Golang Week 9 Homework

Yi-Ting Shih (111550013)
National Yang Ming Chaio Tung University
ytshih@cs.nycu.edu.tw

Code

```go
package intset

import (
  "bytes"
  "fmt"
  "math/bits"
  "slices"
)

type IntSet struct {
  words []uint64
}

func (s *IntSet) Has(x int) bool {
  word, bit := x/64, uint(x%64)
  return word < len(s.words) && s.words[word]&uint64(1<<bit) != 0
}

func (s *IntSet) Add(x int) {
  word, bit := x/64, uint(x%64)
  for word >= len(s.words) {
    s.words = append(s.words, 0)
  }
  s.words[word] |= uint64(1 << bit)
}

func (s *IntSet) UnionWith(t *IntSet) {
  for i, tword := range t.words {
    if i < len(s.words) {
      s.words[i] |= tword
    } else {
      s.words = append(s.words, tword)
    }
  }
}

func (s *IntSet) String() string {
  var buf bytes.Buffer
  buf.WriteByte('{')
  for i, word := range s.words {
    if word == 0 {
      continue
    }
    for j := 0; j < 64; j++ {
      if word&uint64(1<<uint(j)) != 0 {
        if buf.Len() > len("{") {
          buf.WriteByte(' ')
        }
        fmt.Fprintf(&buf, "%d", 64*i+j)
      }
    }
  }
  buf.WriteByte('}')
  return buf.String()
}

func (s *IntSet) Len() int {
  res := 0
  for _, word := range s.words {
    res += bits.OnesCount64(word)
  }
  return res
}
```

```
func (s *IntSet) Remove(x int) {
  word, bit := x/64, uint(x%64)
  if word < len(s.words) {
    s.words[word] &= ^(1 << bit)
  }
}

func (s *IntSet) Clear() {
  s.words = []uint64{}
}

func (s *IntSet) Copy() *IntSet {
  return &IntSet{
    words: slices.Clone(s.words),
  }
}

func (s *IntSet) AddAll(xs ...int) {
  for _, x := range xs {
    s.Add(x)
  }
}

func (s *IntSet) IntersectWith(o *IntSet) {
  for word := range s.words {
    if word >= len(o.words) {
      return
    }
    s.words[word] &= o.words[word]
  }
}

func (s *IntSet) DifferenceWith(o *IntSet) {
  for word := range s.words {
    if word >= len(o.words) {
      return
    }
    s.words[word] &= ^o.words[word]
  }
}

func (s *IntSet) SymmetricDifference(o *IntSet) {
  for word := range s.words {
    if word >= len(o.words) {
      return
    }
    s.words[word] ^= o.words[word]
  }
  for i := len(s.words); i < len(o.words); i++ {
    s.words = append(s.words, o.words[i])
  }
}
```

Result

```
=== RUN   TestLen
--- PASS: TestLen (0.00s)
=== RUN   TestRemove
--- PASS: TestRemove (0.00s)
=== RUN   TestClear
--- PASS: TestClear (0.00s)
=== RUN   TestCopy
--- PASS: TestCopy (0.00s)
=== RUN   TestAddAll
--- PASS: TestAddAll (0.00s)
=== RUN   TestIntersectWith
--- PASS: TestIntersectWith (0.00s)
=== RUN   TestDifferenceWith
--- PASS: TestDifferenceWith (0.00s)
=== RUN   TestSymmetricDifference
--- PASS: TestSymmetricDifference (0.00s)
=== RUN   Example_one
--- PASS: Example_one (0.00s)
```

```
=== RUN   Example_two
--- PASS: Example_two (0.00s)
PASS
ok      intset  (cached)
```