```go
package main

import    "fmt"

// Изменяем только данные слайса-параметра
func f1(c []int)  {
    for i, _ := range c  {
         c[i] +=100
    }
    fmt.Println(c, len(c), cap(c))
}

func f2(pc *[]int)  {
    for i, _ := range *pc  {
        (*pc)[i] +=100
    }
    fmt.Println(*pc, len(*pc), cap(*pc))
}

func f3(c []int) []int {
    for i, _ := range c  {
        c[i] +=100
    }
    fmt.Println(c, len(c), cap(c))
    return c
}

func main() {
    a:= [...]int {0,1,2,3,4,5,6,7,8,9}
    fmt.Println(a)                  // [0 1 2 3 4 5 6 7 8 9]

    arr:= a
    b:= arr[2:7]
    fmt.Println(b, len(b), cap(b))  // [2 3 4 5 6] 5 8
    f1(b)                           // [102 103 104 105 106] 5 8
    fmt.Println(b, len(b), cap(b))  // [102 103 104 105 106] 5 8
    fmt.Println(arr)                // [0 1 102 103 104 105 106 7 8 9]

    arr = a
    b = arr[2:7]
    fmt.Println(b, len(b), cap(b))  // [2 3 4 5 6] 5 8
    f2(&b)                          // [102 103 104 105 106] 5 8
    fmt.Println(b, len(b), cap(b))  // [102 103 104 105 106] 5 8
    fmt.Println(arr)                // [0 1 102 103 104 105 106 7 8 9]

    arr = a
    b = arr[2:7]
    fmt.Println(b, len(b), cap(b))  // [2 3 4 5 6] 5 8
    b = f3(b)                       // [102 103 104 105 106] 5 8
    fmt.Println(b, len(b), cap(b))  // [102 103 104 105 106] 5 8
    fmt.Println(arr)                // [0 1 102 103 104 105 106 7 8 9]
}
```

01a.go

```go
package main

import    "fmt"

// Изменяем только длину слайса-параметра
func f1(c []int)  {
    c = c[:cap(c)]
    fmt.Println(c, len(c), cap(c))
}

func f2(pc *[]int)  {
    *pc = (*pc)[:cap(*pc)]
    fmt.Println(*pc, len(*pc), cap(*pc))
}

func f3(c []int) []int {
    c = c[:cap(c)]
    fmt.Println(c, len(c), cap(c))
    return c
}

func main() {
    a:= [...]int {0,1,2,3,4,5,6,7,8,9}
    fmt.Println(a)                      // [0 1 2 3 4 5 6 7 8 9]

    arr:= a
    b:= arr[2:7]
    fmt.Println(b, len(b), cap(b))      // [2 3 4 5 6] 5 8
    f1(b)                               // [2 3 4 5 6 7 8 9] 8 8
    fmt.Println(b, len(b), cap(b))      // [2 3 4 5 6] 5 8
    fmt.Println(arr)                    // [0 1 2 3 4 5 6 7 8 9]

    arr = a
    b = arr[2:7]
    fmt.Println(b, len(b), cap(b))      // [2 3 4 5 6] 5 8
    f2(&b)                              // [2 3 4 5 6 7 8 9] 8 8
    fmt.Println(b, len(b), cap(b))      // [2 3 4 5 6 7 8 9] 8 8
    fmt.Println(arr)                    // [0 1 2 3 4 5 6 7 8 9]

    arr = a
    b = arr[2:7]
    fmt.Println(b, len(b), cap(b))      // [2 3 4 5 6] 5 8
    b = f3(b)                           // [2 3 4 5 6 7 8 9] 8 8
    fmt.Println(b, len(b), cap(b))      // [2 3 4 5 6 7 8 9] 8 8
    fmt.Println(arr)                    // [0 1 2 3 4 5 6 7 8 9]
}
```

01b.go

```go
package main

import     "fmt"

// Изменяем длину слайса-параметра, а затем данные
func f1(c []int)  {
    c = c[:cap(c)]
    for i, _ := range c  {
        c[i] +=100
    }
    fmt.Println(c, len(c), cap(c))
}

func f2(pc *[]int)  {
    *pc = (*pc)[:cap(*pc)]
    for i, _ := range *pc  {
        (*pc)[i] +=100
    }
    fmt.Println(*pc, len(*pc), cap(*pc))
}

func f3(c []int) []int {
    c = c[:cap(c)]
    for i, _ := range c  {
        c[i] +=100
    }
    fmt.Println(c, len(c), cap(c))
    return c
}

func main() {
    a:= [...]int {0,1,2,3,4,5,6,7,8,9}
    fmt.Println(a)                  // [0 1 2 3 4 5 6 7 8 9]

    arr:= a
    b:= arr[2:7]
    fmt.Println(b, len(b), cap(b))  // [2 3 4 5 6] 5 8
    f1(b)                           // [102 103 104 105 106 107 108 109] 8 8
    fmt.Println(b, len(b), cap(b))  // [102 103 104 105 106] 5 8
    fmt.Println(arr)                // [0 1 102 103 104 105 106 107 108 109]

    arr = a
    b = arr[2:7]
    fmt.Println(b, len(b), cap(b))  // [2 3 4 5 6] 5 8
    f2(&b)                          // [102 103 104 105 106 107 108 109] 8 8
    fmt.Println(b, len(b), cap(b))  // [102 103 104 105 106 107 108 109] 8 8
    fmt.Println(arr)                // [0 1 102 103 104 105 106 107 108 109]

    arr = a
    b = arr[2:7]
    fmt.Println(b, len(b), cap(b))  // [2 3 4 5 6] 5 8
    b = f3(b)                       // [102 103 104 105 106 107 108 109] 8 8
    fmt.Println(b, len(b), cap(b))  // [102 103 104 105 106 107 108 109] 8 8
    fmt.Println(arr)                // [0 1 102 103 104 105 106 107 108 109]
}
```

01c.go

```go
package main

import    "fmt"

// Добавляем в слайс-параметр данные по одному,
// пока не произойдёт перераспределения памяти
func f1(c []int)  {
    ccap:= cap(c)
    for cap(c) == ccap  {
        c = append(c, len(c)*10)
    }
    fmt.Println(c, len(c), cap(c))
}

func f2(pc *[]int)  {
    ccap:= cap(*pc)
    for cap(*pc) == ccap  {
        *pc = append(*pc, len(*pc)*10)
    }
    fmt.Println(*pc, len(*pc), cap(*pc))
}

func f3(c []int) []int {
    ccap:= cap(c)
    for cap(c) == ccap  {
        c = append(c, len(c)*10)
    }
    fmt.Println(c, len(c), cap(c))
    return c
}

func main() {
    a:= [...]int {0,1,2,3,4,5,6,7,8,9}
    fmt.Println(a)                      // [0 1 2 3 4 5 6 7 8 9]

    arr:= a
    b:= arr[2:7]
    fmt.Println(b, len(b), cap(b))      // [2 3 4 5 6] 5 8
    f1(b)                               // [2 3 4 5 6 50 60 70 80] 9 16
    fmt.Println(b, len(b), cap(b))      // [2 3 4 5 6] 5 8
    fmt.Println(arr)                    // [0 1 2 3 4 5 6 50 60 70]

    arr = a
    b = arr[2:7]
    fmt.Println(b, len(b), cap(b))      // [2 3 4 5 6] 5 8
    f2(&b)                              // [2 3 4 5 6 50 60 70 80] 9 16
    fmt.Println(b, len(b), cap(b))      // [2 3 4 5 6 50 60 70 80] 9 16
    fmt.Println(arr)                    // [0 1 2 3 4 5 6 50 60 70]

    arr = a
    b = arr[2:7]
    fmt.Println(b, len(b), cap(b))      // [2 3 4 5 6] 5 8
    b = f3(b)                           // [2 3 4 5 6 50 60 70 80] 9 16
    fmt.Println(b, len(b), cap(b))      // [2 3 4 5 6 50 60 70 80] 9 16
    fmt.Println(arr)                    // [0 1 2 3 4 5 6 50 60 70]
}
```

01d.go

```go
package main

import  "fmt"

// Добавляем в слайс-параметр данные группой, размер
// которой требует перераспределения памяти
func f1(c []int)  {
    cc:= make([]int, cap(c) - len(c) + 1)
    c = append(c, cc...)
    fmt.Println(c, len(c), cap(c))
}

func f2(pc *[]int)  {
    cc:= make([]int, cap(*pc) - len(*pc) + 1)
    *pc = append(*pc, cc...)
    fmt.Println(*pc, len(*pc), cap(*pc))
}

func f3(c []int) []int {
    cc:= make([]int, cap(c) - len(c) + 1)
    c = append(c, cc...)
    fmt.Println(c, len(c), cap(c))
    return c
}

func main() {
    a:= [...]int {0,1,2,3,4,5,6,7,8,9}
    fmt.Println(a)                      // [0 1 2 3 4 5 6 7 8 9]

    arr:= a
    b:= arr[2:7]
    fmt.Println(b, len(b), cap(b))      // [2 3 4 5 6] 5 8
    f1(b)                               // [2 3 4 5 6 0 0 0 0] 9 16
    fmt.Println(b, len(b), cap(b))      // [2 3 4 5 6] 5 8
    fmt.Println(arr)                    // [0 1 2 3 4 5 6 7 8 9]

    arr = a
    b = arr[2:7]
    fmt.Println(b, len(b), cap(b))      // [2 3 4 5 6] 5 8
    f2(&b)                              // [2 3 4 5 6 0 0 0 0] 9 16
    fmt.Println(b, len(b), cap(b))      // [2 3 4 5 6 0 0 0 0] 9 16
    fmt.Println(arr)                    // [0 1 2 3 4 5 6 7 8 9]

    arr = a
    b = arr[2:7]
    fmt.Println(b, len(b), cap(b))      // [2 3 4 5 6] 5 8
    b = f3(b)                           // [2 3 4 5 6 0 0 0 0] 9 16
    fmt.Println(b, len(b), cap(b))      // [2 3 4 5 6 0 0 0 0] 9 16
    fmt.Println(arr)                    // [0 1 2 3 4 5 6 7 8 9]
}
```

01e.go

```go
package main

// Запуск из командной строки:
//         go test -bench . benchmark_sample_test.go
// В данном случае benchmark_sample_test.go - это имя файла,
// в котором находится данная программа. Имя тестируемого
// файла обязательно должно заканиваться на _test

import "testing"

func lala(s []int) {
     for i, x := range(s) {
          s[i] = (x-1)*x*(x+1)
     }
}
// Названия тестируемых функций должны начинаться на Benchmark,
// за которым идёт название, начинающееся с большой буквы

// Оценивается функция BenchmarkLala.
func BenchmarkLala(b *testing.B)  {
     var s [100000]int
    for i:= 0; i < b.N; i++  {
          lala(s[:])
    }
}

// Оценивается функция BenchmarkLalala.
func BenchmarkLalala(b *testing.B)  {
     var s [100000]int
    for k:= 0; k < b.N; k++  {
        for i, x := range s  {
            s[i] = (x-1)*x*(x+1)
        }
    }
}

// Оценивается функция BenchmarkLalala2.
func BenchmarkLalala2(b *testing.B)  {
     s:= make([]int, 100000)
    for k:= 0; k < b.N; k++  {
        for i, x := range s  {
            s[i] = (x-1)*x*(x+1)
        }
    }
}

// OUTPUT
// goos: windows
// goarch: amd64
// BenchmarkLala-4          12103       99203 ns/op
// BenchmarkLalala-4         7063      161336 ns/op
// BenchmarkLalala2-4       12094       98955 ns/op
// PASS
// ok     command-line-arguments   5.765s
```

benchmark_sample_test.go

```
1  package main
2
3  import (
4       "testing"
5       "fmt"
6  )
7
8  func lala(s []int) {
9       for i, x := range(s) {
10           s[i] = (x-1)*x*(x+1)
11      }
12 }
13
14 func main() {
15      var s [100000]int
16      res:= testing.Benchmark(
17           func(b *testing.B) {
18                for i:= 0; i < b.N; i++  {
19                     lala(s[:])
20                }
21           })
22      fmt.Println(res)
23 }
```

benchmark_sample_02.go

```
1  // Запуск:
2  //       go test -bench . insert_test.go
3
4  // файл должен иметь суффикс _test
5  // оцениваемые функции должны начинаться с Benchmark,
6  // за которым идёт название, начинающееся с большой буквы
7
8  package insert
9
10 import (
11      "container/list"
12      "testing"
13 )
14
15 var (
16      slice10   = slice(10)
17      slice100  = slice(100)
18      slice1k   = slice(1000)
19      slice10k  = slice(10000)
20      slice100k = slice(100000)
21      slice1m   = slice(1000000)
22 )
23
24 func slice(size int) []byte { return make([]byte, size) }
25
26 func BenchmarkInsertHeadSlice10(b *testing.B)      {
27     benchmarkInsertSlice(b, slice10) }
28 func BenchmarkInsertHeadSlice100(b *testing.B)     {
29     benchmarkInsertSlice(b, slice100) }
30 func BenchmarkInsertHeadSlice1000(b *testing.B)    {
31     benchmarkInsertSlice(b, slice1k) }
32 func BenchmarkInsertHeadSlice10000(b *testing.B)   {
33     benchmarkInsertSlice(b, slice10k) }
```

```
34 func BenchmarkInsertHeadSlice100000(b *testing.B)  {
35     benchmarkInsertSlice(b, slice100k) }
36 func BenchmarkInsertHeadSlice1000000(b *testing.B) {
37     benchmarkInsertSlice(b, slice1m) }
38
39 func benchmarkInsertSlice(b *testing.B, a []byte) {
40      c := cap(a)
41      for n := 0; n < b.N; n++ {
42           // insert at i trick:
43           // a = append(a[:i], append([]T{x}, a[i:]...)...)
44           // could be simplified as we insert to head
45           a = append([]byte{0}, a...)
46           // reset a
47           a = a[:c:c]
48      }
49 }
50
51 var (
52      linkedList10   = linkedList(10)
53      linkedList100  = linkedList(100)
54      linkedList1k   = linkedList(1000)
55      linkedList10k  = linkedList(10000)
56      linkedList100k = linkedList(100000)
57      linkedList1m   = linkedList(1000000)
58 )
59
60 func linkedList(size int) *list.List {
61      l := list.New()
62      for i := 0; i < size; i++ {
63           l.PushBack(0)
64      }
65      return l
66 }
67
68 func BenchmarkInsertHeadList10(b *testing.B)      {
69     benchmarkInsertList(b, linkedList10) }
70 func BenchmarkInsertHeadList100(b *testing.B)     {
71     benchmarkInsertList(b, linkedList100) }
72 func BenchmarkInsertHeadList1000(b *testing.B)    {
73     benchmarkInsertList(b, linkedList1k) }
74 func BenchmarkInsertHeadList10000(b *testing.B)   {
75     benchmarkInsertList(b, linkedList10k) }
76 func BenchmarkInsertHeadList100000(b *testing.B)  {
77     benchmarkInsertList(b, linkedList100k) }
78 func BenchmarkInsertHeadList1000000(b *testing.B) {
79     benchmarkInsertList(b, linkedList1m) }
80
81 func benchmarkInsertList(b *testing.B, l *list.List) {
82      for n := 0; n < b.N; n++ {
83           l.PushFront(0)
84           // reset l
85           l.Remove(l.Front())
86      }
87 }
```

insert_test.go