

# Go Generics: First Impressions

@nikolaydubina  
2022-04-27

# Validation with go-playground

[github.com/go-playground/validator](https://github.com/go-playground/validator)

- Define as struct tags
- Initialize *Validator* instance
- *Validator* constructed with reflection
- Very fast 100s of ns

```
// User contains user information
type User struct {
    FirstName string `validate:"required"`
    LastName  string `validate:"required"`
    Age       uint8  `validate:"gte=0,lte=130"`
    Email     string `validate:"required,email"`
    FavouriteColor string `validate:"iscolor"` // alias for 'hexcolor|rgb|rgba|hsl|hsla'
    Addresses []*Address `validate:"required,dive,required"` // a person can have a home and cottage...
}

// Address houses a users address information
type Address struct {
    Street string `validate:"required"`
    City   string `validate:"required"`
    Planet string `validate:"required"`
    Phone  string `validate:"required"`
}

// use a single instance of Validate, it caches struct info
var validate *validator.Validate

func main() {
    validate = validator.New()

    validateStruct()
    validateVariable()
}
```

# Validation with native Go constructs

[github.com/nikolaydubina/validate](https://github.com/nikolaydubina/validate)

- Fast and small LOC validation
- No reflection
- No gencode
- No tags
- Composable
- Short notation
- How convenient, small LOC, and fast can it be?

```
// Employee is example of struct with validatable fields and nested structure
type Employee struct {
    Name      string
    Age       int
    Color     Color      // custom func Validate()
    Education Education // nested with Validate()
    Salary    float64
    Experience time.Duration
    Birthday  time.Time
    VacationStart time.Time
}

func (s Employee) Validate() error {
    return validate.All(
        validate.OneOf("name", s.Name, "Zeus", "Hera"),
        validate.OneOf("age", s.Age, 35, 55),
        validate.Min("age", s.Age, 10), // same field validated again
        s.Color.Validate(),
        s.Education.Validate(),
        validate.Max("salary", s.Salary, 123.456),
        validate.Max("duration", s.Experience, time.Duration(1)*time.Hour),
        validate.After("birthday", s.Birthday, time.Date(1984, 1, 1, 0, 0, 0, 0, time.UTC)),
        validate.Before("vacation_start", s.VacationStart, time.Date(2024, 1, 1, 0, 0, 0, 0, time.UTC))
    )
}

// Education is another custom struct
type Education struct {
    Duration int
    SchoolName string
}

func (e Education) Validate() error {
    if (e.Duration % 17) == 5 {
        return errors.New("my special error")
    }
    return validate.All(
        validate.Min("", e.Duration, 10),
        validate.OneOf("", e.SchoolName, "KAIST", "Stanford"),
    )
}
```

# Learnings: Go generics work well

.. and barely noticeable

```
func (s Employee) Validate() error {  
    return validate.All(  
        validate.OneOf("name", s.Name, "Zeus", "Hera"),  
        validate.OneOf("age", s.Age, 35, 55),  
        validate.Min("age", s.Age, 10), // same field val.  
        s.Color.Validate()).
```

```
74 type errOneOf[T any] struct {  
75     Name string  
76     Val   T  
77     Options []T  
78 }  
79  
80 func (e errOneOf[T]) Error() string {  
81     return fmt.Sprintf("%s(%v) not in %v", e.Name, e.Val, e.Options)  
82 }  
83  
84 func OneOf[T comparable](name string, v T, options ...T) error {  
85     for _, q := range options {  
86         if v == q {  
87             return nil  
88         }  
89     }  
90     return errOneOf[T]{Name: name, Val: v, Options: options}  
91 }
```

# Learnings: Go generics have many limitations

- Can not convert T1 to T2

*No way to express convertibility*

The design has no way to express convertibility between two different type parameters. For example, there is no way to write this function:

```
// Copy copies values from src to dst, converting them as they go.
// It returns the number of items copied, which is the minimum of
// the lengths of dst and src.
// This implementation is INVALID.
func Copy[T1, T2 any](dst []T1, src []T2) int {
    for i, x := range src {
        if i > len(dst) {
            return i
        }
        dst[i] = T1(x) // INVALID
    }
    return len(src)
}
```

The conversion from type `T2` to type `T1` is invalid, as there is no constraint on either type that permits the conversion. Worse, there is no way to write such a constraint in general. In the particular case where both `T1` and `T2` have limited type sets this function can be written as described earlier when discussing [type conversions using type sets](#). But, for example, there is no way to write a constraint for the case in which `T1` is an interface type and `T2` is a type that implements that interface.

It's worth noting that if `T1` is an interface type then this can be written using a conversion to the empty interface type and a type assertion, but this is, of course, not compile-time type-safe.

```
// Copy copies values from src to dst, converting them as they go.
// It returns the number of items copied, which is the minimum of
// the lengths of dst and src.
func Copy[T1, T2 any](dst []T1, src []T2) int {
    for i, x := range src {
        if i > len(dst) {
            return i
        }
        dst[i] = (interface{})(x).(T1)
    }
    return len(src)
}
```

# Learnings: Go generics have many limitations

- Can not convert T1 to T2
- Can not identify matched replaced type

*# Identifying the matched predeclared type*

The design doesn't provide any way to test the underlying type matched by a `~T` constraint element. Code can test the actual type argument through the somewhat awkward approach of converting to an empty interface type and using a type assertion or a type switch. But that lets code test the actual type argument, which is not the same as the underlying type.

Here is an example that shows the difference.

```
type Float interface {
    ~float32 | ~float64
}

func NewtonSqrt[T Float](v T) T {
    var iterations int
    switch (interface{})(v).(type) {
    case float32:
        iterations = 4
    case float64:
        iterations = 5
    default:
        panic(fmt.Sprintf("unexpected type %T", v))
    }
    // Code omitted.
}

type MyFloat float32

var G = NewtonSqrt(MyFloat(64))
```

This code will panic when initializing `G`, because the type of `v` in the `NewtonSqrt` function will be `MyFloat`, not `float32` or `float64`. What this function actually wants to test is not the type of `v`, but the approximate type that `v` matched in the constraint's type set.

One way to handle this would be to permit writing approximate types in a type switch, as in `case ~float32:`. Such a case would match any type whose underlying type is `float32`. This would be meaningful, and potentially useful, even in type switches outside of generic functions.

# Learnings: Go generics have many limitations

- Can not convert T1 to T2
- Can not identity matched replaced type
- T can not be slice, nor array, not map
  - Can not iterate over it
  - Can not access at index

```
1 package main
2
3 import (
4     "fmt"
5 )
6
7 func Print[T any](s T) {
8     for _, v := range s {
9         fmt.Print(v)
10    }
11 }
12
13 func main() {
14     x := map[string]int{"a": 1}
15     Print(x)
16 }
17
```

```
./prog.go:8:20: cannot range over s (variable of type T constrained by any) (T has no core type)
./prog.go:14:11: undefined: string
```

Go build failed.

# Learnings: Go generics have many limitations

- Can not convert T1 to T2
- Can not identity matched replaced type
- T can not be slice, nor array, not map
- ..and many more

## *Omissions*

We believe that this design covers the basic requirements for generic programming. However, there are a number of programming constructs that are not supported.

- No specialization. There is no way to write multiple versions of a generic function that are designed to work with specific type arguments.
- No metaprogramming. There is no way to write code that is executed at compile time to generate code to be executed at run time.
- No higher level abstraction. There is no way to use a function with type arguments other than to call it or instantiate it. There is no way to use a generic type other than to instantiate it.
- No general type description. In order to use operators in a generic function, constraints list specific types, rather than describing the characteristics that a type must have. This is easy to understand but may be limiting at times.
- No covariance or contravariance of function parameters.
- No operator methods. You can write a generic container that is compile-time type-safe, but you can only access it with ordinary methods, not with syntax like `c[k]`.
- No currying. There is no way to partially instantiate a generic function or type, other than by using a helper function or a wrapper type. All type arguments must be either explicitly passed or inferred at instantiation time.
- No variadic type parameters. There is no support for variadic type parameters, which would permit writing a single generic function that takes different numbers of both type parameters and regular parameters.
- No adaptors. There is no way for a constraint to define adaptors that could be used to support type arguments that do not already implement the constraint, such as, for example, defining an `==` operator in terms of an `Equal` method, or vice-versa.
- No parameterization on non-type values such as constants. This arises most obviously for arrays, where it might sometimes be convenient to write `type Matrix[n int] [n][n]float64`. It might also sometimes be useful to specify significant values for a container type, such as a default value for elements.



# Learnings: wrapping functions slows down

Time increases and mallocs too

```
func (s Employee) Validate() error {
    return validate.All(
        validate.OneOf[string]{Name: "name", Value: s.Name, Values: []string{"Zeus", "Hera"}},
        validate.OneOf[int]{Name: "age", Value: s.Age, Values: []int{35, 55}},
        validate.Min[int]{Name: "age", Value: s.Age, Min: 10}, // same field validated again
        s.Color,
        s.Education,
        validate.Max[float64]{Name: "salary", Value: s.Salary, Max: 123.456},
        validate.Max[time.Duration]{Name: "duration", Value: s.Experience, Max: time.Duration(1000000000)},
        validate.After{Name: "birthday", Value: s.Birthday, Time: time.Date(1984, 1, 1, 0, 0, 0, 0, time.Now())},
        validate.Before{Name: "vacation_start", Value: s.VacationStart, Time: time.Date(2024, 1, 1, 0, 0, 0, 0, time.Now())},
    )
}
```

```
$ go test -bench=. -benchtime=10s -benchmem ./...
```

```
goos: darwin
goarch: amd64
pkg: github.com/nikolaydubina/validate
cpu: VirtualApple @ 2.50GHz
BenchmarkEmployee_Error_Message-10      3579223      3379 ns/op      2761 B/op      62 allocs/op
BenchmarkEmployee_Error-10              9361948      1277 ns/op      1344 B/op      34 allocs/op
BenchmarkEmployee_Success-10            25418672      474 ns/op       552 B/op      14 allocs/op
BenchmarkEmployeeSimple_Error_Message-10 8757170      1364 ns/op      992 B/op      28 allocs/op
BenchmarkEmployeeSimple_Error-10        30418941      394 ns/op       504 B/op      13 allocs/op
BenchmarkEmployeeSimple_Success-10       65194581      184 ns/op       224 B/op       6 allocs/op
BenchmarkEmployeeNoContainers_Error_Message-10 24971338      483 ns/op       280 B/op      10 allocs/op
BenchmarkEmployeeNoContainers_Error-10   72736639      165 ns/op       136 B/op       5 allocs/op
BenchmarkEmployeeNoContainers_Success-10 143333276      83 ns/op        64 B/op       2 allocs/op
PASS
ok      github.com/nikolaydubina/validate      124.950s
```

```
$ go test -bench=. -benchtime=10s -benchmem ./...
```

```
goos: darwin
goarch: amd64
pkg: github.com/nikolaydubina/validate
cpu: VirtualApple @ 2.50GHz
BenchmarkEmployee_Error_Message-10      3744121      3229 ns/op      2376 B/op      56 allocs/op
BenchmarkEmployee_Error-10              12533948      958 ns/op       904 B/op      23 allocs/op
BenchmarkEmployee_Success-10            100000000      115 ns/op        80 B/op       3 allocs/op
BenchmarkEmployeeSimple_Error_Message-10 9488436      1263 ns/op      840 B/op      25 allocs/op
BenchmarkEmployeeSimple_Error-10        44261380      270 ns/op       344 B/op       9 allocs/op
BenchmarkEmployeeSimple_Success-10       243491635      49 ns/op        48 B/op       2 allocs/op
BenchmarkEmployeeNoContainers_Error_Message-10 28089966      427 ns/op       248 B/op       9 allocs/op
BenchmarkEmployeeNoContainers_Error-10   142881793      85 ns/op        88 B/op       3 allocs/op
BenchmarkEmployeeNoContainers_Success-10 1000000000      4 ns/op         0 B/op       0 allocs/op
PASS
ok      github.com/nikolaydubina/validate      120.565s
```

# Learnings: passing functions through maps slows down

Overhead of indirection and map is higher than saving from strings

```
func (s Employee) Validate() error {
    return validate.All(map[string]error{
        "name":      validate.OneOf(s.Name, "Zeus", "Hera"),
        "age":        validate.OneOf(s.Age, 35, 55),
        "age_2":      validate.Min(s.Age, 10), // same field validated again
        "color":      s.Color.Validate(),
        "education":  s.Education.Validate(),
        "salary":     validate.Max(s.Salary, 123.456),
        "duration":   validate.Max(s.Experience, time.Duration(1)*time.Hour),
        "birthday":   validate.After(s.Birthday, time.Date(1984, 1, 1, 0, 0, 0, time.UTC)),
        "vacation_start": validate.Before(s.VacationStart, time.Date(2024, 1, 1, 0, 0, 0, time.UTC)),
    })
}
```

```
goos: darwin
goarch: amd64
pkg: github.com/nikolaydubina/validate
cpu: VirtualApple @ 2.50GHz
BenchmarkEmployee_Error_Message-10      2698065      4359 ns/op      3866 B/op      57 allocs/op
BenchmarkEmployee_Error-10              6993564      1714 ns/op      2058 B/op      21 allocs/op
BenchmarkEmployee_Success-10            14948445      810 ns/op       1329 B/op       7 allocs/op
BenchmarkEmployeeSimple_Error_Message-10 8243392      1460 ns/op      1112 B/op      26 allocs/op
BenchmarkEmployeeSimple_Error-10        33798837      356 ns/op       496 B/op       7 allocs/op
BenchmarkEmployeeSimple_Success-10       66953932      182 ns/op       96 B/op        3 allocs/op
BenchmarkEmployeeNoContainers_Error_Message-10 19754359      600 ns/op       576 B/op      10 allocs/op
BenchmarkEmployeeNoContainers_Error-10   61285670      194 ns/op       368 B/op       3 allocs/op
BenchmarkEmployeeNoContainers_Success-10 123293473      98 ns/op        48 B/op       1 allocs/op
PASS
ok      github.com/nikolaydubina/validate      128.263s
```

```
$ go test -bench=. -benchtime=10s -benchmem ./...
goos: darwin
goarch: amd64
pkg: github.com/nikolaydubina/validate
cpu: VirtualApple @ 2.50GHz
BenchmarkEmployee_Error_Message-10      3744121      3229 ns/op      2376 B/op      56 allocs/op
BenchmarkEmployee_Error-10              12533948      958 ns/op       904 B/op      23 allocs/op
BenchmarkEmployee_Success-10            100000000      115 ns/op       80 B/op       3 allocs/op
BenchmarkEmployeeSimple_Error_Message-10 9488436      1263 ns/op      840 B/op      25 allocs/op
BenchmarkEmployeeSimple_Error-10        44261380      270 ns/op      344 B/op       9 allocs/op
BenchmarkEmployeeSimple_Success-10      243491635      49 ns/op       48 B/op        2 allocs/op
BenchmarkEmployeeNoContainers_Error_Message-10 28089966      427 ns/op      248 B/op       9 allocs/op
BenchmarkEmployeeNoContainers_Error-10  142881793      85 ns/op       88 B/op       3 allocs/op
BenchmarkEmployeeNoContainers_Success-10 1000000000      4 ns/op        0 B/op       0 allocs/op
PASS
ok      github.com/nikolaydubina/validate      120.565s
```

# Learnings: fmt is slow + lazy execution

- Hot-path: error does not happen
  - Avoid mallocs when error not happens
  - Terminate ASAP
- Lazy execution in Go is possible
- fmt is expensive
  - create container needed for rendering
  - invoke render only to when Error() invoked
  - try to render with non-fmt first errors. New is fast

```
74 type errOneOf[T any] struct {
75     Name string
76     Val   T
77     Options []T
78 }
79
80 func (e errOneOf[T]) Error() string {
81     return fmt.Sprintf("%s(%v) not in %v", e.Name, e.Val, e.Options)
82 }
83
84 func OneOf[T comparable](name string, v T, options ...T) error {
85     for _, q := range options {
86         if v == q {
87             return nil
88         }
89     }
90     return errOneOf[T]{Name: name, Val: v, Options: options}
91 }
```

```
$ go test -bench=. -benchtime=10s -benchmem ./...
```

```
goos: darwin
```

```
goarch: amd64
```

```
pkg: github.com/nikolaydubina/validate
```

```
cpu: VirtualApple @ 2.50GHz
```

BenchmarkEmployee_Error_Message-10	3744121	3229 ns/op	2376 B/op	56 allocs/op
BenchmarkEmployee_Error-10	12533948	958 ns/op	904 B/op	23 allocs/op
BenchmarkEmployee_Success-10	100000000	115 ns/op	80 B/op	3 allocs/op
BenchmarkEmployeeSimple_Error_Message-10	9488436	1263 ns/op	840 B/op	25 allocs/op
BenchmarkEmployeeSimple_Error-10	44261380	270 ns/op	344 B/op	9 allocs/op
BenchmarkEmployeeSimple_Success-10	243491635	49 ns/op	48 B/op	2 allocs/op
BenchmarkEmployeeNoContainers_Error_Message-10	28089966	427 ns/op	248 B/op	9 allocs/op
BenchmarkEmployeeNoContainers_Error-10	142881793	85 ns/op	88 B/op	3 allocs/op
BenchmarkEmployeeNoContainers_Success-10	100000000	4 ns/op	0 B/op	0 allocs/op

```
PASS
```

```
ok      github.com/nikolaydubina/validate
```

```
120.565s
```

More benchmarking errors

<https://github.com/nikolaydubina/go-bench-errors>

# Final outcome?

- Used generics only for function parameters
- Functions instead of interfaces
- Fairly fast
- ~90LOC ok syntax
- Generics do reduce duplication for basic types
- Generics are not as flexible

# Official Go recommendation

“..should avoid type parameters until you notice that you are about to write the exact same code multiple times.”

- Don't replace interface types with type parameters
- Don't use type parameters if method implementations differ

Ref. <https://go.dev/blog/when-generics>

# Other mechanisms: go:generate and tags

<https://github.com/nikolaydubina/go-featureprocessing>

- Generate in-place indexing for invocations
- For speed + json tags

```
//go:generate go run github.com/nikolaydubina/go-featureprocessing/cmd/generate -struct=Employee
```

```
type Employee struct {  
    Age      int    `feature:"identity"`  
    Salary   float64 `feature:"minmax"`  
    Kids     int    `feature:"maxabs"`  
    Weight   float64 `feature:"standard"`  
    Height   float64 `feature:"quantile"`  
    City     string `feature:"onehot"`  
    Car      string `feature:"ordinal"`  
    Income   float64 `feature:"kbins"`  
    Description string `feature:"tfidf"`  
    SecretValue float64  
}
```

```
187 // TransformInplace transforms struct into feature vector accordingly to transformers, and does so inplace  
188 func (e *AllTransformersFeatureTransformer) TransformInplace(dst []float64, s *AllTransformers) {  
189     if s == nil || e == nil || len(dst) != e.NumFeatures() {  
190         return  
191     }  
192     idx := 0  
193  
194     dst[idx] = e.Name0.Transform(float64(s.Name0))  
195     idx++  
196  
197     dst[idx] = e.Name1.Transform(float64(s.Name1))  
198     idx++  
199  
200     dst[idx] = e.Name2.Transform(float64(s.Name2))  
201     idx++  
202  
203     dst[idx] = e.Name3.Transform(float64(s.Name3))  
204     idx++  
205  
206     dst[idx] = e.Name4.Transform(float64(s.Name4))  
207     idx++  
208  
209     e.Name5.TransformInplace(dst[idx+e.Name5.NumFeatures():], s.Name5)  
210     idx += e.Name5.NumFeatures()  
211  
212     dst[idx] = e.Name6.Transform((s.Name6))  
213     idx++  
214  
215 // FeatureNames provides names of features that match output of transform  
216 func (e *AllTransformersFeatureTransformer) FeatureNames() []string {  
217     if e == nil {  
218         return nil  
219     }  
220     return e8.FeatureNames(), s.Name6  
221  
222     idx := 0  
223     names := make([]string, e.NumFeatures())  
224  
225     names[idx] = "Name0"  
226     idx++  
227  
228     names[idx] = "Name1"  
229     idx++  
230  
231     names[idx] = "Name2"  
232     idx++  
233  
234     names[idx] = "Name3"  
235     idx++  
236  
237     names[idx] = "Name4"  
238     idx++  
239  
240     for _, w := range e.Name5.FeatureNames() {  
241         names[idx] = "Name5_" + w  
242         idx++  
243     }  
244  
245     names[idx] = "Name6"  
246     idx++  
247  
248     names[idx] = "Name7"  
249     idx++  
250  
251     for _, w := range e.Name8.FeatureNames() {  
252         names[idx] = "Name8_" + w  
253         idx++  
254     }  
255  
256     for _, w := range e.Name9.FeatureNames() {  
257         names[idx] = "Name9_" + w  
258         idx++  
259     }  
260  
261     return names  
262 }
```

# Other mechanisms: go:generate

<https://github.com/nikolaydubina/go-featureprocessing>

- Very fast
- Functions get inlined

However!

- Laborious to write AST generator

```
2 // Code generated by go-featureprocessing DO NOT EDIT
3
4 package {{$.PackageName}}
5
6 import (
7     "sync"
8
9     fp "github.com/nikolaydubina/go-featureprocessing/transformers"
10 )
11
12 // {{$.StructName}}FeatureTransformer is a feature processor for {{$.StructName}}.
13 // It was automatically generated by go-featureprocessing tool.
14 type {{$.StructName}}FeatureTransformer struct {
15     {{range $t, $tr := $.Fields}}{{$.Name}} fp.{{$.Transformer}} ` + "`" + `json:"{{$.Name}}_{{$.TransformerTag}}" ` + "`" + `
16     {{end}}
17 }
18
19 // Fit fits transformer for each field
20 func (e *{{$.StructName}}FeatureTransformer) Fit(s []{{$.StructName}}) {
21     if e == nil || len(s) == 0 {
22         return
23     }
24     {{if $.HasNumericalTransformers}}dataNum := make([]float64, len(s)){{end}}
25     {{if $.HasStringTransformers}}dataStr := make([]string, len(s)){{end}}
26
27     for i, v := range s {
28         {{if $.NumericalInput}}dataNum[i] = float64(v.{{$.Name}}){{else}}dataStr[i] = v.{{$.Name}}{{end}}
29     }
30
31     e.{{$.Name}}.Fit({{if $.NumericalInput}}dataNum{{else}}dataStr{{end}})
32 }
33
34 // Transform transforms struct into feature vector according to transformers
35 func (e *{{$.StructName}}FeatureTransformer) Transform(s *{{$.StructName}}) []float64 {
36     if s == nil || e == nil {
37         return nil
38     }
39
40     // ...
41
42     // ...
43
44     // ...
45
46     // ...
47
48     // ...
49
50     // ...
51
52     // ...
53
54     // ...
55
56     // ...
57
58     // ...
59
60     // ...
61
62     // ...
63
64     // ...
65
66     // ...
67
68     // ...
69
70     // ...
71
72     // ...
73
74     // ...
75
76     // ...
77
78     // ...
79
80     // ...
81
82     // ...
83
84     // ...
85
86     // ...
87
88     // ...
89
90     // ...
91
92     // ...
93
94     // ...
95
96     // ...
97
98     // ...
99
100     // ...
101
102     // ...
103
104     // ...
105
106     // ...
107
108     // ...
109
110     // ...
111
112     // ...
113
114     // ...
115
116     // ...
117
118     // ...
119
120     // ...
121
122     // ...
123
124     // ...
125
126     // ...
127
128     // ...
129
130     // ...
131
132     // ...
133
134     // ...
135
136     // ...
137
138     // ...
139
140     // ...
141
142     // ...
143
144     // ...
145
146     // ...
147
148     // ...
149
150     // ...
151
152     // ...
153
154     // ...
155
156     // ...
157
158     // ...
159
160     // ...
161
162     // ...
163
164     // ...
165
166     // ...
167
168     // ...
169
170     // ...
171
172     // ...
173
174     // ...
175
176     // ...
177
178     // ...
179
180     // ...
181
182     // ...
183
184     // ...
185
186     // ...
187
188     // ...
189
190     // ...
191
192     // ...
193
194     // ...
195
196     // ...
197
198     // ...
199
200     // ...
201
202     // ...
203
204     // ...
205
206     // ...
207
208     // ...
209
210     // ...
211
212     // ...
213
214     // ...
215
216     // ...
217
218     // ...
219
220     // ...
221
222     // ...
223
224     // ...
225
226     // ...
227
228     // ...
229
230     // ...
231
232     // ...
233
234     // ...
235
236     // ...
237
238     // ...
239
240     // ...
241
242     // ...
243
244     // ...
245
246     // ...
247
248     // ...
249
250     // ...
251
252     // ...
253
254     // ...
255
256     // ...
257
258     // ...
259
260     // ...
261
262     // ...
263
264     // ...
265
266     // ...
267
268     // ...
269
270     // ...
271
272     // ...
273
274     // ...
275
276     // ...
277
278     // ...
279
280     // ...
281
282     // ...
283
284     // ...
285
286     // ...
287
288     // ...
289
290     // ...
291
292     // ...
293
294     // ...
295
296     // ...
297
298     // ...
299
300     // ...
301
302     // ...
303
304     // ...
305
306     // ...
307
308     // ...
309
310     // ...
311
312     // ...
313
314     // ...
315
316     // ...
317
318     // ...
319
320     // ...
321
322     // ...
323
324     // ...
325
326     // ...
327
328     // ...
329
330     // ...
331
332     // ...
333
334     // ...
335
336     // ...
337
338     // ...
339
340     // ...
341
342     // ...
343
344     // ...
345
346     // ...
347
348     // ...
349
350     // ...
351
352     // ...
353
354     // ...
355
356     // ...
357
358     // ...
359
360     // ...
361
362     // ...
363
364     // ...
365
366     // ...
367
368     // ...
369
370     // ...
371
372     // ...
373
374     // ...
375
376     // ...
377
378     // ...
379
380     // ...
381
382     // ...
383
384     // ...
385
386     // ...
387
388     // ...
389
390     // ...
391
392     // ...
393
394     // ...
395
396     // ...
397
398     // ...
399
400     // ...
401
402     // ...
403
404     // ...
405
406     // ...
407
408     // ...
409
410     // ...
411
412     // ...
413
414     // ...
415
416     // ...
417
418     // ...
419
420     // ...
421
422     // ...
423
424     // ...
425
426     // ...
427
428     // ...
429
430     // ...
431
432     // ...
433
434     // ...
435
436     // ...
437
438     // ...
439
440     // ...
441
442     // ...
443
444     // ...
445
446     // ...
447
448     // ...
449
450     // ...
451
452     // ...
453
454     // ...
455
456     // ...
457
458     // ...
459
460     // ...
461
462     // ...
463
464     // ...
465
466     // ...
467
468     // ...
469
470     // ...
471
472     // ...
473
474     // ...
475
476     // ...
477
478     // ...
479
480     // ...
481
482     // ...
483
484     // ...
485
486     // ...
487
488     // ...
489
490     // ...
491
492     // ...
493
494     // ...
495
496     // ...
497
498     // ...
499
500     // ...
501
502     // ...
503
504     // ...
505
506     // ...
507
508     // ...
509
510     // ...
511
512     // ...
513
514     // ...
515
516     // ...
517
518     // ...
519
520     // ...
521
522     // ...
523
524     // ...
525
526     // ...
527
528     // ...
529
530     // ...
531
532     // ...
533
534     // ...
535
536     // ...
537
538     // ...
539
540     // ...
541
542     // ...
543
544     // ...
545
546     // ...
547
548     // ...
549
550     // ...
551
552     // ...
553
554     // ...
555
556     // ...
557
558     // ...
559
560     // ...
561
562     // ...
563
564     // ...
565
566     // ...
567
568     // ...
569
570     // ...
571
572     // ...
573
574     // ...
575
576     // ...
577
578     // ...
579
580     // ...
581
582     // ...
583
584     // ...
585
586     // ...
587
588     // ...
589
590     // ...
591
592     // ...
593
594     // ...
595
596     // ...
597
598     // ...
599
600     // ...
601
602     // ...
603
604     // ...
605
606     // ...
607
608     // ...
609
610     // ...
611
612     // ...
613
614     // ...
615
616     // ...
617
618     // ...
619
620     // ...
621
622     // ...
623
624     // ...
625
626     // ...
627
628     // ...
629
630     // ...
631
632     // ...
633
634     // ...
635
636     // ...
637
638     // ...
639
640     // ...
641
642     // ...
643
644     // ...
645
646     // ...
647
648     // ...
649
650     // ...
651
652     // ...
653
654     // ...
655
656     // ...
657
658     // ...
659
660     // ...
661
662     // ...
663
664     // ...
665
666     // ...
667
668     // ...
669
670     // ...
671
672     // ...
673
674     // ...
675
676     // ...
677
678     // ...
679
680     // ...
681
682     // ...
683
684     // ...
685
686     // ...
687
688     // ...
689
690     // ...
691
692     // ...
693
694     // ...
695
696     // ...
697
698     // ...
699
700     // ...
701
702     // ...
703
704     // ...
705
706     // ...
707
708     // ...
709
710     // ...
711
712     // ...
713
714     // ...
715
716     // ...
717
718     // ...
719
720     // ...
721
722     // ...
723
724     // ...
725
726     // ...
727
728     // ...
729
730     // ...
731
732     // ...
733
734     // ...
735
736     // ...
737
738     // ...
739
740     // ...
741
742     // ...
743
744     // ...
745
746     // ...
747
748     // ...
749
750     // ...
751
752     // ...
753
754     // ...
755
756     // ...
757
758     // ...
759
760     // ...
761
762     // ...
763
764     // ...
765
766     // ...
767
768     // ...
769
770     // ...
771
772     // ...
773
774     // ...
775
776     // ...
777
778     // ...
779
780     // ...
781
782     // ...
783
784     // ...
785
786     // ...
787
788     // ...
789
790     // ...
791
792     // ...
793
794     // ...
795
796     // ...
797
798     // ...
799
800     // ...
801
802     // ...
803
804     // ...
805
806     // ...
807
808     // ...
809
810     // ...
811
812     // ...
813
814     // ...
815
816     // ...
817
818     // ...
819
820     // ...
821
822     // ...
823
824     // ...
825
826     // ...
827
828     // ...
829
830     // ...
831
832     // ...
833
834     // ...
835
836     // ...
837
838     // ...
839
840     // ...
841
842     // ...
843
844     // ...
845
846     // ...
847
848     // ...
849
850     // ...
851
852     // ...
853
854     // ...
855
856     // ...
857
858     // ...
859
860     // ...
861
862     // ...
863
864     // ...
865
866     // ...
867
868     // ...
869
870     // ...
871
872     // ...
873
874     // ...
875
876     // ...
877
878     // ...
879
880     // ...
881
882     // ...
883
884     // ...
885
886     // ...
887
888     // ...
889
890     // ...
891
892     // ...
893
894     // ...
895
896     // ...
897
898     // ...
899
900     // ...
901
902     // ...
903
904     // ...
905
906     // ...
907
908     // ...
909
910     // ...
911
912     // ...
913
914     // ...
915
916     // ...
917
918     // ...
919
920     // ...
921
922     // ...
923
924     // ...
925
926     // ...
927
928     // ...
929
930     // ...
931
932     // ...
933
934     // ...
935
936     // ...
937
938     // ...
939
940     // ...
941
942     // ...
943
944     // ...
945
946     // ...
947
948     // ...
949
950     // ...
951
952     // ...
953
954     // ...
955
956     // ...
957
958     // ...
959
960     // ...
961
962     // ...
963
964     // ...
965
966     // ...
967
968     // ...
969
970     // ...
971
972     // ...
973
974     // ...
975
976     // ...
977
978     // ...
979
980     // ...
981
982     // ...
983
984     // ...
985
986     // ...
987
988     // ...
989
990     // ...
991
992     // ...
993
994     // ...
995
996     // ...
997
998     // ...
999
1000    // ...
1001
1002    // ...
1003
1004    // ...
1005
1006    // ...
1007
1008    // ...
1009
1010    // ...
1011
1012    // ...
1013
1014    // ...
1015
1016    // ...
1017
1018    // ...
1019
1020    // ...
1021
1022    // ...
1023
1024    // ...
1025
1026    // ...
1027
1028    // ...
1029
1030    // ...
1031
1032    // ...
1033
1034    // ...
1035
1036    // ...
1037
1038    // ...
1039
1040    // ...
1041
1042    // ...
1043
1044    // ...
1045
1046    // ...
1047
1048    // ...
1049
1050    // ...
1051
1052    // ...
1053
1054    // ...
1055
1056    // ...
1057
1058    // ...
1059
1060    // ...
1061
1062    // ...
1063
1064    // ...
1065
1066    // ...
1067
1068    // ...
1069
1070    // ...
1071
1072    // ...
1073
1074    // ...
1075
1076    // ...
1077
1078    // ...
1079
1080    // ...
1081
1082    // ...
1083
1084    // ...
1085
1086    // ...
1087
1088    // ...
1089
1090    // ...
1091
1092    // ...
1093
1094    // ...
1095
1096    // ...
1097
1098    // ...
1099
1100    // ...
1101
1102    // ...
1103
1104    // ...
1105
1106    // ...
1107
1108    // ...
1109
1110    // ...
1111
1112    // ...
1113
1114    // ...
1115
1116    // ...
1117
1118    // ...
1119
1120    // ...
1121
1122    // ...
1123
1124    // ...
1125
1126    // ...
1127
1128    // ...
1129
1130    // ...
1131
1132    // ...
1133
1134    // ...
1135
1136    // ...
1137
1138    // ...
1139
1140    // ...
1141
1142    // ...
1143
1144    // ...
1145
1146    // ...
1147
1148    // ...
1149
1150    // ...
1151
1152    // ...
1153
1154    // ...
1155
1156    // ...
1157
1158    // ...
1159
1160    // ...
1161
1162    // ...
1163
1164    // ...
1165
1166    // ...
1167
1168    // ...
1169
1170    // ...
1171
1172    // ...
1173
1174    // ...
1175
1176    // ...
1177
1178    // ...
1179
1180    // ...
1181
1182    // ...
1183
1184    // ...
1185
1186    // ...
1187
1188    // ...
1189
1190    // ...
1191
1192    // ...
1193
1194    // ...
1195
1196    // ...
1197
1198    // ...
1199
1200    // ...
1201
1202    // ...
1203
1204    // ...
1205
1206    // ...
1207
1208    // ...
1209
1210    // ...
1211
1212    // ...
1213
1214    // ...
1215
1216    // ...
1217
1218    // ...
1219
1220    // ...
1221
1222    // ...
1223
1224    // ...
1225
1226    // ...
1227
1228    // ...
1229
1230    // ...
1231
1232    // ...
1233
1234    // ...
1235
1236    // ...
1237
1238    // ...
1239
1240    // ...
1241
1242    // ...
1243
1244    // ...
1245
1246    // ...
1247
1248    // ...
1249
1250    // ...
1251
1252    // ...
1253
1254    // ...
1255
1256    // ...
1257
1258    // ...
1259
1260    // ...
1261
1262    // ...
1263
1264    // ...
1265
1266    // ...
1267
1268    // ...
1269
1270    // ...
1271
1272    // ...
1273
1274    // ...
1275
1276    // ...
1277
1278    // ...
1279
1280    // ...
1281
1282    // ...
1283
1284    // ...
1285
1286    // ...
1287
1288    // ...
1289
1290    // ...
1291
1292    // ...
1293
1294    // ...
1295
1296    // ...
1297
1298    // ...
1299
1300    // ...
1301
1302    // ...
1303
1304    // ...
1305
1306    // ...
1307
1308    // ...
1309
1310    // ...
1311
1312    // ...
1313
1314    // ...
1315
1316    // ...
1317
1318    // ...
1319
1320    // ...
1321
1322    // ...
1323
1324    // ...
1325
1326    // ...
1327
1328    // ...
1329
1330    // ...
1331
1332    // ...
1333
1334    // ...
1335
1336    // ...
1337
1338    // ...
1339
1340    // ...
1341
1342    // ...
1343
1344    // ...
1345
1346    // ...
1347
1348    // ...
1349
1350    // ...
1351
1352    // ...
1353
1354    // ...
1355
1356    // ...
1357
1358    // ...
1359
1360    // ...
1361
1362    // ...
1363
1364    // ...
1365
1366    // ...
1367
1368    // ...
1369
1370    // ...
1371
1372    // ...
1373
1374    // ...
1375
1376    // ...
1377
1378    // ...
1379
1380    // ...
1381
1382    // ...
1383
1384    // ...
1385
1386    // ...
1387
1388    // ...
1389
1390    // ...
1391
1392    // ...
1393
1394    // ...
1395
1396    // ...
1397
1398    // ...
1399
1400    // ...
1401
1402    // ...
1403
1404    // ...
1405
1406    // ...
1407
1408    // ...
1409
1410    // ...
1411
1412    // ...
1413
1414    // ...
1415
1416    // ...
1417
1418    // ...
1419
1420    // ...
1421
1422    // ...
1423
1424    // ...
1425
1426    // ...
1427
1428    // ...
1429
1430    // ...
1431
1432    // ...
1433
1434    // ...
1435
1436    // ...
1437
1438    // ...
1439
1440    // ...
1441
1442    // ...
1443
1444    // ...
1445
1446    // ...
1447
1448    // ...
1449
1450    // ...
1451
1452    // ...
1453
1454    // ...
1455
1456    // ...
1457
1458    // ...
1459
1460    // ...
1461
1462    // ...
1463
1464    // ...
1465
1466    // ...
1467
1468    // ...
1469
1470    // ...
1471
1472    // ...
1473
1474    // ...
1475
1476    // ...
1477
1478    // ...
1479
1480    // ...
1481
1482    // ...
1483
1484    // ...
1485
1486    // ...
1487
1488    // ...
1489
1490    // ...
1491
1492    // ...
1493
1494    // ...
1495
1496    // ...
1497
1498    // ...
1499
1500    // ...
1501
1502    // ...
1503
1504    // ...
1505
1506    // ...
1507
1508    // ...
1509
1510    // ...
1511
1512    // ...
1513
1514    // ...
1515
1516    // ...
1517
1518    // ...
1519
1520    // ...
1521
1522    // ...
1523
1524    // ...
1525
1526    // ...
1527
1528    // ...
1529
1530    // ...
1531
1532    // ...
1533
1534    // ...
1535
1536    // ...
1537
1538    // ...
1539
1540    // ...
1541
1542    // ...
1543
1544    // ...
1545
1546    // ...
1547
1548    // ...
1549
1550    // ...
1551
1552    // ...
1553
1554    // ...
1555
1556    // ...
1557
1558    // ...
1559
1560    // ...
1561
1562    // ...
1563
1564    // ...
1565
1566    // ...
1567
1568    // ...
1569
1570    // ...
1571
1572    // ...
1573
1574    // ...
1575
1576    // ...
1577
1578    // ...
1579
1580    // ...
1581
1582    // ...
1583
1584    // ...
1585
1586    // ...
1587
1588    // ...
1589
1590    // ...
1591
1592    // ...
1593
1594    // ...
1595
1596    // ...
1597
1598    // ...
1599
1600    // ...
1601
1602    // ...
1603
1604    // ...
1605
1606    // ...
1607
1608    // ...
1609
1610    // ...
1611
1612    // ...
1613
1614    // ...
1615
1616    // ...
1617
1618    // ...
1619
1620    // ...
1621
1622    // ...
1623
1624    // ...
1625
1626    // ...
1627
1628    // ...
1629
1630    // ...
1631
1632    // ...
1633
1634    // ...
1635
1636    // ...
1637
1638    // ...
1639
1640    // ...
1641
1642    // ...
1643
1644    // ...
1645
1646    // ...
1647
1648    // ...
1649
1650    // ...
1651
1652    // ...
1653
1654    // ...
1655
1656    // ...
1657
1658    // ...
1659
1660    // ...
1661
1662    // ...
1663
1664    // ...
1665
1666    // ...
1667
1668    // ...
1669
1670    // ...
1671
1672    // ...
1673
1674    // ...
1675
1676    // ...
1677
1678    // ...
1679
1680    // ...
1681
1682    // ...
1683
1684    // ...
1685
1686    // ...
1687
1688    // ...
1689
1690    // ...
1691
1692    // ...
1693
1694    // ...
1695
1696    // ...
1697
1698    // ...
1699
1700    // ...
1701
1702    // ...
1703
1704    // ...
1705
1706    // ...
1707
1708    // ...
1709
1710    // ...
1711
1712    // ...
1713
1714    // ...
1715
1716    // ...
1717
1718    // ...
1719
1720    // ...
1721
1722    // ...
1723
1724    // ...
1725
1726    // ...
1727
1728    // ...
1729
1730    // ...
1731
1732    // ...
1733
1734    // ...
1735
1736    // ...
1737
1738    // ...
1739
1740    // ...
1741
1742    // ...
1743
1744    // ...
1745
1746    // ...
1747
1748    // ...
1749
1750    // ...
1751
1752    // ...
1753
1754    // ...
1755
1756    // ...
1757
1758    // ...
1759
1760    // ...
1761
1762    // ...
1763
1764    // ...
1765
1766    // ...
1767
1768    // ...
1769
1770    // ...
1771
1772    // ...
1773
1774    // ...
1775
1776    // ...
1777
1778    // ...
1779
1780    // ...
1781
1782    // ...
1783
1784    // ...
1785
1786    // ...
1787
1788    // ...
1789
1790    // ...
1791
1792    // ...
1793
1794    // ...
1795
1796    // ...
1797
1798    // ...
1799
1800    // ...
1801
1802    // ...
1803
1804    // ...
1805
1806    // ...
1807
1808    // ...
1809
1810    // ...
1811
1812    // ...
1813
1814    // ...
1815
1816    // ...
1817
1818    // ...
1819
1820    // ...
1821
1822    // ...
1823
1824    // ...
1825
1826    // ...
1827
1828    // ...
1829
1830    // ...
1831
1832    // ...
1833
1834    // ...
1835
1836    // ...
1837
1838    // ...
1839
1840    // ...
1841
1842    // ...
1843
1844    // ...
1845
1846    // ...
1847
1848    // ...
1849
1850    // ...
1851
1852    // ...
1853
1854    // ...
1855
1856    // ...
1857
1858    // ...
1859
1860    // ...
1861
1862    // ...
1863
1864    // ...
1865
1866    // ...
1867
1868    // ...
1869
1870    // ...
1871
1872    // ...
1873
1874    // ...
1875
1876    // ...
1877
1878    // ...
1879
1880    // ...
1881
1882    // ...
1883
1884    // ...
1885
1886    // ...
1887
1888    // ...
1889
1890    // ...
1891
1892    // ...
1893
1894    // ...
1895
1896    // ...
1897
1898    // ...
1899
1900    // ...
1901
1902    // ...
1903
1904    // ...
1905
1906    // ...
1907
1908    // ...
1909
1910    // ...
1911
1912    // ...
1913
1914    // ...
1915
1916    // ...
1917
1918    // ...
1919
1920    // ...
1921
1922    // ...
1923
1924    // ...
1925
1926    // ...
1927
1928    // ...
1929
1930    // ...
1931
1932    // ...
1933
1934    // ...
1935
1936    // ...
1937
1938    // ...
1939
1940    // ...
1941
1942    // ...
1943
1944    // ...
1945
1946    // ...
1947
1948    // ...
1949
1950    // ...
1951
1952    // ...
1953
1954    // ...
1955
1956    // ...
1957
1958    // ...
1959
1960    // ...
1961
1962    // ...
1963
1964    // ...
1965
1966    // ...
1967
1968    // ...
1969
1970    // ...
1971
1972    // ...
1973
1974    // ...
1975
1976    // ...
1977
1978    // ...
1979
1980    // ...
1981
1982    // ...
1983
1984    // ...
1985
1986    // ...
1987
1988    // ...
1989
19
```

Thank you!