@python_walker が2020年10月29日に更新

最長増加部分列(LIS)の長さを求める

Python, アルゴリズム, 競技プログラミング

この記事の目的

最近、競技プログラミングに手を出し始めました。競プロに関してはほとんどの素人なので難しいことは何もわかりません。そんな状態ですので、さっそくLISの長さを求めるアルゴリズムで躓きました。LISは、競プロをやっている人にとっては常識で、超有名問題らしいのですが、理解するのにだいぶ苦労しました。なので私の理解したことをここに書き留めておきます。

なにぶん競プロ初心者なので、初歩的なこともまだ十分に分かっておらず、説明が長ったらしくなってしまっていますがご容赦ください。なので手っ取り早くコードだけ知りたいという人はほかのサイトをあたるか、一気にページの一番下に進んでください。Python3での実装を示しています。

どこがわからなかったのか

最初にわからなくなったときに、蟻本やWebでアルゴリズムを調べました。これらに載っているのはほとんど同じ内容で、

1. 数列の長さと対応する長さの数列の最終要素の最小値を格納するためのDPテーブルを作る(INFで初期化)
2. n番目までの数列で作れる最長の増加部分列の最終要素がDPテーブルの数字より小さければ値を更新する
3. できたテーブルで最初にINFでない最大の長さが求める長さ

このなかで、1で「数列の最終要素の最小値」を格納する理由については、昇順の数列をなるべく長くしたいから最終要素はなるべく小さいほうがその次に値を追加しやすいんだろうなという感じで何となく理解できました。

しかし、問題は2でした。どうやってテーブルを更新するの？**二分探索**で更新するとこ見つけるとか書いてあるけど、どういうこと？

LISのアルゴリズム

問題を整理するために、最初に短い数列でLISの長さをアルゴリズム通りに手で求めてみます。操作自体は理解している人はここは飛ばしてよいと思います。数列「1 3 5 2 4 6」を使ってみます。作れる部分列の長さの最大値は6なので、DPテーブルは6列用意します。手順通り、最初はINFで初期化しておきます。

増加部分列の長さ	**1**	**2**	**3**	**4**	**5**	**6**
数列の最終要素の最小値	INF	INF	INF	INF	INF	INF

最初に数列の一番最初の要素だけを使って部分列を作ります。1で作れる部分列は1だけです。長さは1で最終要素も1なので、DPテーブルは、

増加部分列の長さ	**1**	**2**	**3**	**4**	**5**	**6**
数列の最終要素の最小値	**1**	INF	INF	INF	INF	INF

と更新します。次に2番目までの要素で増加部分列を作ります。作れる部分列は
「1」、「3」、「1 3」
長さ1の部分列は2つできて、そのうち最終要素の最小値は1でDPテーブルの値と同じなので更新はなしです。「1　3」は長さ2の部分列です。最終要素は3ですが、DPテーブルはINFなので更新します。

増加部分列の長さ	**1**	**2**	**3**	**4**	**5**	**6**
数列の最終要素の最小値	1	**3**	INF	INF	INF	INF

次は「1, 3, 5」です。これで作れる増加列は
「1」「3」「5」「1, 3」「1, 5」「3, 5」「1, 3, 5」
ですので、上と同じように更新すると

増加部分列の長さ	**1**	**2**	**3**	**4**	**5**	**6**
数列の最終要素の最小値	1	3	**5**	INF	INF	INF

次は「1, 3, 5, 2」で作れる増加部分列は
「1」「3」「5」「2」「1, 3」「1, 5」「3, 5」**「1, 2」**「1, 3, 5」

なので

増加部分列の長さ	**1**	**2**	**3**	**4**	**5**	**6**
数列の最終要素の最小値	1	**2**	5	INF	INF	INF

次は「1, 3, 5, 2, 4」で作れる増加部分列は
「1」「3」「5」「2」「4」「1, 3」「1, 5」「3, 4」「3, 5」「1, 2」「1, 4」「1, 3, 5」**「1, 2, 4」**
なので

増加部分列の長さ	**1**	**2**	**3**	**4**	**5**	**6**
数列の最終要素の最小値	1	2	**4**	INF	INF	INF

次は「1, 3, 5, 2, 4, 6」で作れる増加部分列は
「1」「3」「5」「2」「4」「6」「1, 3」「1, 5」「3, 4」「3, 5」「1, 2」「1, 4」「1, 6」「1, 3, 5」「1 3 6」「1, 2, 4」
「1, 5, 6」「3, 4, 6」**「1, 2, 4, 6」**
なので

増加部分列の長さ	**1**	**2**	**3**	**4**	**5**	**6**
数列の最終要素の最小値	1	2	4	**6**	INF	INF

できたテーブルを見ると増加部分列の長さが5であるところからINFが始まっています。なのでLISの長さは4であることがわかります。

ここからどうするか

さて、長さ6の数列を使ってLISアルゴリズムを手で試してみました。ここから問題になるのが**今やったことをどうプログラムで表現するか**です。さて、どこに着目してプログラムを書き起こしますか？

ある要素までの数字を使って作れる増加列を列挙してテーブルの更新箇所を探す部分に着目してしまうと沼です（私と同じです）。着目すべきなのはテーブルです。最終的なテーブルの下段に着目すると、数字が昇順になっていることがわかります。これは偶然でしょうか？これが偶然出ないのなら、操作で、考える部分列の長さを1増やしてテーブルを更新する作業は、部分列を列挙することでは全くなく、**昇順が崩れないように新たに加わった数字をDPテーブルに入れ込むことです**。少し言い方が抽象的になってしまったので、例を提示します。今、DPテーブルが次のような場合であったとしましょう。

増加部分列の長さ	**1**	**2**	**3**	**4**	**5**	**6**
数列の最終要素の最小値	1	4	6	INF	INF	INF

次に2（長さを1増やすために追加した要素が2であったと仮定）を加えて考えます。今までだったら、上の操作でしたら作れる増加部分列を全部列挙してDPテーブルの更新箇所を探しました。しかし、DPテーブルの下段が昇順に並んでいるという説が正しいのならば、更新は次のようになるべきです。

増加部分列の長さ	**1**	**2**	**3**	**4**	**5**	**6**
数列の最終要素の最小値	1	**2**	6	INF	INF	INF

ほかの部分に2を入れると下段の単調増加性は崩れてしまいます。操作の実態がこのようなものであれば実装は簡単です。**二分探索**で更新箇所を探すことができます。

そもそも仮説は正しいのか

上で考えた仮説は正しいのでしょうか。これを確かめるためには、状況をもう少し抽象化してみます。ちょっとややこしい感じになりますが、見た目ほど複雑ではないので許してください。今考える数列をa_nとして、a_1からa_iまでの要素までででDPテーブルを埋めた状態を考えます。DPテーブルは長さjのところまで埋まっているとします。

数列の最終要素の最小値	a'_1	a'_2	...	a'_{j-1}	a'_j	INF	...

$a'_1, a'_2, \ldots, a'_j$は$a_1, a_2, \ldots, a_i$のどれかに対応します。また、$a'_1, a'_2, \ldots, a'_j$は単調増加であると仮定します（この仮定は帰納法を使うための仮定です）。

次に$a_1 \sim a_{i+1}$までの数列で増加部分列を見つけます。いくつかの状況が考えられるでしょう。

(1) $a'_j < a_{i+1}$のとき

この時が一番簡単で、テーブルでjのところがa'_jで埋まっているということは最終要素がa'_jで長さjの増加部分列があるということを言っているので、その部分列の最後にa_{i+1}を加えてやれば長さ$j+1$の増加部分列の出来上がりです。つまりテーブルの更新は

増加部分列の長さ	**1**	**2**	**...**	$j-1$	j	$j+1$	**...**
数列の最終要素の最小値	a'_1	a'_2	...	a'_{j-1}	a'_j	a_{i+1}	...

(2) $a'_j > a_{i+1}$のとき
この場合は1ずつ前の要素を見ていって、a_{i+1}がある長さkのところで$a'_k < a_{i+1}$になったとします。状況としては、

$$a'_k < a_{i+1} < a'_{k+1}$$

といった感じになっています。次に何をやるかというと、a_{i+1}とa'_{k+1}に着目します。今、$a_{i+1} < a'_{k+1}$なので、最終要素がa'_{k+1}で長さ$k+1$の増加部分列を持ってきて、最終要素をa_{i+1}に差し替えてやれば、長さは$k+1$のままで最終要素の大きさを前より小さくできます。なのでDPテーブルは次のように更新されます。

増加部分列の長さ	**1**	**...**	$k+1$	**...**	j	$j+1$	**...**
数列の最終要素の最小値	a'_1	...	a_{i+1}	...	a'_j	INF	...

2とおりの更新状況が考えられましたが、どちらの更新を行ってもテーブルの単調増加性は失われないことが見て取れます。つまり帰納法により、数列の最終要素の最小値の列は単調増加列になることが言えました。

実装する

やっとここまでたどり着きました。実装はPythonで行ってみます。

tanithonistaさんのご指摘を受け修正しました(2019/2/13)
shakayamiさんのご指摘を受け修正しました(2019/2/23)

```
#リストseqからLISの長さを出す

import bisect

LIS = [seq[0]]
for i in range(len(seq)):
    if seq[i] > LIS[-1]:
        LIS.append(seq[i])
    else:
        LIS[bisect.bisect_left(LIS, seq[i])] = seq[i]

print(len(LIS))
```

二分探索で更新箇所を探すのはPythonの標準ライブラリの `bisect` を使っています。

```
bisect.bisect_left(LIS, seq[i])
```

の部分は、 `LIS` の中に `seq[i]` を入れる場合に `seq[i]` が入るべきインデックスを返します。

また、Pythonでは簡単に配列の長さを変えられるので、必要な長さの配列を用意して最初に十分大きな数字で初期化することはせず、必要に応じて配列の長さを長くしています。その結果答えは配列自身の長さを返すだけで大丈夫になっています。

@python_walker

趣味でプログラミングをしています。未熟者ではありますがよろしくお願いします。

フォロー

関連記事 Recommended by

組み合わせ最適化パターン別解法
by ta-ka

LIS でも大活躍！ DP の配列使いまわしテクニックを特集
by drken

蟻本をPythonで (初級編)
by saba

next_permutationがイマイチよくわからなかったのでまとめてみた
by Nikkely

『Qiitaユーザーが選ぶ、2019年に読んで良かった技術書』アンケート結果発表
PR Qiita Zine

自社開発AIで売上4倍。100年食堂のデータ経営改革
PR ビズヒント

この記事は以下の記事からリンクされています

あのアルゴリズムはどこ？　Pythonを使用してAtCoderの緑色や水色を目指す方に、30以上のアルゴリズムスニペットと100問以上の問題（ACコード付き）を紹介！

からリンク 4 months ago

 **ナップサックに荷物を詰めまくる** からリンク 1 year ago

 **レッドコーダーが教える、競プロ・AtCoder上達のガイドライン【中級編：目指せ水色コーダー！】** からリンク 1 year ago

 **最長増加部分列長を二分探索を使わずに高速で求めるお話** からリンク 2 years ago

コメント

 @tanithonista 2019-02-13 04:25

こんにちは．LISの長さを求めるアルゴリズムのPython実装部分ですが

```
for i in range(len(seq)):
    if seq[i] < LIS[-1]:
        ...
```

となっていますが，正しくは

```
for i in range(len(seq)):
    if seq[i] > LIS[-1]:
        ...
```

ではないでしょうか？
検討のほどよろしくお願いします．

 0

 @python_walker 2019-02-13 13:46

ご指摘ありがとうございます。その通りですね、修正いたしました。

LGTM 1

@shakayami 2019-02-22 20:54

こんにちは。LISのサイズを出力する部分ですが

```
print(len(seq))
```

となっていますが、正しくは

```
print(len(LIS))
```

ではないでしょうか？
検討のほどよろしくお願いします。

LGTM 0

 @python_walker 2019-02-23 12:54

ご指摘ありがとうございます。修正しました

LGTM 0

@7iva

2019-03-21 17:31

こんにちは。小さな事ですが、LISのアルゴリズムの説明において

> と更新します。次に2番目までの要素で増加部分列を作ります。作れる部分列は
> 「1」、「1 3」、「1 3」
> 長さ1の部分列は2つできて、そのうち最終要素の最小値は1でDPテーブルの値と同じなので更新はなしです。

となっていますが、
「1」、「3」、「1 3」
なのではないでしょうか
検討よろしくおねがいします。

LGTM 0

@python_walker

2019-03-21 19:33

ご指摘ありがとうございます。誤植ですね、修正いたしました。

LGTM 0

@yaok

2019-07-21 12:46

初めまして、私も最近競プロを始めた者です。LISに関してよく分かる良記事だと感じました、ありがとうございます。

一つ重箱の隅をつつくようですが、後半単調増加の証明の(2), k及びk+1添字の項はダッシュ付きじゃないかな？と思いました。勘違いでしたらすみません、ご検討お願い致します。

0

 @minomonter 2020-01-25 19:56

初めまして。最近競プロを初めてこの記事に行き着きました。LISの説明に特化していてとてもわかりやすかったです。ありがとうございます。

細かいことなのですが、章「LISのアルゴリズム」で例として出されている数列「1 3 5 2 4 6」は、部分列の長さとして登場する「1 2 3 4 5 6」という数字と一致していて紛らわしいので「11 13 15 12 14 16」など別の数字に置き換えると混乱することなくすんなり理解できると感じました。修正のご検討をよろしくお願いします。

0

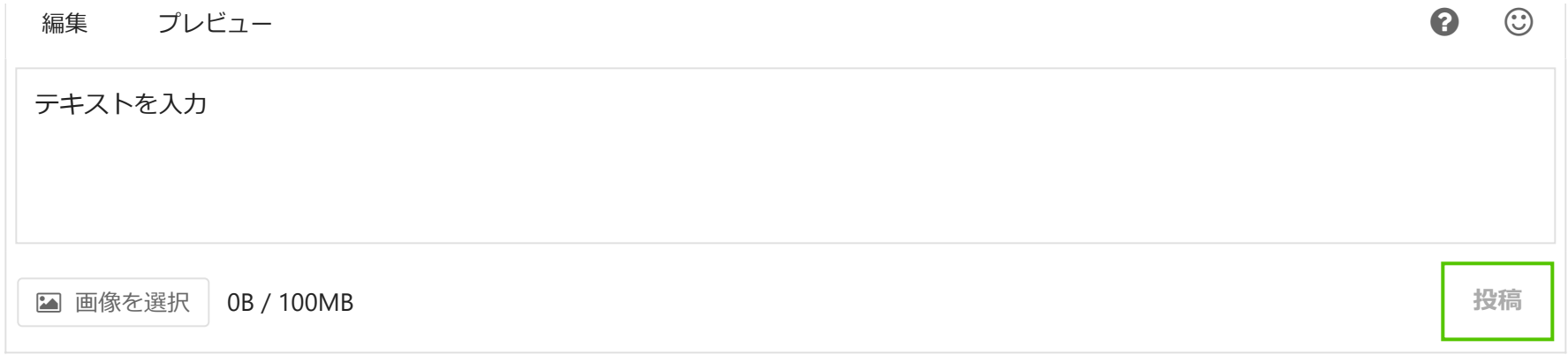
 @python_walker 2020-10-29 18:27

@mine691 ありがとうございます。

' を忘れている部分を修正。kはLIS配列のインデックスなので、その要素には ' を付けるべきだと思う。 by mine691 2020/10/26 16:02

0

 **投稿する**



How developers code is here.