**AtCoder ABC164レポート**

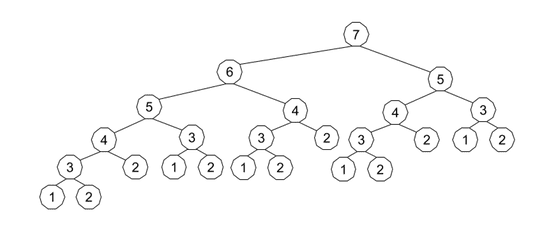
**用語解説**

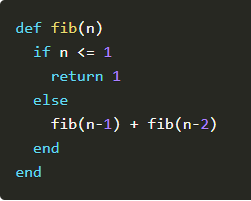
・DP（動的計画法）

動的計画法は大きく分けて2種類のものがある。1つはトップダウン（メモ化再帰）、２つ目はボトムアップ（分割統治法）。

1. トップダウン（メモ化再帰）

n番目のフィボナッチ数列を考える

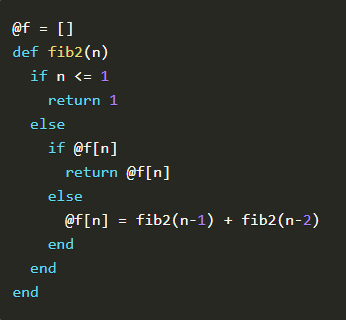




（全探索）

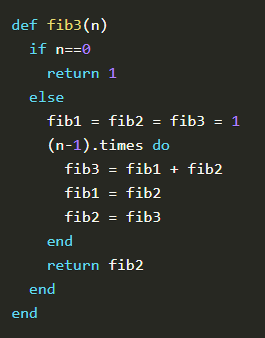
このコードでは例えば７番目の数を出す際にはその手前の６番目と５番目の数を指す必要がある。６番目の数を出すには５番目と４番目が必要。５番目を出すには４番目を３番目が必要。と順番に全ての必要な数を繰り返し計算している。つまり2＾n回分の計算が必要になる。

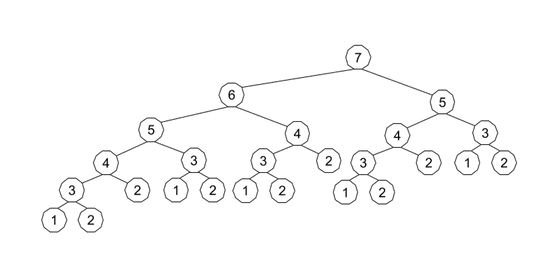
メモ化再帰で考えると…



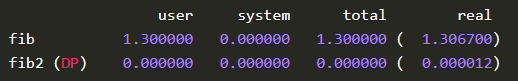
計算した結果を記録（メモ）しておいて、同じ数に関しては１度しか計算しないようにする方法。こうすることにより計算量がグッと減る。

1. ボトムアップ（分割統治法）





１（下）から順に足していく方法。実際に紙に書いてみるとわかりやすい。





ベンチマークしてみるとこの通りです。全探索との差は一目瞭然。

今回のABC164のD問題の解説にDPという見慣れない言葉があったので調べてみました。計算量を考える上で重要な知識なので、しっかり身に着けておきましょう。

参考にしたサイト

<https://www.jabba.cloud/20161020172918/>

練習問題

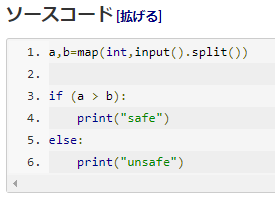
<https://qiita.com/drken/items/dc53c683d6de8aeacf5a>

精進していきましょう！

**ABC164　泰山の回答**



**泰山の解答**

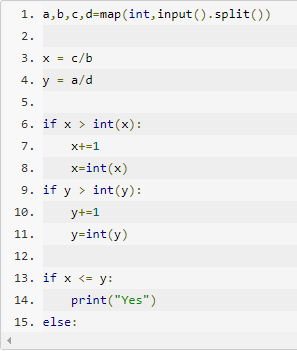


超簡単ですね。WAを出さないようにチェックしてから提出しましょう。

**B - Battle**



**泰山の解答**

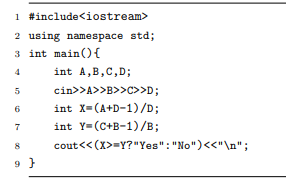




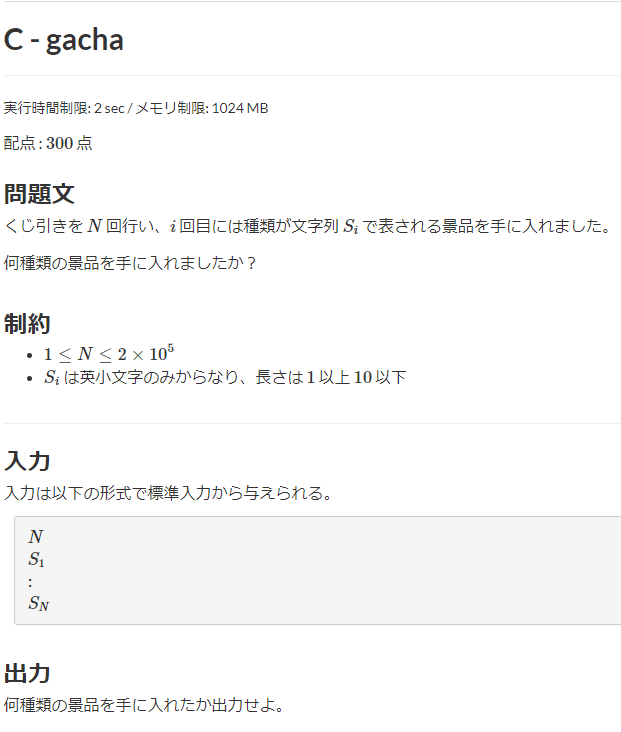
**解答の方針**

まず、ｘ＝ｙのときは先行の高橋君が勝つということに注意する。体力を相手の攻撃力で割ったとき、倒すのに必要なターン数はint(x)+1回なので、割り切れなかったときはプラス１する（６～１１行目）。そして最後に比較して勝敗を判断する。

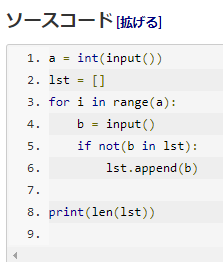
**模範解答**



C++ではint型同士の割り算は小数点が出ても切り捨てられるため、このような操作になっているようです。与えられた数値をdouble（C++における少数ありの数値の型）にするには数値に.0をいちいちつけないといけないので、この場合は変換はできません（自分は知識がまだ浅いのでできるかもしれませんが）。できたとしても、この式で通した方が行数も少なくすっきりしますね。



**泰山の解答**



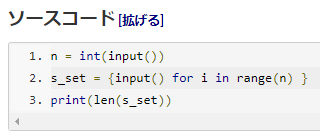


**解答の方針**

まず空のリストを作っておく。その次にinputされた文字列がそのリストにあるかないかを判定する（5行目）。もしなければ、そのリストに入れ、ないなら次の入力に移る。こうすることで同じ文字列がないリストが作成され、最後にlen()で大きさを出力する。

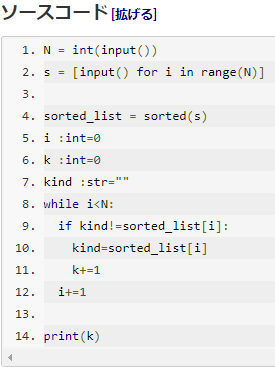
今回は計算量オーバーで不正解となった。5行目はおそらくリストの中を全探索するコードになっているので、入力の数が多く、あまり同じ文字列がない場合はかなりの計算量になってしまう。

**ほかの人の解答**

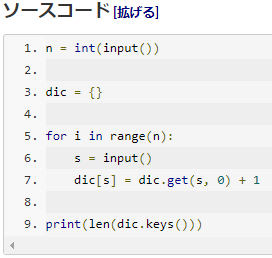


set()は重複するものをまとめてくれる。set(s)でもよいが、この解答のように {} で囲むことによって最初からset型のオブジェクトを作れる。異なる型を要素として持つこともできる。ただし、リスト型のような更新可能なオブジェクトは登録できない。タプルはOK。

また、set型は順序をもたないので、生成時の順序は記憶されない。また、元のリストの順序は保持されない。



ソートしていく解答。setを使わなくても正解できる解答がありました。ソートのやり方もちゃんと身に着けておかないといけないですね。sorted()の計算量はO(nlogn)（←おそらく中身はクイックソート）で、ifの中身はO(n)なので大体の計算量はO(nlogn)です。このことから、許容範囲の計算量になっていることがわかります。



これは辞書を使った方法。コードの意味がまだ分かっておりません。誰か教えてください。



**他の人の解答**

