2.3 最大最小

問題

長さ N の整数列 A が与えられます。A の（添字の）異なる 2 要素の差の絶対値の最大値を求めてください。

制約

1. $2 \leq N \leq 100$
2. $1 \leq a_i \leq 10^9 (1 \leq i \leq N)$

```
var N = Int.random(in: 2...100)
var A = Array<Int>()
for _ in 1 ... N{
    A.append(Int.random(in: 1...1000000000))
    //A.append(Int.random(in: 1...100))
}
```

Codes 2.5 初期コード

ヒント

絶対値を計算したいなら abs() 関数を使う。その場合は import Foundation を使う。使わなくてもできる。

2.3.1 解答例

Codes 2.6 解答例

```
//初期コード
var N = Int.random(in: 2...100)
var A = Array<Int>()
for _ in 1 ... N{
    A.append(Int.random(in: 1...1000000000))
    //A.append(Int.random(in: 1...100))
}

//回答コード
print(A)
var (min,max) = (A[0],A[0])
for a in A{
    if a < min{
        min = a
    }
    if a > max{
        max = a
    }
}
print(max-min)
```

2.3.2 ポイント

1. if 文を使って判定する

別解

Codes 2.7 別解

```
//初期コード
var N = Int.random(in: 2...100)
var A = Array<Int>()
for _ in 1 ... N{
    A.append(Int.random(in: 1...1000000000))
    //A.append(Int.random(in: 1...100))
}
//別解
var max2 = 0
//abs()関数を使う
for i in A{
    for j in A{
        1+1
        if max2 < abs(i-j){
            max2 = abs(i-j)
        }
```

```
    }
}

print(max2)
//別解
var max3 = 0
for x in 0 ..< A.count{
    for y in x+1 ..< A.count{
        let num = abs(A[x] - A[y])
        if max3 < num{
            max3 = num
        }
    }
}
print(max3)
```

2.6　三つの数字の足し算

問題

二つの整数 K,S が与えられます。3 つの変数 X,Y,Z があり、$0 \leq X, Y, Z \leq K$ をみたす整数の値をとります。$X + Y + Z = S$ をみたす X,Y,Z への値の割当は何通りありますか。[a]

[a] AtCoder Beginner Contest 051-B

制約

1. $2 \leq K \leq 2500$
2. $0 \leq S \leq 3K$
3. K,S は整数

```
let K = Int.random(in: 2...2500)
let S = Int.random(in: 0...3*K)
```

Codes 2.13　初期コード

ヒント

とにかく全ての X,Y,Z について調べるという戦法でも良いが、時間がかかりすぎる。X と Y を決めれば必要な Z がわかることを利用する。また、回数を表す変数が必要。

2.6.1　解答例

```
let K = Int.random(in: 2...2500)
let S = Int.random(in: 0...3*K)

var caseNum = 0

for X in 0 ... K{
    for Y in 0 ... K{
        let Z = S - X - Y
        if Z >= 0 && Z <= K{
            caseNum += 1
        }
    }
}

print("答えは\(caseNum)通りです。")
```

2.6.2　参考（全探索の方法）

Codes 2.14　全探索

```
let K = Int.random(in: 2...2500)
let S = Int.random(in: 0...3*K)

var caseNum = 0

for X in 0 ... K{
    for Y in 0 ... K{
        for Z in 0 ... K{
            if X + Y + Z == S{
                caseNum += 1
            }
        }
    }
}

print("答えは\(caseNum)通りです。")
```

2.6.3　ポイント

1. X,Y を決めるとあるべき Z を決めることができるが、この Z が 0 以上 K 以下であるかを確認する。
2. for 文を重ねる

2.13 Sum of product of pairs

問題

N 個の整数 $A_1, A_2, ..., A_N$ が与えられます．$1 \leq i < j \leq N$ を満たす全ての $A_i \times A_j$ についての和を $\mathrm{mod}(10^9 + 7)$ で求めてください．[a]

[a] AtCoder Beginner Contest 177-C

制約

1. $2 \leq N \leq 2 \times 10^5$
2. $0 \leq A_i \leq 10^9$
3. 入力は全て整数である

```
let N = Int.random(in: 2...Int(2e5))
var A = Array<Int>()
for _ in 0..<N{
    let Ai = Int.random(in: 0...Int(1e9))
    A.append(Ai)
}
```

Codes 2.27 初期コード

ヒント

特になし

2.14.1 解答例

Codes 2.30 解答例

```
var N = Int.random(in: 2...Int(2e5))
var A = Array<Int>()
for _ in 0..<N{
    let Ai = Int.random(in: 0...Int(1e9))
    A.append(Ai)
}



var sum = 0
var squareSum = 0
for i in 0..<N{
    sum += A[i]
    squareSum += A[i]*A[i]
}

let result = (sum*sum-squareSum)/2 % Int(1e9+7)
```

2.14.2 ポイント

1. 式変形をする

2.16 DFS

問題

数列 $a_1, a_2, ..., a_N$ が与えられます．その中からいくつかを選びます．選んだ項の総和をちょうど K にすることができるか判定してください．K にすることができるならば `yes` そうでなければ `no` を出力してください．

制約

1. $1 \leq N \leq 20$
2. $-10^8 \leq a_i \leq 10^8$
3. $-10^8 \leq K \leq 10^8$
4. パラメタは全て整数である

```
let N = Int.random(in: 1...20)
let K = Int.random(in: -Int(1e8)...Int(1e8))

var a = [Int]()
for _ in 1...N{
    a.append(Int.random(in: -Int(1e8)...Int(1e8)))
}
```

Codes 2.33　初期コード

ヒント

深さ優先探索を使う．

2.16.1 解答例

Codes 2.34 解答例

```
let N = Int.random(in: 1...20)
let K = Int.random(in: -Int(1e8)...Int(1e8))

var a = [Int]()
for _ in 1...N{
    a.append(Int.random(in: -Int(1e8)...Int(1e8)))
}

func dfs(i:Int,sum:Int) -> Bool{
    // N 個のとき sum と K が等しいかを返す
    if i == N { return sum == K }

    // i 番目の項を使用しない場合
    if dfs(i: i+1, sum: sum) { return true }

    // i 番目の項を使用する場合
    if dfs(i: i+1, sum: sum+a[i]){ return true }

    return false
}

print(dfs(i: 0, sum: 0) ? "yes" : "no")
```

2.16.2 ポイント

1. 深さ優先探索は一番初めの状態から辿り着ける全ての状態を生成します．全状態を列挙するのに優れた手法です．

2.17　ゆとりのないフィボナッチ

問題

関数の再帰処理を使用してフィボナッチ数列の第 n 項を生成するプログラムを作成してください．フィボナッチ数列は以下の定義の数列とします．

$$a_n = a_{n-1} + a_{n-2}, a_1 = 1, a_2 = 1$$

制約

1. $1 \leq n \leq 20$
2. 解答は初期コードの**//解答はここに**部分に記す．
3. 解答は 1 行のみで記す
4. パラメタは整数である

```
let n = Int.random(in: 1...20)
func fibo(n:Int) -> Int{
    return // 解答はここに
}
```

Codes 2.35　初期コード

ヒント

三項演算子を用いる．

2.17.1 解答例

Codes 2.36 解答例

```
let n = Int.random(in: 1...20)
func fibo(n:Int) -> Int{
    return n > 2 ? fibo(n: n-1) + fibo(n: n-2) : 1
}
fibo(n: n) // これで目的のものを求めることができる

// このようにするとa_n = a_n-1 + a_n-2となっていることを確認できる
for i in 1...10{
    print(fibo(n: i))
}
```