the max number in the array that is used as a
denominator that reduces the indexes of each element and increases
the range of each bucket.
float findMax(float A[], int n)
{
    if (n == 1)
        return A[0];
    return max(A[n-1], findMax(A, n-1));
}
// Function to sort arr[] of size n using bucket sort algorithm
void bucketSort(float arr[], int n)
{   //find the max number
    float max=findMax(arr,n);
    // 1) Create n empty buckets
    vector<float> b[n];
    // 2) Insert array elements in different buckets
```

```
    for (int i=0; i<n; i++)
    {
       int bi = n*arr[i]/(max+1); // Index in bucket
       b[bi].push_back(arr[i]);
    }
    // 3) Sort individual buckets using insertion sort
algorithm(built-in function in c++)
    for (int i=0; i<n; i++)
       sort(b[i].begin(), b[i].end());
    // 4) Gather 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];
}
/* The main function */
int main() {
    int n;
    cout<<"Enter the number of elements need to be sorted: ";
    cin>>n;
    float arr[n];
    cout<<"Enter the space-separated elements need to be sorted: ";
    for(int i=0; i<n; i++)
      cin>>arr[i];
    bucketSort(arr, n);
    cout << "Sorted array is: \n";
    for (int i=0; i<n; i++)
      cout << arr[i] << " ";
    return 0;
}
```

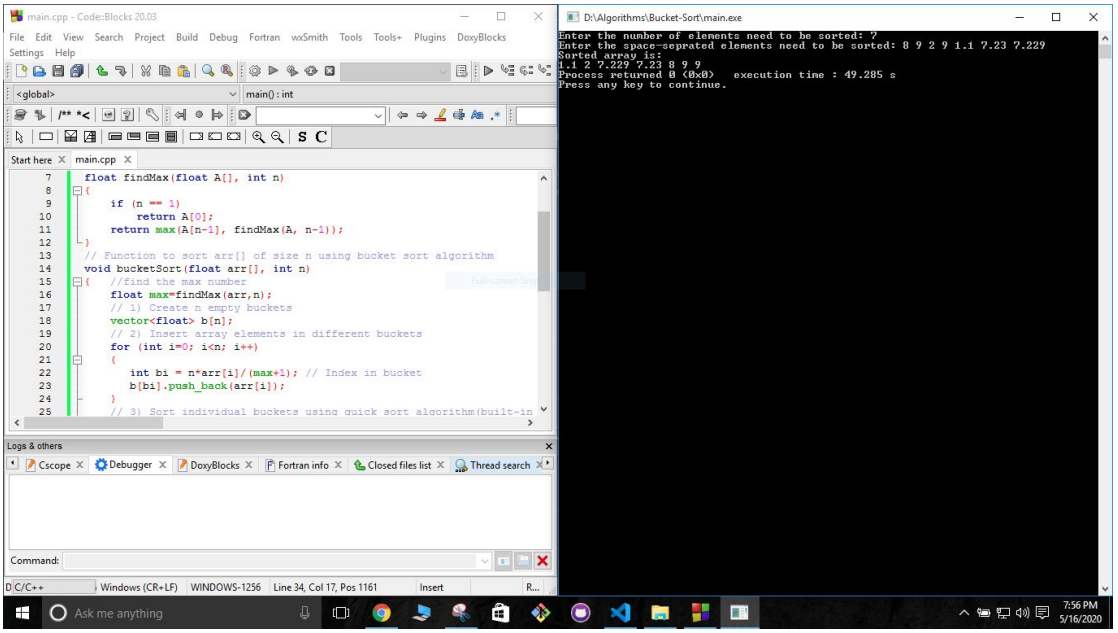
The following three screenshots display the variables, input, and output of the program through the timeline of execution:


main.cpp [Bucket-Sort] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global>
bucketSort(float arr[], int n) : void
main.cpp
// 1) Create n empty buckets
vector<float> b[n];
// 2) Insert 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 using quick sort algorithm(built-in functio
for (int i=0; i<n; i++)
sort(b[i].begin(), b[i].end());
// 4) Gather 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];
}
/* The main function */
Management
Projects Files FSymbols
Workspace
Bucket-Sort
Sources
main.cpp
Watches
n 4
Locals
j 1
i 0
b 0x29fe38
index 1
Logs & others
Debugger DoxyBlocks Fortran info
At D:\Algorithms\Bucket-Sort\main.cpp:28
At D:\Algorithms\Bucket-Sort\main.cpp:27
At c:\mingw\lib\gcc\mingw32\6.3.0\include\c++
\bits\stl_vector.h:656
At D:\Algorithms\Bucket-Sort\main.cpp:27
Command:
D:\Alg... C/C++ Windows (CR+LF) WINDOWS-1256 Line 27, Col 1, Pos 734 Insert Read/Write default
D:\Algorithms\Bucket-Sort\bin\Debug\Buc...
Enter the number of elements need to be sorted: 4
Enter the space-separated elements need to be sor
ted: .5 .3 .2 .8
Ask me anything
5:27 PM
5/16/2020


main.cpp [Bucket-Sort] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global>
main() : int
main.cpp iostream stl_vector.h stl_construct.h alloc_traits.h new_allocator.h allocator.h
for (int i = 0; i < n; i++)
for (int j = 0; j < b[i].size(); j++)
arr[index++] = b[i][j];
}
/* The main function */
int main()
{
int n;
cout<<"Enter the number of elements need to be sorted: ";
cin>>n;
float arr[n];
cout<<"Enter the space-separated elements need to be sorted: ";
for(int i=0; i<n; i++)
cin>>arr[i];
bucketSort(arr, n);
cout << "Sorted array is: \n";
for (int i=0; i<n; i++)
cout << arr[i] << " ";
return 0;
}
Management
Projects Files FSymbols
Workspace
Bucket-Sort
Sources
main.cpp
Watches
Function arguments
Locals
i 0
n 4
arr 0x29fed8
Logs & others
Debugger DoxyBlocks Fortran info
86/cygwin.S
At ../../../src/gcc-6.3.0/libgcc/config/i386/
cygwin.S:168
Continuing...
At D:\Algorithms\Bucket-Sort\main.cpp:44
Command:
D:\Alg... C/C++ Windows (CR+LF) WINDOWS-1256 Line 44, Col 1, Pos 1175 Insert Read/Write default
D:\Algorithms\Bucket-Sort\bin\Debug\Buc...
Enter the number of elements need to be sorted: 4
Enter the space-separated elements need to be sor
ted: .5 .3 .2 .8
Sorted array is:
Ask me anything
5:18 PM
5/16/2020


main.cpp [Bucket-Sort] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global>
main() : int
main.cpp iostream stl_vector.h stl_construct.h alloc_traits.h new_allocator.h allocator.h
for (int i = 0; i < n; i++)
for (int j = 0; j < b[i].size(); j++)
arr[index++] = b[i][j];
}
/* The main function */
int main()
{
int n;
cout<<"Enter the number of elements need to be sorted: ";
cin>>n;
float arr[n];
cout<<"Enter the space-separated elements need to be sorted: ";
for(int i=0; i<n; i++)
cin>>arr[i];
bucketSort(arr, n);
cout << "Sorted array is: \n";
for (int i=0; i<n; i++)
cout << arr[i] << " ";
return 0;
}
Management
Projects Files FSymbols
Workspace
Bucket-Sort
Sources
main.cpp
Watches
Function arguments
Locals
i 3
n 4
arr 0x29fed8
Logs & others
Debugger DoxyBlocks Fortran info
At D:\Algorithms\Bucket-Sort\main.cpp:43
At D:\Algorithms\Bucket-Sort\main.cpp:44
At D:\Algorithms\Bucket-Sort\main.cpp:43
At D:\Algorithms\Bucket-Sort\main.cpp:44
At D:\Algorithms\Bucket-Sort\main.cpp:43
Command:
D:\Alg... C/C++ Windows (CR+LF) WINDOWS-1256 Line 43, Col 1, Pos 1146 Insert Read/Write default
D:\Algorithms\Bucket-Sort\bin\Debug\Buc...
Enter the number of elements need to be sorted: 4
Enter the space-separated elements need to be sor
ted: .5 .3 .2 .8
Sorted array is:
0.2 0.3 0.5 0.8
Ask me anything
5:20 PM
5/16/2020

Another example:



Discussion

Bucket Sort Time Complexity Analysis

1. Time complexity of findMax() = O(g(n))

$g(n) = g(n-1) + 1$

base case: $g(1) = 1$

$g(n-1) = g(n-2) + 1 \quad \Rightarrow \quad g(n) = g(n-2) + 2$

Suppose, $k = n - 1$

$g(n) = g(n-k) + k$

$= g(n-n+1) + n - 1$

$= g(1) + n - 1 = 1 + n - 1 = n$

$O(g(n)) = O(n)$

2. Time complexity of sort() = O(n) in best case = O(n^2) in worse case.

Suppose, n_i represents the number of elements in bucket i. So $n = \sum_{i=1}^{n} n_i$

In the best case:

Time complexity of sorting all buckets = $\sum_{i=1}^{n}(O(n_i)) = O(\sum_{i=1}^{n}(n_i)) = O(n)$

In the worst and average case:

Time complexity of sorting all buckets = $\sum_{i=1}^{n}(O(n_i\text{^}2)) = O(\sum_{i=1}^{n}(n_i\text{^}2)) =$ O(n^2)

3. Time complexity of bucketSort() function (Bucket Sort Algorithm)

T(n) = T(n)[findMax()] + T(n)[insert elements into buckets] + n*T(n_i)[sort()] + T(n)[gather elements] + c

As proved in point 2, n*T(n_i)[sort()] = T(n)[sort()]

So, T(n) = n + c_1*n + T(n)[sort()] + n + c

In the best case:

T(n) = n + c_1*n + n + n + c

T(n) = c_2*n + c

T(n) = O(n)

In the worst and average case:

T(n) = n + c_1*n + n^2 + n + c

T(n) = c_2*n + n^2 + c

T(n) = O(n^2)

The best case is when the elements are uniformly distributed in buckets with an almost equal number of elements in each bucket and are already sorted in buckets.

The worst case is that when there are close range elements in the array, they are likely to be placed in the same bucket. This may result in some buckets having more elements than others. The complexity is even worse when the elements are in reverse order.

Bucket Sort Space Complexity Analysis

S(n) = number of elements in the list = number of buckets + total number of elements in each bucket = k + n

So, S(n) = O(n+k)

The main limitation of this algorithm is that It is not efficient in the cases when elements are not uniformly distributed. It's a very big limitation because when that happens the algorithm loses its main advantage which is the linear time complexity.

Conclusion

Sorting algorithms should be as fast as possible. So we learn many different sorting algorithms because there is not an efficient algorithm everywhere, which is efficient in one place, not efficient in some other place. Bucket sort is an efficient algorithm when the elements need to be sorted are uniformly distributed. It sorts them in a linear time. otherwise it is not efficient.

References

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. Introduction to Algorithms, Third Edition (3rd. ed.). The MIT Press, pages 197 : 213.
2. Horsmalahti, P. (2012). Comparison of bucket sort and radix sort. *arXiv preprint arXiv:1206.3511*.

Web-Sites:

1. Bucket Sort- **kent.edu**
2. Bucket sort- **slideshare.net**