Longest Increasing Sub sequence বাংলা টিউটোরিয়াল

Longest Increasing Subsequences টিউটোরিয়াল

স্বাগতম !!! এটি আমার লিখা প্রথম টিউটোরিয়াল । ভাষাজনিত ভুলত্রুটি ক্ষমাসুন্দর দৃষ্টিতে দেখবেন আর কোথাও কোন ভুল ধরা পড়লে কমেন্টে জানাবেন । চেষ্টা করব ভুল শুধরে দেবার জন্য ।

আজকের টপিক হলঃ Longest Increasing Subsequence (LIS)
মূলত LIS এর দুইটি মোটামুটি Efficient এ্যালগরিদম আছে । একটি হচ্ছে Dynamic Programming যার কমপ্লেক্সিটি হল O(N^2) আর আরেকটি হল Binary Search এর যার কমপ্লেক্সিটি হল O(NlgN) । এছাড়া আরো একটি টেকনিক আপনারা পাবেন ইউটিউবে যেটি একটি বাংলা টিউটোরিয়াল এবং সেটি শুধুমাত্র Distinct নাম্বারের জন্য কাজ করে ।

তো আজকে যে টেকনিক টি আলোচনা করব সেটি হল O(NlgN) । যেটি O(N^2) টেকনিক থেকেও এফিসিয়েন্ট । তো দেরি না করে শুরু করা যাক ।

[ ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ][ ] [ ]

প্রথমেই Subsequence কি এটা একটু জেনে নেই ……

Subsequence হল কোন একটা Sequence থেকে কিছু সঙ্খ্যক ইলিমেন্ট নিয়ে তৈরি করা একটি Sequence । এই ইলিমেন্ট গুলো পরস্পর পাশাপাশি নাও হতে পারে । যেমন ধরা যাক : [3 8 1 9 3 7] . যদি এখান থেকে আমি [8 1 9 3] নিই তাহলে এটা একটা Subsequence হবে । আমরা যদি আমরা [3 1 9 7] নিই তাহলে সেটাও একটা Subsequence হবে । একইভাবে [3 8 9 3 7] তাহলে এটাও একটা Subsequence হবে । তবে মনে রাখতে হবে Subsequence এর লেন্থ সর্বনিম্ন 1 এবং সর্বোচ্চ কমপ্লিট Array Size ও হতে পারে । তাহলে সে হিসেবে [3 8 1 9 3 7] তার নিজের একটা Subsequence ।

এরপর আসি Longest Increasing Subsequence কি ?

Longest Increasing Subsequence হচ্ছে এমন একটা Sequence যেটা সাইজে বড় এবং যে Sequence এর প্রত্যেক টা ইলিমেন্ট তার পরবর্তী ইলিমেন্ট এর চাইতে ছোট হবে ।

ধরা যাক একটা Sequence [3 8 1 9] . এখানে Subsequence গুলো হচ্ছে :

```
[3], [ 3 8 ], [3 1], [3 9], [3 8 1 ],[3 8 9], [3 1 9], [ 3 8 1 9], [8], [8 1], [8, 9]
[8 1 9],[1], [1 9], [9].
```

তাহলে এখানে,

Length 1 এর কয়টা Subsequence আছে? 4 টা
Length 2 এর কয়টা Subsequence আছে? 6 টা
Length 3 এর কয়টা Subsequence আছে? 4 টা
Length 4 এর কয়টা Subsequence আছে? 1 টা

এখন এদের মধ্যে সবচে বড় সাইজের Subsequence বের করতে হবে যেখানে ইলিমেন্ট গুলো Increasing অর্ডারে থাকে। তাহলে উপরের উদাহরনে LIS এর সাইজ হবে : 3 এবং LIS হবে [3 8 9] । এই Sequence এ 3, 8, 9 তিনটা ইলিমেন্টই increasing order এ আছে। **[3 < 8 < 9]**

Algorithm জানার আগে জানিয়ে রাখি এই এ্যালগরিদম শেখার জন্য Binary Search এ্যালগরিদম এর কন্সেপ্ট জানা লাগবে। যদি জানা না থাকে তাহলে এখান থেকে শিখে নেবার জন্য অনুরোধ করা যাচ্ছে এবং শেখার পর সামনের দিকে আগানোর জন্য পরামর্শ দেয়া যাচ্ছে।

[ ] [
] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ][ ] [ ]

প্রথমেই আমরা যেই কাজটা করব সেটা হল একটা Array নেবো যার নাম Main । এই এ্যারে তে আমার মূল ভ্যালু গুলো থাকবে আমরা যাদের LIS বের করব।

ধরা যাক Main টা হল :

```
Main [] = [ 3 4 -1 5 8 2 3 12 7 9 10 ]

Index[]  = [ 0 1 2 3 4 5 6 7 8 9 10 11]
```

এরপর আমরা আরো ৩ টা Array নেবো যাদের নাম দিলাম :
Temp[], Val[] এবং Res[] । একটা ভ্যারিয়েবল নিলাম Length যে LIS এর লেংথ কে স্টোর করবে।

- ⇨ Temp এ্যারে টা স্টোর করবে কোন একটা নির্দিষ্ট লেংথ এর LIS এর মিনিমাম ভ্যালুর index টি। এই এ্যারে টা একটা টেম্পোরারী এ্যারে এবং এটা পরবর্তীতে

আমাদের LIS লেংথ যদি বাড়ে তাহলে সেক্ষেত্রে সাহায্য করবে । আমরা কেন নির্দিষ্ট লেংথ এর LIS এর মিনিমাম ভ্যালু টি রাখছি ? কারণ হচ্ছে এরপরে আমার যে নাম্বার গুলো আসবে সেগুলোকে নিয়েও হয়তো আমরা বর্তমান LIS Length এর চাইতে বড় length এর LIS গঠন করতে পারি । পুরো বিষয়টি এখন পরিষ্কার না হলে থাক । পরবর্তীতে বিস্তারিত করতে গিয়ে বোঝা যাবে আশা করছি।

⇨ Res এ্যারে টা স্টোর করবে LIS এর রেজাল্ট । মানে হল Res এ্যারে টা আমার যেই LIS টা আসবে সেই LIS এর পাথ প্রিন্ট করার জন্য কাজে লাগবে ।

⇨ Val এ্যারে টা Temp এ্যারে এর i তম ইন্ডেক্স এ যেই ভ্যালুটা স্টোর করা থাকবে Main এ্যারে এর সেই ভ্যালু তম ইন্ডেক্স এর ভ্যালু কে স্টোর করবে । ধরা যাক Temp[i] = X . তাহলে, Val[j] = Main [X] মানে হল Val[j] = Main [Temp [i]] .

বেসিক জিনিস গুলা আপাতত শেষ । ও হ্যা উপরের এ্যারে টার LIS হবে - [-1 2 3 7 9 10] . কিভাবে সেটা নিচের এ্যালগরদিম অনুসারে ব্যাখ্যা করার চেষ্টা চালাব ।

ALGORITHM :

⇨ এবার প্রথম কাজ হল Res এ্যারে এর সবগুলো ইলিমেন্ট কে -1 করে দেয়া । কেন করেছি সেটা এ্যাল্গরিদম শেষে বোঝা যাবে ।

⇨ এরপর আমরা i = 0 থেকে ভ্যালু এক্সেস করা শুরু করব । প্রথমেই আছে i = 0. তার মানে প্রথম ইলিমেন্ট আমরা by default নিয়ে ফেললাম । যেহেতু লেংথ 1 কে আমরা By Default নিচ্ছি সেহেতু আমাদের এ্যারে গুলোর কোন চেঞ্জ হবে না । শুধুমাত্র Val এ্যারে এর 0 তম ইন্ডেক্স এ Main[Temp[0]] = Main [0]] = 3 বসবে । কেন By Default নিচ্ছি ? কারণ প্রত্যেকটা সঙ্খ্যা নিজেই একটা 1 লেন্থ এর LIS .

```
Main [] = [ 3 4 -1 5 8 2 3 12 7 9 10 ]

Temp []  = [ 0 0 0 0 0 0 0 0 0 0 0 ]

Res [] = [ -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1] .

Val [] = [ 3 ]

Length = 0.
```

⇨ এবার i = 1 হল । এখন আমরা সার্চ করে দেখব Main[1] = 4 ইলিমেন্ট টা Val এ্যারে এর ঠিক কোন জায়গায় পড়ে ? দেখা যাচ্ছে এটি পড়বে Val এ্যারে এর ডান দিকে বাইরে । এর মানে হল এই ভ্যালু টা কারেন্ট লেংথ এর মধ্যে যেই ইলিমেন্ট গুলো আছে তাদের সবার চাইতে বড় । এক্ষেত্রে আমাদের Temp এ্যারে এর 2nd পজিশনে i বসাবো এবং Res এর 2nd পজিশনে বসাবো এর ঠিক আগের ইন্ডেক্স এর Temp এ্যারে তে যে ভ্যালু টা স্টোর করা আছে সেই ভ্যালু টা । আর ভ্যালু এ্যারে তে 4 বসাতে হবে । তাহলে আমরা এটা কে এ্যারে তে যোগ করব এবং আমাদের Length কে 1 বাড়াবো ? তাহলে আমাদের এ্যারে এর অবস্থা দাঁড়াবে :

```
Main [] = [ 3 4 -1 5 8 2 3 12 7 9 10 ]

Temp []  = [ 0 1 0 0 0 0 0 0 0 0 0 ]

Res [] = [ -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1] .

Val [] = [ 3 4 ]

Length = 1.
```

⇨ এখন i = 2 হল । এখন আমরা সার্চ করে দেখব Main[2] = -1 ইলিমেন্ট টা Val এ্যারে এর ঠিক কোন পজিশনে পড়ে ? দেখা যাচ্ছে এটি পড়বে Val এ্যারে এর বাম দিকে বাইরে । তার মানে হল এই ইলিমেন্ট টা এই পর্যন্ত পাওয়া সবচাইতে ছোট ইলিমেন্ট এবং এই ইলিমেন্ট টা Val এ্যারে এর 0th পজিশনে বসবে ।

আমরা Val এ্যারের 0th পজিশন এবং Temp এ্যারে এর 0th পজিশন কে আপডেট করব । যেটা আগে ছিল Val[0] = 3 সেটা হবে Val[0] = -1 এবং Temp[0] = 0 হবে Temp[0] =2 . তবে এক্ষেত্রে LIS Length কিন্তু বাড়বে না কারণ এটি কিন্তু আমাদের ডানে যুক্ত হয় নি ।

এবং Res[] এ্যারে এর 0th পজিশন ও আপডেট হবে না । কারণ এটাই বর্তমানে সর্বনিম্ন ইলিমেন্ট যার কারনে এটা সর্ব বামে বসেছে । তাহলে আমাদের এ্যারে গুলোর যে অবস্থা টা হবে –

```
Main [] = [ 3 4 -1 5 8 2 3 12 7 9 10 ]

Temp []  = [ 2 1 0 0 0 0 0 0 0 0 0 ]

Res [] = [ -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1] .

Val [] = [ -1 4 ]

Length = 1.
```

এখন থেকে কিছু জিনিস আমরা একটু মাথায় রাখার চেষ্টা করব । কোন একটা ইলিমেন্ট -

** ডানে যুক্ত হবার মানে হল নতুন কোন একটা ইলিমেন্ট যেটা বর্তমান LIS লিস্ট এর সর্বশেষ ইলিমেন্ট এর চাইতে বড় এবং সেটি বর্তমান LIS লিস্ট এর মান 1 বৃদ্ধি করে ।

** কিন্তু একেবারে বামে যাবার মানে হল, হয় এই ইলিমেন্ট টা কে আমাদের বাদ দিতে হবে যাতে করে আমাদের LIS টা ঠিক থাকে অথবা এই সর্বনিম্ন ভ্যালু টা কে কোন একটা Sub Sequence এর প্রথম ইলিমেন্ট করে নতুন একটা Sub Sequence তৈরি করা যায় কিনা দেখতে হবে যার লেংথ বর্তমান Sub Sequence এর লেংথ চাইতে বড় ।

** আর যদি ওই ইলিমেন্ট এর পজিশন Val এ্যারে এর 1st এবং Last ইন্ডেক্স এর মাঝখানে হয় তাহলে বুঝতে হবে পজিশনে এই ভ্যালু টা বসালে আমার LIS টা ইনপুটে আছে এরকম শেষের দিকের সর্বনিম্ন ভ্যালু গুলোর মাধ্যমে Sorted থাকবে । একটা উদাহরন দেয়া যায় ধরা যাক একটা Array হল [2 3 5 4] এবং LIS আসলো [2 3 5] , [2 3 4] এই এ্যালগরিদম অনুসারে প্রথমে [2 3 5] LIS হবে কিন্তু যখনি 4 কে introduce করা হবে তখনি LIS হবে [2 3 4] । আশা করছি বিষয়টি পরিষ্কার হয়েছে ।

⇨ এবার i = 3 . তাহলে Main[3] = 5 । যেটা পড়বে Val এ্যারে এর সবচাইতে বাইরে মানে সর্ব ডানের ইলিমেন্ট । যার মানে হল LIS এ যুক্ত হওয়া নতুন সদস্য । এক্ষেত্রে আমাদের এ্যারে আপডেট হবে ঠিক একইভাবে যেভাবে আমরা i = 1 এর জন্য করেছিলাম । LIS length বেড়ে 1 + 1 = 2 হবে ।

```
Main [] = [ 3 4 -1 5 8 2 3 12 7 9 10 ]

Temp []  = [ 2 1 3 0 0 0 0 0 0 0 0 ]

Res [] = [ -1 0 -1 1 -1 -1 -1 -1 -1 -1 -1] .

Val [] = [ -1 4 5 ]
```

```
Length = 1+1.
```

Res[3] তে কেন আমরা 1 বসালাম । কারন হল Temp[2] এর আগের ইন্ডেক্স Temp[1] এর ভ্যালু হল 1 । যার মানে হচ্ছে আমরা Res[3] পাচ্ছি Res[1] থেকে এবং Res[1] পাচ্ছি Res[0] থেকে । SO এখন পর্যন্ত আমাদের LIS হল [3 4 5] যেটা Temporary ।

⇨ এবার i = 4 . Main[4] = 8 . একই ভাবে আপডেট হবে ।

```
Main [] = [ 3 4 -1 5 8 2 3 12 7 9 10 ]

Temp []  = [ 2 1 3 4 0 0 0 0 0 0 0 ]

Res [] = [ -1 0 -1 1 3 -1 -1 -1 -1 -1 -1] .

Val [] = [ -1 4 5 8 ]

Length = 1+1+1.
```

⇨ So now, i = 5 . তাহলে Main[5] = 2 । ইলিমেন্ট 2 কিন্তু ডানদিকে বা বামদিকে কোনভাবেই আমাদের Val এ্যারের বাইরে যাবে না । তাহলে ?? ইলিমেন্ট টি Val এ্যারে এর মাঝখানে কোন একটা জায়গায় পড়বে এবং আমাদের Val এ্যারে কে সর্টেড রাখবে । তবে এক্ষেত্রে Res এ্যারে এর চেঞ্জ এর কারনে বর্তমান LIS কিন্তু চেঞ্জ হতে পারে কিন্তু LIS Length এর কোন চেঞ্জ আসবে না । তাহলে 2 ইলিমেন্টটি Val এ্যারে এর 2nd পজিশনে বসবে । আমাদের এ্যারে এর চেঞ্জ গুলো হবে -

```
Main [] = [ 3 4 -1 5 8 2 3 12 7 9 10 ]

Temp []  = [ 2 5 3 4 0 0 0 0 0 0 0 ]

Res [] = [ -1 0 -1 1 3 2 -1 -1 -1 -1 -1] .

Val [] = [ -1 2 5 8 ]

Length = 1+1+1.
```

Res[5] এ কেন আমরা 2 বসালাম ? কারণ Val এ্যারে এর 2nd পজিশন এ যেহেতু আমরা 2 বসাচ্ছি এবং একইসাথে Temp[1] কে আপডেট করছি i = 5 দিয়ে । রুল অনুসারে আমাকে Temp এর যে পজিশন টা আপডেট হচ্ছে সে পজিশনের আগের পজিশনে থাকা Temp এ্যারে এর ভ্যালু টা আমাকে Res এ্যারে এর কারেন্ট পজিশনে বসাতে হবে । সে কারনে Res[i] = 2 .

⇨ i = 6 and Main[i] = 3 । ইলিমেন্ট 3 কে সার্চ করলে দেখা যাবে এটি Val এ্যারে এর 3rd পজিশনে পড়ছে । So , আপডেট গুলো হবে –

```
Main [] = [ 3 4 -1 5 8 2 3 12 7 9 10 ]

Temp []  = [ 2 5 6 4 0 0 0 0 0 0 0 ]

Res [] = [ -1 0 -1 1 3 2 5 -1 -1 -1 -1] .

Val [] = [ -1 2 3 8 ]

Length = 1+1+1.
```

⇨ i = 7 and Main[i] = 12 । ইলিমেন্ট 12 কে সার্চ করলে দেখা যাবে এটি ডানদিকে Val এ্যারে এর বাইরে পড়বে । যার মানে এটি এখন পর্যন্ত সবচাইতে বড় ইলিমেন্ট । So, Rule মোতাবেক আমাদের আপডেট গুলো হবে –

```
 Main [] = [ 3 4 -1 5 8 2 3 12 7 9 10 ]

Temp []  = [ 2 5 6 4 7 0 0 0 0 0 0 ]

Res [] = [ -1 0 -1 1 3 2 5 4 -1 -1 -1] .

Val [] = [ -1 2 3 8 12 ]

Length = 1+1+1+1.
```

এবং LIS Length ও 1 বাড়বে ।

⇨ i = 8 and Main[i] = 7 । ইলিমেন্ট 7 এর অবস্থান হবে Val এ্যারে এর মাঝখানে । তাহলে আপডেট গুলো হবে –

```
Main [] = [ 3 4 -1 5 8 2 3 12 7 9 10 ]

Temp []  = [ 2 5 6 8 7 0 0 0 0 0 0 ]

Res [] = [ -1 0 -1 1 3 2 5 4 6 -1 -1] .

Val [] = [ -1 2 3 7 12 ]

Length = 1+1+1+1.
```

⇨ i = 9 and Main[i] = 9 . ইলিমেন্ট 9 এর অবস্থান হবে হবে Val এ্যারে মাঝখানে তাহলে আপডেট গুলো হবে –

```
Main [] = [ 3 4 -1 5 8 2 3 12 7 9 10 ]

Temp []  = [ 2 5 6 8 9 0 0 0 0 0 0 ]

Res [] = [ -1 0 -1 1 3 2 5 4 6 8 -1] .

Val [] = [ -1 2 3 7 9 ]
```

```
Length = 1+1+1+1.
```

⇨ Finally, i = 10 and Main[i] = 10 . ইলিমেন্ট 10 এর অবস্থান হবে ডানদিকে Val এ্যারে এর বাইরে । যার মানে হল এটি Val এ্যারে তে থাকা এই পর্যন্ত সবচাইতে বড় ইলিমেন্ট । তার মানে নতুন একটা ইলিমেন্ট LIS এ যোগ হবে এবং LIS Length ও 1 বাড়বে ।

```
Main [] = [ 3 4 -1 5 8 2 3 12 7 9 10 ]

Temp []  = [ 2 5 6 8 9 10 0 0 0 0 0 ]

Res [] = [ -1 0 -1 1 3 2 5 4 6 8 9] .

Val [] = [ -1 2 3 7 9 10 ]

Length = 1+1+1+1+1.
```

তাহলে আমাদের LIS এর Length হবে Length = 5 + 1 = 6 । কেন 1 যোগ করলাম ? কারণ প্রতিটা সঙ্খ্যা এককভাবে একটি LIS । আমরা কিন্তু যখন LIS এর ২য় ইলিমেন্ট টি পেয়েছিলাম তখন থেকেই LIS কাউন্টিং শুরু করেছিলাম । এজন্য শেষে এসে 1 যোগ করে দিয়েছি । চাইলে আমরা শুরুতেই Length = 1 ধরে নিতে পারি ।

So , এটাই হচ্ছে কিভাবে আমাদের LIS এর Length বের করা যায় তার এ্যালগরিদম ।
এই যে আমরা প্রতিটা স্টেপ এ বলছি সার্চ করার কথা । মানে প্রতিটা ভ্যালু আমরা সার্চ করে দেখে নিচ্ছি এটা কোথায় বসবে এই সার্চ টা আমাদের Binary Search দিয়ে করা লাগবে । কেন বাইনারী সার্চ দিয়ে করব ? কারন কমপ্লেক্সিটি log2(N) এবং আমাদের Val এ্যারে টা সব সময় সর্টেড থাকবে । এজন্যে আগেই Binary Search সম্পর্কে শিখে নিতে বলেছি ।

LIS PATH PRINT

যদি আমরা LIS এর ইলিমেন্ট গুলো বের করতে চাই তাহলে আমাদের যেটা করতে হবে সেটা হল ।

প্রথমে আমরা একটা Variable নিলাম index . এই index ভ্যারিয়েবল টা আমরা Temp এ্যারে এর Length (LIS এর Length টা) তম পজিশনের ভ্যালু টা দ্বারা ইনিশিয়ালাইজ করলাম । মানে , index = Temp[Length] । এরপর একটা ভেক্টরে (Vector) ততক্ষন পর্যন্ত ভ্যালু Main[Res[Index]] পুশ (Push) করতে থাকবো যতক্ষন না পর্যন্ত আমাদের Res[index] == -1 হয় । একটা স্যুডো দেখানো যায় –

```
Index <- Temp[Length – 1]; // Length-1 যেহেতু আমি ইন্ডেক্সিং টা 0 থেকে শুরু করেছি
Path [];
Path <- Main[index];
While(Res[index] !=-1){
     Path <- Main[Result[index]];
     index <- Result[index];
}
```

এভাবেই আমরা কোন একটা এ্যারে এর LIS বের করতে পারি । সম্পূর্ন LIS লিস্ট টা প্রিন্ট করার জন্য আমাকে ভেক্টরের উলটো দিক থেকে প্রিন্ট করতে হবে ।

**** এ্যালগরিদম এর কমপ্লেক্সিটি O(NlgN)

সম্পূর্ন কোড পাওয়া যাবেঃ এখানে

ধন্যবাদ ধৈর্য্য ধরে পড়ার জন্য । আশা করছি ভবিষ্যতে আরো লিখতে পারবো ।