

AtCoder Beginner Contest 006

解説



AtCoder株式会社 代表取締役
高橋 直大

- 競技プログラミングをやったことがない人へ
 - まずはこっちのスライドを見よう！
 - <http://www.slideshare.net/chokudai/abc004>

A問題 世界のFIZZBUZZ

1. 問題概要
2. アルゴリズム

- 1ケタの数字Nが与えられる
- Nが3の倍数、または3が含まれるはYES、そうでないならNOと出力しなさい

- 基本的なプログラムの流れ
 - 標準入力から、必要な入力を受け取る
 - 今回の場合は、 n という1つの整数
 - 問題で与えられた処理を行う
 - 今回は、3で割り切れる、または、3を含むかを調べる
 - 標準出力へ、答えを出力する

- 入力

- 1つの数字を、標準入力から受け取る

- Cであれば、`scanf("%d", &n);` など
 - C++であれば、`cin >> n;`
 - 入力の受け取り方は、下記の練習問題に記載があります。
 - http://practice.contest.atcoder.jp/tasks/practice_1

- 整数 n が、以下の条件を満たすかどうか判定する
 - 3で割り切れるかどうか
 - 文字3を含むかどうか
 - これは、if文と、`||`などの論理演算子を組み合わせることで表記できる。
 - `if((3で割り切れる) || (文字3を含む)) print("YES");`
- もし条件を満たすのであればYES、満たさないのであればNOを出力する

- 3で割り切れるかどうか
 - 余剰を計算する演算子を使用する
 - 大抵の言語においては% たまにmod
 - $7\%3$ を計算すると、1が出てくる、みたいな使い方が出来る
 - これが0になっているかどうか判定するだけ
- 3を含むかどうか
 - そもそも、今回の問題では、 n が1ケタ
 - 3を含む n は、 $n=3$ の場合のみ
 - これは、3で割り切れる数字でもあるので、上の条件に含まれる
 - よって、考慮しなくても良い。

- 3を含むかどうか
 - もし、 n が何桁もあった場合
 - 13など、3の倍数でないが、3を含む n が存在する
 - やり方は何通りか存在する
 - n を文字列として持ち、文字3を含むか調べる
 - Findなどの、文字列検索を行うアルゴリズムを使う
 - Forループやforeachなどで1文字ずつ調べても良い
 - 1ケタずつ整数として調べる。
 - まず、 $\%10$ を使い、1ケタ目の数字だけ取り出す
 - 次に、それが3であれば終了し、そうでなければ、 $/10$ して次の桁に移行する。

- 1ケタずつ整数として調べる。
 - まず、%10を使い、1ケタ目の数字だけ取り出す
 - 次に、それが3であれば終了し、そうでなければ、/10して次の桁に移行する。
- $1342 \% 10 = 2 \leftarrow 3$ でないので、次の値は10で割って134
- $134 \% 10 = 4 \leftarrow 3$ でないので、次の値は10で割って13
- $13 \% 10 = 3 \leftarrow 3$ なので終了

- 出力

- 求めた答えを、標準出力より出力する。
- 言語によって違います。
 - `printf("YES\n");` (C)
 - `cout << "YES" << endl;` (C++)
 - `System.out.println("YES");` (Java)
 - 各言語の標準出力は、下記の練習問題に記載があります。
 - http://practice.contest.atcoder.jp/tasks/practice_1

B問題 トリボナッチ数列

1. 問題概要
2. アルゴリズム

- 整数 n が与えられる
- トリボナッチ数列の第 n 項を答えなさい
 - ただし、数字が大きくなるので、10007で割った余りを出力しなさい。
- トリボナッチ数列とは、3つ前の数字までを足した数字が、次の数字になる数列の事を言います。
 - 0,0,1,1,2,4,7,13,24...みたいなもの

- 入力
 - 整数 n を受け取る
 - さっきと同じです！

- 処理
 - 4番目から順番に求める
 - 1,2,3番目は、予めコンピュータに入力しておく。
 - やり方は主に2通り
 - 1つ前、2つ前、3つ前の数字を、変数に入れておく
 - $a = 0, b = 0, c = 1$ のような感じ
 - 1巡したら、 $a = 0, b = 1, c = 1$ のようにローテーションさせる
 - 過去の全ての結果を配列に確保してしまう。
 - 入力が100万までなので、長さ100万の配列を確保する
 - 計算するときは、 $ar[n] = ar[n-1] + ar[n-2] + ar[n-3]$ といった感じ

- 注意点

- B問題の答えは、数字が非常に大きくなる！
 - Int型やlong型では収まりません
- 答えるべきものは、10007で割った余り
- この時、途中の計算式でも、10007で割った余りを使って
良い
 - 最後にだけ計算しようとする、途中で桁溢れが起きてしまいます。

- 出力
 - A問題と同じく、答えを出力するだけ
 - `Print(ar[n])`みたいな感じ

- おまけ
 - 再帰関数でやると、計算量が膨大になります。
 - 動的計画法みたいにやりましょう。
 - メモ化再帰でもOK

C問題 スフィンクスのなぞなぞ

1. 問題概要
2. アルゴリズム

- スフィックスがなぞなぞを出します。
- 「この街には人間が N 人いる。人間は、大人、老人、赤ちゃんの 3 通りだ。
この街にいる人間の、足の数の合計は M 本で、大人の足は 2 本、老人の足は 3 本、赤ちゃんの足は 4 本と仮定した場合、存在する人間の組み合わせとしてあり得るものを 1 つ答えよ。」
 - 答えられないと留年する
- N と M が入力されるので、答えを返すプログラムを作りなさい。
 - 答えが存在しない場合は -1 -1 -1 を出力する

- 制約

- 部分点1

- $N \leq 100$
 - $M \leq 500$

- 部分点2

- $N \leq 1500$
 - $M \leq 7500$

- 満点

- $N \leq 100000$
 - $M \leq 500000$

- N, M に対して、
 - $N = a + b + c$
 - $M = 2a + 3b + 4c$
 - $a, b, c \geq 0$
 - なお、大人の人数が a 、老人の人数が b 、赤ちゃんが c とする
- の連立方程式を解かなければならない？

- $N \leq 100$ の時、 a, b, c を全て調べることが可能
 - $a, b, c = \{0, 0, 100\}, \{0, 1, 99\}, \dots, \{100, 0, 0\}$ のように全チェック
 - もしこれで、足の本数が M と一致していたら、それを出力
 - 全ての a, b, c の組み合わせで条件を満たさなかった時は、正しい組み合わせは存在しないので、-1 -1 -1を出力

- $N \leq 1500$ の時、 a, b, c を全て調べることが不可能？
 - 高速に計算できる方法を考える
 - a, b が決まった時、 $c = N - a - b$
 - これを使うと、やはり全列挙が可能となる。
 - 同様に、それぞれの組み合わせに対して、足の本数が一致する組み合わせが存在するか調べるだけ。

• 解法1 つるかめ算

- a,b,cのどれかを決めてしまう。
 - 例えば、 $b=0$ としてみる。
- すると、 $M = 2*a + 4*c$
 - これはただのつるかめ算
- 大人、老人、子供のどれか1つの人数だけ全探索を行い、残りの2つを、つるかめ算により求める。
 - 人数が負になったり、整数にならない場合は失敗
- 成功する組み合わせがあったらそれを出力
- ダメだったら-1 -1 -1を出力

- 解法2 老人の数の固定
 - 老人が2人いた場合
 - 子供1、大人1に変換することが可能
 - つまり、老人が2人以上いる組み合わせは、全て老人が1人以下の組み合わせに変換することが出来る
 - つまり、老人が0人いる場合と、1人いる場合だけ考えれば良い！
 - そこまで分かれば全探索すれば良い

- おまけ

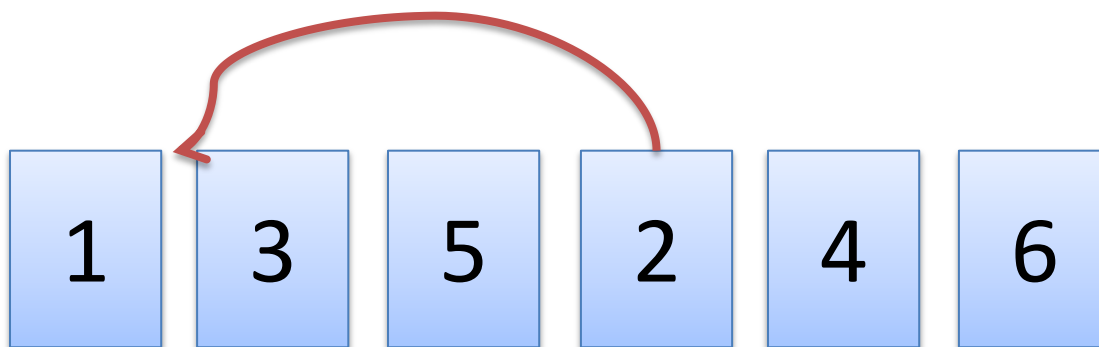
- 解法1と解法2を組み合わせると、満点解法で解けます
- 老人を0,1の2パターンに固定する以外にも、 $O(1)$ で解ける解き方はいくつか存在します。
 - 回答が複数ありますが、偶奇さえきちんと合わせれば、大体つじつまを合わせられます。
- 回答が複数あるので、連立方程式ライブラリとかで解こうとするのは非推奨です。

D問題 トランプ挿入ソート

1. 問題概要
2. アルゴリズム

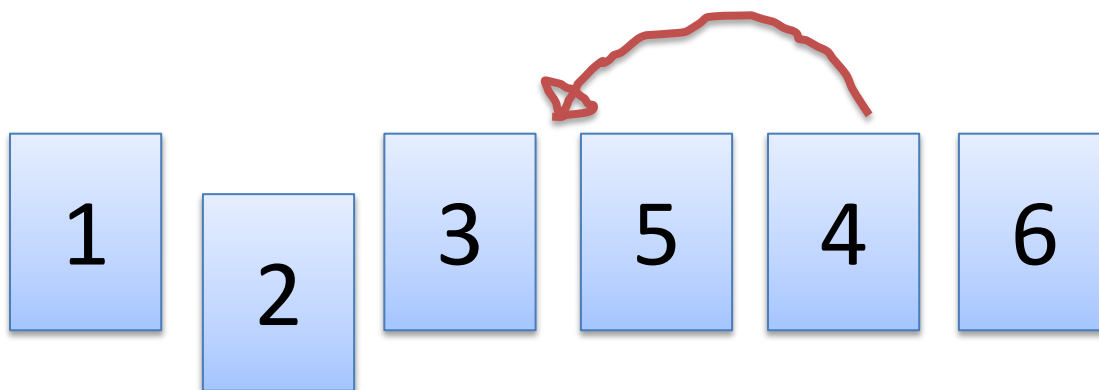
- 1～Nまでの数字が書かれたカードがN枚存在する
 - 山札からカードを1枚抜き取り、任意の場所に挿入することが可能
 - 山札をソートしたい時に、並び替える必要のある最小数を求めなさい
- 実は、多くのコンテストに出題されている、超典型問題です。

- 問題のイメージ
 - 以下のようなカードが与えられる
 - カードを抜いて入れ替える



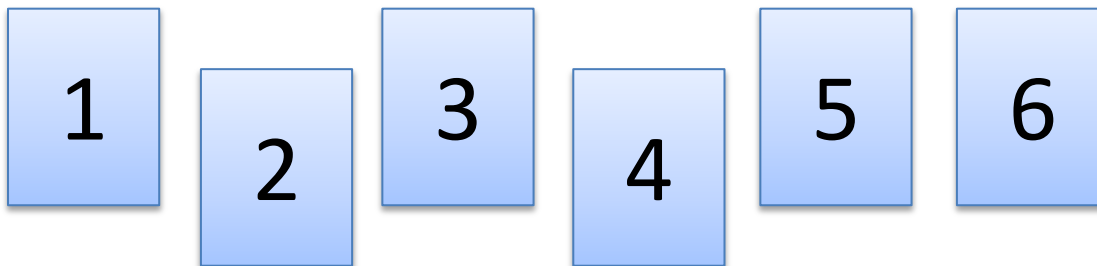
- 問題のイメージ

- 以下のようなカードが与えられる
- カードを抜いて入れ替える
 - これを繰り返してソートさせる



- 問題のイメージ

- 以下のようなカードが与えられる
- カードを抜いて入れ替える
 - これを繰り返してソートさせる
- ソートが完了するまでの最小手数を出力する
 - 今回の場合は2

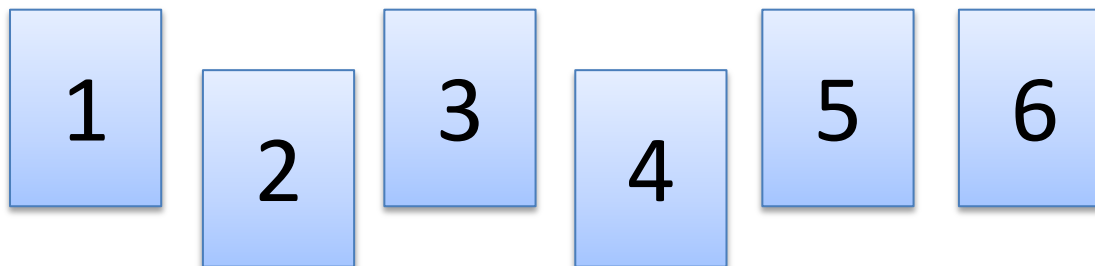


- 制約
 - 部分点1
 - $N \leq 16$
 - 部分点2
 - $N \leq 1000$
 - 満点
 - $N \leq 30000$

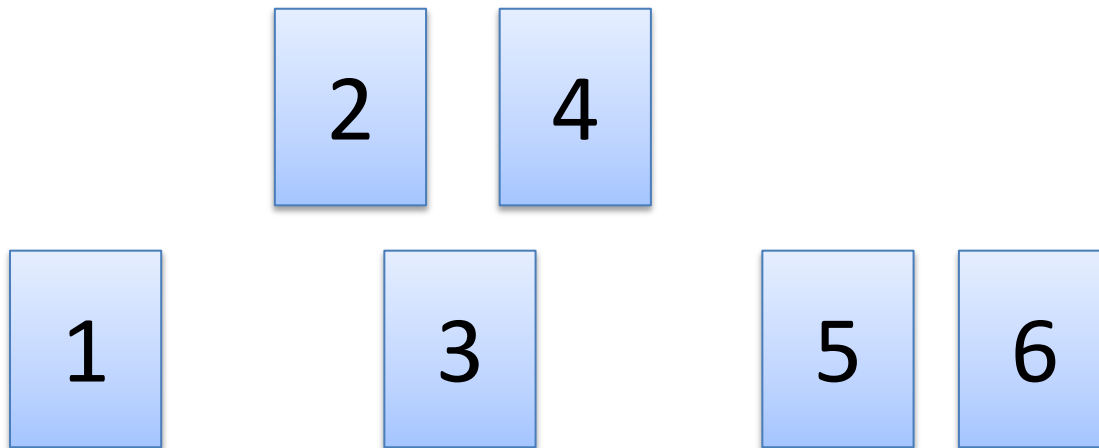
- 考察

- 全ての並び替えパターンは、 $n!$ 通り存在する
 - これを全て考えるのは不可能
- 何か工夫をしなければ、解くことは難しい

- 先ほどのサンプルでは、「2」と「4」のカードを動かすことになった。



- 先ほどのサンプルでは、「2」と「4」のカードを動かすことになった。
 - 1個ずつ動かしたが、図のように一気に複数抜いて、一気に複数差し込んでも良い。



- 先ほどのサンプルでは、「2」と「4」のカードを動かすことになった。
 - 1個ずつ動かしたが、図のように一気に複数抜いて、一気に複数差し込んでも良い。
 - この時、差し込んでソートできる条件は、取り除いた列がソート済みになっていること

1

3

5

6

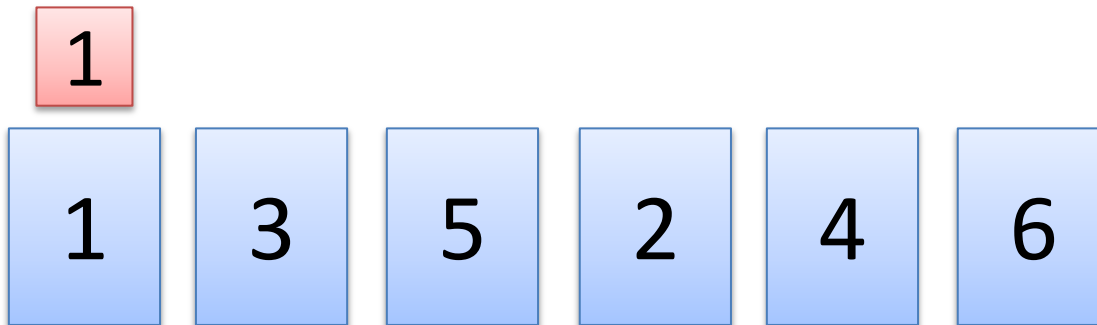
- つまり、「どのカードを残すか」を全探索することによって、その残されたカードがソート済みになっているかどうか調べれば良い。
 - この組み合わせは、 N に対して 2^N 通りしか存在しない。
 - N が16程度であれば間に合う
- 2^N 通りに対し、全列挙を行い、増加列になっているものの中で、最も残す数が多いものを調べれば良い。
 - ABC002 D問題「派閥」と同じ考え方
 - 002のDと同じアルゴリズムが10点でごめんなさい><

- 2^n 全列挙の仕方
 - 深さ優先探索を使う
 - 普通に1つずつ、使う使わないを判定する
 - 自然な実装になりやすい？
 - 整数のbitを用いた探索を使う
 - ABC002と同様
 - 時間がないので後で書いて再アップロードします > <

- さらに計算を早くするには？
 - 残す列がソート済みになっていれば良い、という点に着目する
- 「どのように入れ替えるか」ではなく、「カードを抜いてソートされた状態にすると、残ったカードの数が最大にする方法」を考える

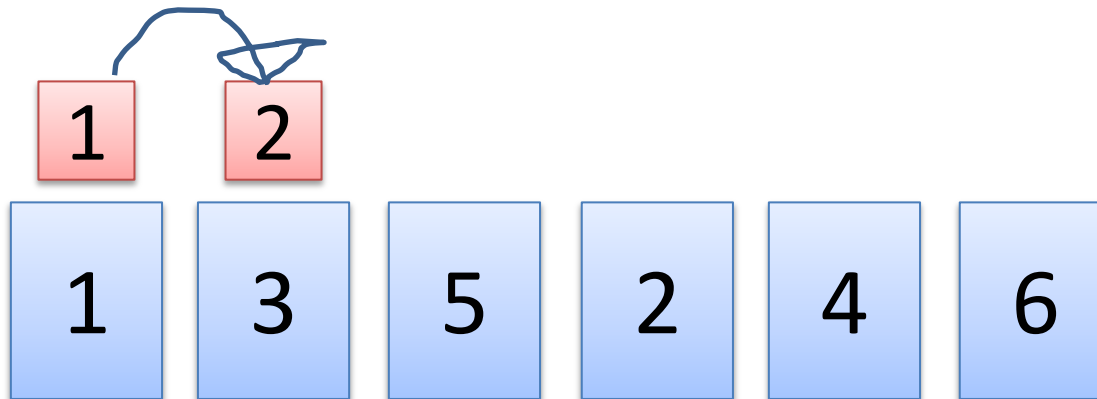
- 増加列を求める方法

- 普通に深さ優先探索をすると前回と同じ
- 動的計画法を使おう！
 - 左から順番に、「最後にそのカードを使った時の、最大の列の長さはいくつか」を計算していく
 - 最初は1



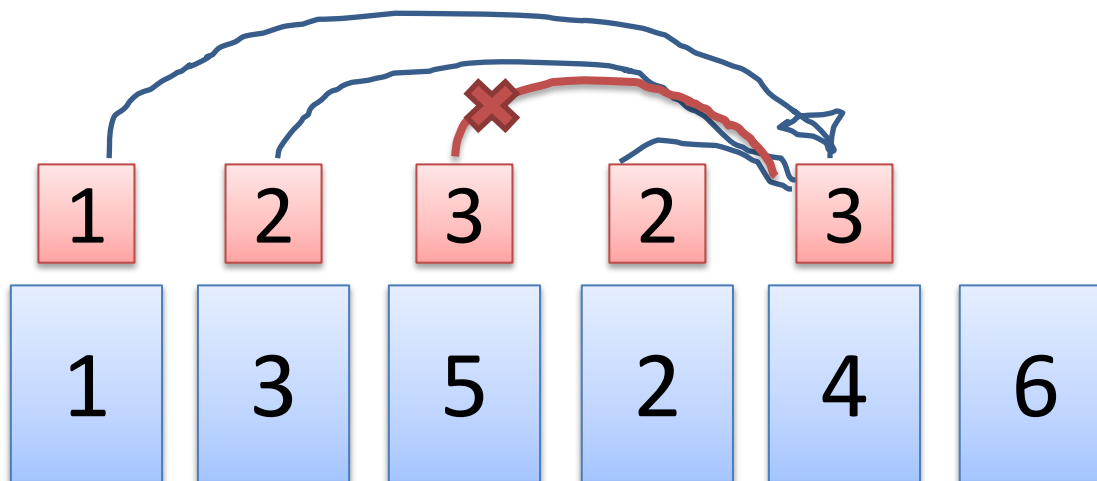
- 増加列を求める方法

- 普通に深さ優先探索をすると前回と同じ
- 動的計画法を使おう！
 - 左から順番に、「最後にそのカードを使った時の、最大の列の長さはいくつか」を計算していく
 - 最初は1
 - 次の値は、前の値を利用して、最大値から計算する



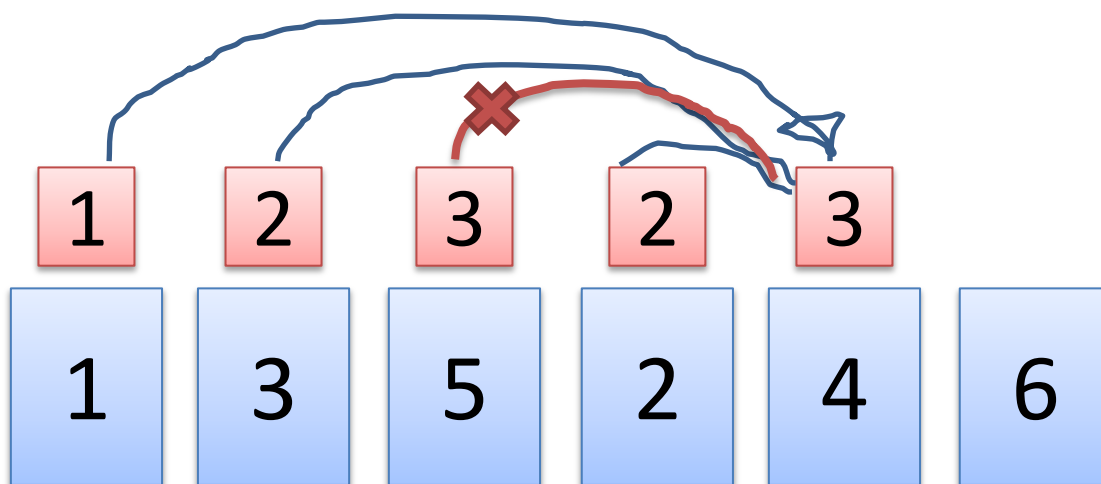
- 増加列を求める方法

- 普通に深さ優先探索をすると前回と同じ
- 動的計画法を使おう！
 - 左から順番に、「最後にそのカードを使った時の、最大の列の長さはいくつか」を計算していく
 - 次の値は、前の値を利用して、最大値から計算する



- 計算量の考察

- 各カードを選ぶ部分が $O(n)$
- そのカードから前のカードの部分列の最大値を選ぶ部分が $O(n)$
- 併せて $O(n^2)$ 1000程度であれば計算可能となる。



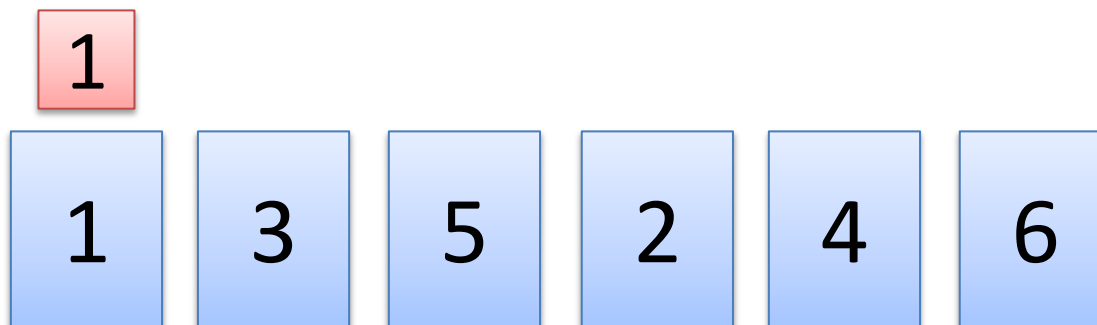
- さらに計算を早くするには？
 - データの持ち方を変えよう！
 - ここまで出てきた中で、k枚の部分列が作れるもののうち、もっともカードの値が小さいものを持つ

増加列	0	1	2	3	4	5
カード	-INF	INF	INF	INF	INF	INF



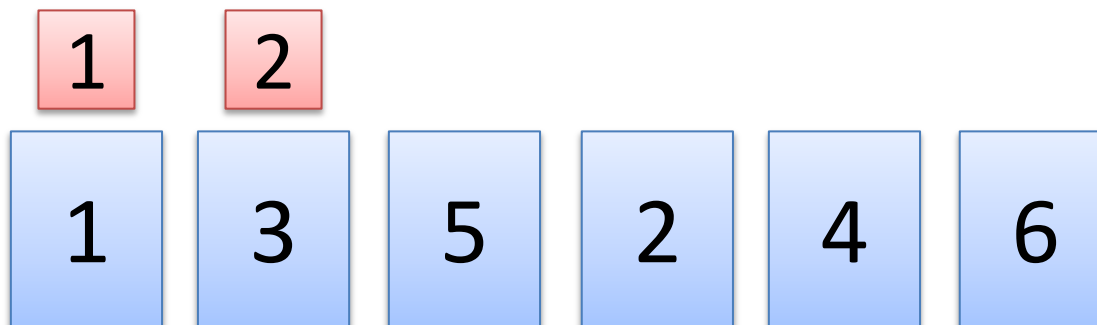
- さらに計算を早くするには？
 - データの持ち方を変えよう！
 - ここまで出てきた中で、k枚の部分列が作れるもののうち、もっともカードの値が小さいものを持つ

増加列	0	1	2	3	4	5
カード	-INF	1	INF	INF	INF	INF



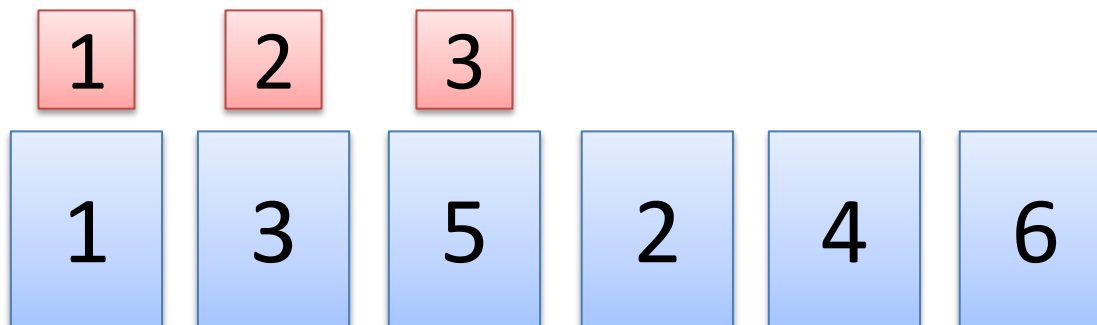
- さらに計算を早くするには？
 - データの持ち方を変えよう！
 - ここまで出てきた中で、k枚の部分列が作れるもののうち、もっともカードの値が小さいものを持つ

増加列	0	1	2	3	4	5
カード	-INF	1	3	INF	INF	INF



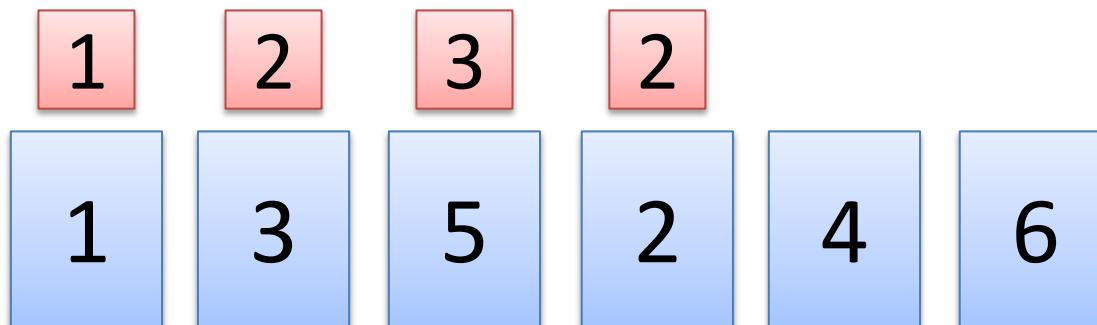
- さらに計算を早くするには？
 - データの持ち方を変えよう！
 - ここまで出てきた中で、k枚の部分列が作れるもののうち、もっともカードの値が小さいものを持つ

増加列	0	1	2	3	4	5
カード	-INF	1	3	5	INF	INF



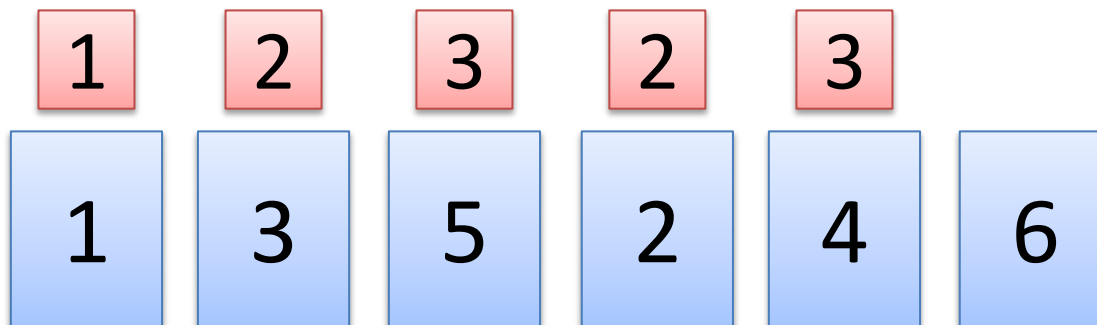
- さらに計算を早くするには？
 - データの持ち方を変えよう！
 - ここまで出てきた中で、k枚の部分列が作れるもののうち、もっともカードの値が小さいものを持つ

増加列	0	1	2	3	4	5
カード	-INF	1	2	5	INF	INF



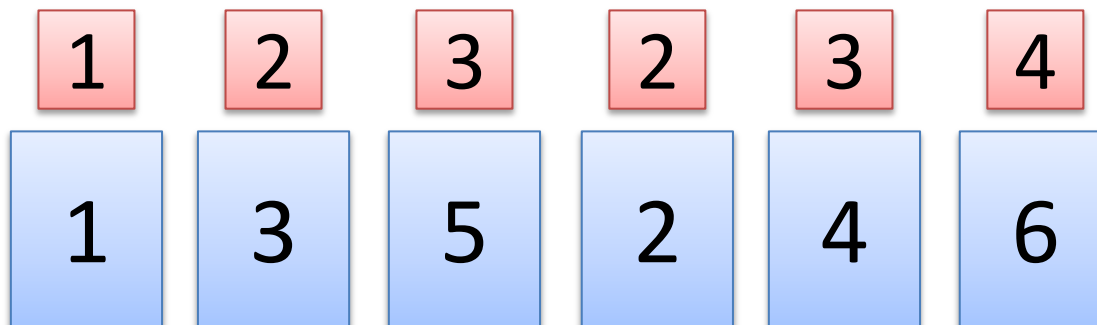
- さらに計算を早くするには？
 - データの持ち方を変えよう！
 - ここまで出てきた中で、k枚の部分列が作れるもののうち、もっともカードの値が小さいものを持つ

増加列	0	1	2	3	4	5
カード	-INF	1	2	4	INF	INF



- さらに計算を早くするには？
 - データの持ち方を変えよう！
 - ここまで出てきた中で、k枚の部分列が作れるもののうち、もっともカードの値が小さいものを持つ

増加列	0	1	2	3	4	5
カード	-INF	1	2	4	6	INF

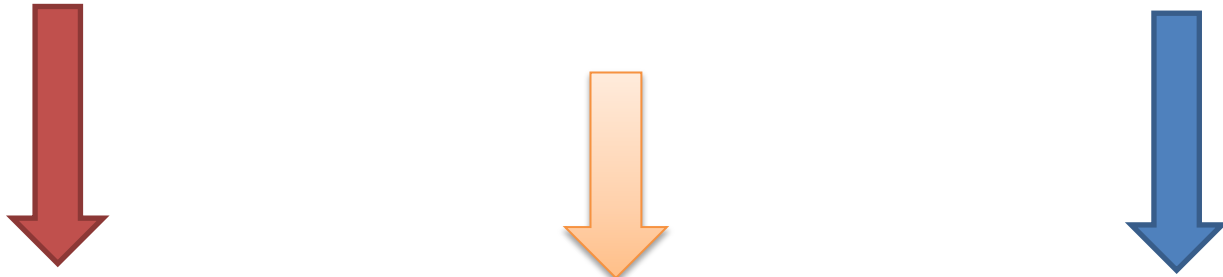


- 考察

- このカードの配列は、絶対に昇順になっている

- よって、足すべき部分は1か所しかなく、二分探索で求めることが可能である

- 下図は、今見ているカードを3だとする



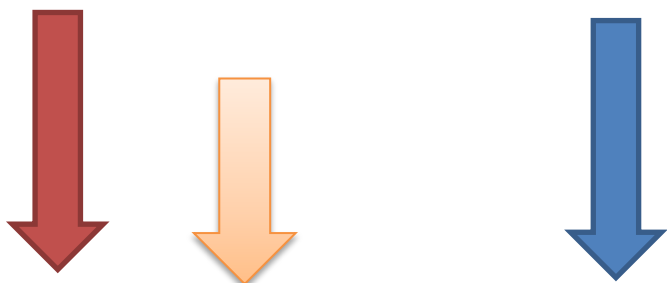
増加列	0	1	2	3	4	5
カード	-INF	1	2	4	6	INF

- 考察

- このカードの配列は、絶対に昇順になっている

- よって、足すべき部分は1か所しかなく、二分探索で求めることが可能である

- 下図は、今見ているカードを3だとする



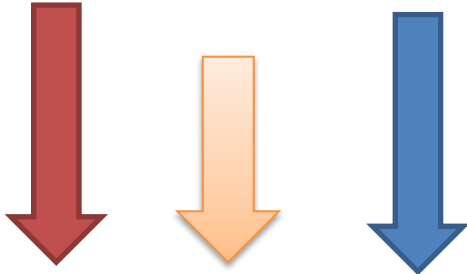
増加列	0	1	2	3	4	5
カード	-INF	1	2	4	6	INF

- 考察

- このカードの配列は、絶対に昇順になっている

- よって、足すべき部分は1か所しかなく、二分探索で求めることが可能である

- 下図は、今見ているカードを3だとする



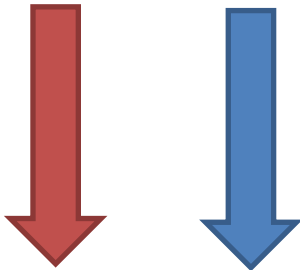
増加列	0	1	2	3	4	5
カード	-INF	1	2	4	6	INF

- 考察

- このカードの配列は、絶対に昇順になっている

- よって、足すべき部分は1か所しかなく、二分探索で求めることが可能である

- 下図は、今見ているカードを3だとする こんな感じで求まる



増加列	0	1	2	3	4	5
カード	-INF	1	2	4	6	INF

- 考察

- このカードの配列は、絶対に昇順になっている
 - よって、足すべき部分は1か所しかなく、二分探索で求めることが可能である
- よって、計算量は $O(n\log n)$ となる。

増加列	0	1	2	3	4	5
カード	-INF	1	2	4	6	INF

- おまけ
 - 最長増加部分列(Longest Increasing Subsequence)と呼ばれる有名なアルゴリズムです。
 - 動的計画法に慣れていれば、知らなくても解ける問題ではあります。