A Pancake Sorting Problem

We have discussed Pancake Sorting in the previous post. Following is a problem based on Pancake Sorting.
Given an an unsorted array, sort the given array. You are allowed to do only following operation on array.

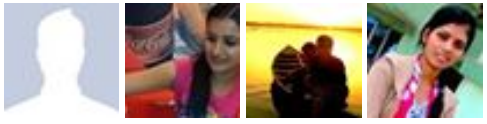
```
flip(arr, i): Reverse array from 0 to i
```

Imagine a hypothetical machine where flip(i) always takes O(1) time. **Write an efficient program for sorting a given array in O(nLogn) time on the given machine**. If we apply the same algorithm here, the time taken will be O(n^2) because the algorithm calls findMax() in a loop and find findMax() takes O(n) time even on this hypothetical machine.

We can use insertion sort that uses binary search. The idea is to run a loop from second element to last element (from i = 1 to n-1), and one by one insert arr[i] in arr[0..i-1] (like standard insertion sort algorithm). When we insert an element arr[i], we can use binary search to find position of arr[i] in O(Logi) time. Once we have the position, we can use some flip operations to put arr[i] at its new place. Following are abstract steps.

```
// Standard Insertion Sort Loop that starts from second element
for (i=1; i < n; i++) ----> O(n)
{
  int key = arr[i];

  // Find index of ceiling of arr[i] in arr[0..i-1] using binary search
  j = celiSearch(arr, key, 0, i-1); ----> O(logn) (See this)
```

```
  // Apply some flip operations to put arr[i] at correct place
}
```

Since flip operation takes O(1) on given hypothetical machine, total running time of above algorithm is O(nlogn). Thanks to Kumar for suggesting above problem and algorithm.

Let us see how does the above algorithm work. ceilSearch() actually returns the index of the smallest element which is greater than arr[i] in arr[0..i-1]. If there is no such element, it returns -1. Let the returned value be j. If j is -1, then we don't need to do anything as arr[i] is already the greatest element among arr[0..i]. Otherwise we need to put arr[i] just before arr[j].
So how to apply flip operations to put arr[i] just before arr[j] using values of i and j. Let us take an example to understand this. Let i be 6 and current array be {12, 15, 18, 30, 35, 40, **20**, 6, 90, 80}. To put 20 at its new place, the array should be changed to {12, 15, 18, **20**, 30, 35, 40, 6, 90, 80}. We apply following steps to put 20 at its new place.

1) Find j using ceilSearch (In the above example j is 3).
2) flip(arr, j-1) (array becomes {18, 15, 12, 30, 35, 40, **20**, 6, 90, 80})
3) flip(arr, i-1); (array becomes {40, 35, 30, 12, 15, 18, **20**, 6, 90, 80})
4) flip(arr, i); (array becomes {**20**, 18, 15, 12, 30, 35, 40, 6, 90, 80})
5) flip(arr, j); (array becomes {12, 15, 18, **20**, 30, 35, 40, 6, 90, 80})

Following is C implementation of the above algorithm.

```
#include <stdlib.h>
#include <stdio.h>

/* A Binary Search based function to get index of ceiling of x in
   arr[low..high] */
int ceilSearch(int arr[], int low, int high, int x)
{
    int mid;

    /* If x is smaller than or equal to the first element,
       then return the first element */
    if(x <= arr[low])
        return low;

    /* If x is greater than the last element, then return -1 */
    if(x > arr[high])
        return -1;
```

Geometric Algorithms

Popular Posts

All permutations of a given string
Memory Layout of C Programs
Understanding "extern" keyword in C
Median of two sorted arrays
Tree traversal without recursion and without stack!
Structure Member Alignment, Padding and Data Packing
Intersection point of two Linked Lists
Lowest Common Ancestor in a BST.
Check if a binary tree is BST or not
Sorted Linked List to Balanced BST

```
    /* get the index of middle element of arr[low..high]*/
    mid = (low + high)/2;  /* low + (high - low)/2 */

    /* If x is same as middle element, then return mid */
    if(arr[mid] == x)
        return mid;

    /* If x is greater than arr[mid], then either arr[mid + 1]
      is ceiling of x, or ceiling lies in arr[mid+1...high] */
    if(arr[mid] < x)
    {
        if(mid + 1 <= high && x <= arr[mid+1])
            return mid + 1;
        else
            return ceilSearch(arr, mid+1, high, x);
    }

    /* If x is smaller than arr[mid], then either arr[mid]
       is ceiling of x or ceiling lies in arr[mid-1...high] */
    if (mid - 1 >= low && x > arr[mid-1])
        return mid;
    else
        return ceilSearch(arr, low, mid - 1, x);
}

/* Reverses arr[0..i] */
void flip(int arr[], int i)
{
    int temp, start = 0;
    while (start < i)
    {
        temp = arr[start];
        arr[start] = arr[i];
        arr[i] = temp;
        start++;
        i--;
    }
}

/* Function to sort an array using insertion sort, binary search and f
void insertionSort(int arr[], int size)
{
    int i, j;

    // Start from the second element and one by one insert arr[i]
    // in already sorted arr[0..i-1]
    for(i = 1; i < size; i++)
```

```
    {
        // Find the smallest element in arr[0..i-1] which is also grea
        // or equal to arr[i]
        int j = ceilSearch(arr, 0, i-1, arr[i]);

        // Check if there was no element greater than arr[i]
        if (j != -1)
        {
            // Put arr[i] before arr[j] using following four flip oper
            flip(arr, j-1);
            flip(arr, i-1);
            flip(arr, i);
            flip(arr, j);
        }
    }
}

/* A utility function to print an array of size n */
void printArray(int arr[], int n)
{
    int i;
    for (i = 0; i < n; ++i)
        printf("%d ", arr[i]);
}

/* Driver program to test insertion sort */
int main()
{
    int arr[] = {18, 40, 35, 12, 30, 35, 20, 6, 90, 80};
    int n = sizeof(arr)/sizeof(arr[0]);
    insertionSort(arr, n);
    printArray(arr, n);
    return 0;
}
```

Output:

```
6 12 18 20 30 35 35 40 80 90
```

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Recent Comments

Related Tpoics:

- Remove minimum elements from either side such that 2*min becomes more than max
- Divide and Conquer | Set 6 (Search in a Row-wise and Column-wise Sorted 2D Array)
- Bucket Sort
- Kth smallest element in a row-wise and column-wise sorted 2D array | Set 1
- Find the number of zeroes
- Find if there is a subarray with 0 sum
- Divide and Conquer | Set 5 (Strassen's Matrix Multiplication)
- Count all possible groups of size 2 or 3 that have sum as multiple of 3

Writing code in comment? Please use **ideone.com** and share the link here.

10 Comments GeeksforGeeks

Sort by Newest

vinod • 4 months ago

Following is a divide and conquer method for this problem. Used a special flip two indices. (the flip function given in the problem statement can also be used

http://ideone.com/6vOUmk

  • Reply • Share ›

Prateek Caire • 9 months ago

We can also build max heap. top most element will give the position of max ar NLogN

```
/* Paste your code here (You may delete these lines if not writing c
```

1   • Reply • Share ›

kartheek J • a year ago

We Can Also Merge Sort Right..
In The Merge Instead Replacing Reversing/Flip Will be there as Flip O(1) Opre complete.

```
/* Paste your code here (You may delete these lines if not writing c
```

  • Reply • Share ›

reallyunreal • a year ago

Doesn't binary search take place on an already sorted array?

  • Reply • Share ›

algobard reallyunreal • a year ago

Yes, it does. And that's what is being *correctly* used here.

This approach uses insertion sort. So going by how insertion sort work
sorted.
Just pen down an example and observe :)

```
/* Paste your code here (You may delete these lines if not wri
```


ibnipun10 • a year ago

Bubble sort can also work here, right?

sreeram • a year ago

Lets say we know how to swap two elements,we can apply any one O(nlogn)
say i 2){
flip(arr,j-1);
flip(arr,j-i-2);
flip(arr,j-1);}
flip(arr,j);
flip(arr,j-i);
flip(arr,j);
}

Please let me know if I am wrong !

sreeram ➦ sreeram • a year ago

sorry for typos I meant this

say i 2){
flip(arr,j-1);
flip(arr,j-i-2);
flip(arr,j-1);}

```
flip(arr,j);
flip(arr,j-i);
flip(arr,j);
}
```

sreeram sreeram • a year ago

```
[say i < j
swap(int *arr,int i ,int j ){
flip(arr,j-1);
flip(arr,j-i-2);
flip(arr,j-1);}
flip(arr,j);
flip(arr,j-i);
flip(arr,j);
}]
```

1 • Reply • Share ›

Suku • a year ago

facebook interview question

• Reply • Share ›