Construction of Longest Monotonically Increasing Subsequence (N log N)

In my previous post, I have explained about longest monotonically increasing sub-sequence (LIS) problem in detail. However, the post only covered code related to querying size of LIS, but not the construction of LIS. I left it as an exercise. If you have solved, cheers. If not, you are not alone, here is code.

If you have not read my previous post, read here. Note that the below code prints LIS in reverse order. We can modify print order using a stack (explicit or system stack). I am leaving explanation as an exercise (easy).

```
#include <iostream>
#include <string.h>
#include <stdio.h>
using namespace std;

// Binary search
int GetCeilIndex(int A[], int T[], int l, int r, int key) {
   int m;

   while( r - l > 1 ) {
      m = l + (r - l)/2;
      if( A[T[m]] >= key )
         r = m;
      else
         l = m;
   }

   return r;
}

int LongestIncreasingSubsequence(int A[], int size) {
   // Add boundary case, when array size is zero
   // Depend on smart pointers

   int *tailIndices = new int[size];
   int *prevIndices = new int[size];
   int len;

   memset(tailIndices, 0, sizeof(tailIndices[0])*size);
   memset(prevIndices, 0xFF, sizeof(prevIndices[0])*size);

   tailIndices[0] = 0;
   prevIndices[0] = -1;
   len = 1; // it will always point to empty location
   for( int i = 1; i < size; i++ ) {
      if( A[i] < A[tailIndices[0]] ) {
```

```
            // new smallest value
            tailIndices[0] = i;
        } else if( A[i] > A[tailIndices[len-1]] ) {
            // A[i] wants to extend largest subsequence
            prevIndices[i] = tailIndices[len-1];
            tailIndices[len++] = i;
        } else {
            // A[i] wants to be a potential condidate of future subsequence
            // It will replace ceil value in tailIndices
          int pos = GetCeilIndex(A, tailIndices, -1, len-1, A[i]);

          prevIndices[i] = tailIndices[pos-1];
          tailIndices[pos] = i;
        }
    }
    cout << "LIS of given input" << endl;
    for( int i = tailIndices[len-1]; i >= 0; i = prevIndices[i] )
        cout << A[i] << "   ";
    cout << endl;

    delete[] tailIndices;
    delete[] prevIndices;

    return len;
}

int main() {
    int A[] = { 2, 5, 3, 7, 11, 8, 10, 13, 6 };
    int size = sizeof(A)/sizeof(A[0]);

    printf("LIS size %d\n", LongestIncreasingSubsequence(A, size));

    return 0;
}
```

Run on IDE

Exercises:

1. You know Kadane's algorithm to find maximum sum sub-array. Modify Kadane's algorithm to trace starting and ending location of maximum sum sub-array.

2. Modify Kadane's algorithm to find maximum sum sub-array in a circular array. Refer GFG forum for many comments on the question.

3. Given two integers A and B as input. Find number of Fibonacci numbers existing in between these two numbers (including A and B). For example, A = 3 and B = 18, there are 4 Fibonacci numbers in between {3, 5, 8, 13}. Do it in O(log K) time, where K is max(A, B). What is your observation?

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

39 Comments Category: Arrays

About Venki

Software Engineer

View all posts by Venki →

Related Posts:

- Longest Span with same Sum in two Binary arrays
- Count Inversions of size three in a give array
- Find the subarray with least average
- Count triplets with sum smaller than a given value
- Find zeroes to be flipped so that number of consecutive 1's is maximized
- Reorder an array according to given indexes
- Find maximum value of Sum(i*arr[i]) with only rotations on given array allowed
- Find maximum average subarray of k length

(Login to Rate and Mark)

4 Average Difficulty : **4/5.0**
Based on **2** vote(s)

Add to TODO List

Mark as DONE

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.