MAPPE

= FUNZIONE (in senso matematico)

$$f: A \rightarrow B$$

MAPPA = insieme di coppie (chiave, valore)

chiave: T_1 — valore: T_2

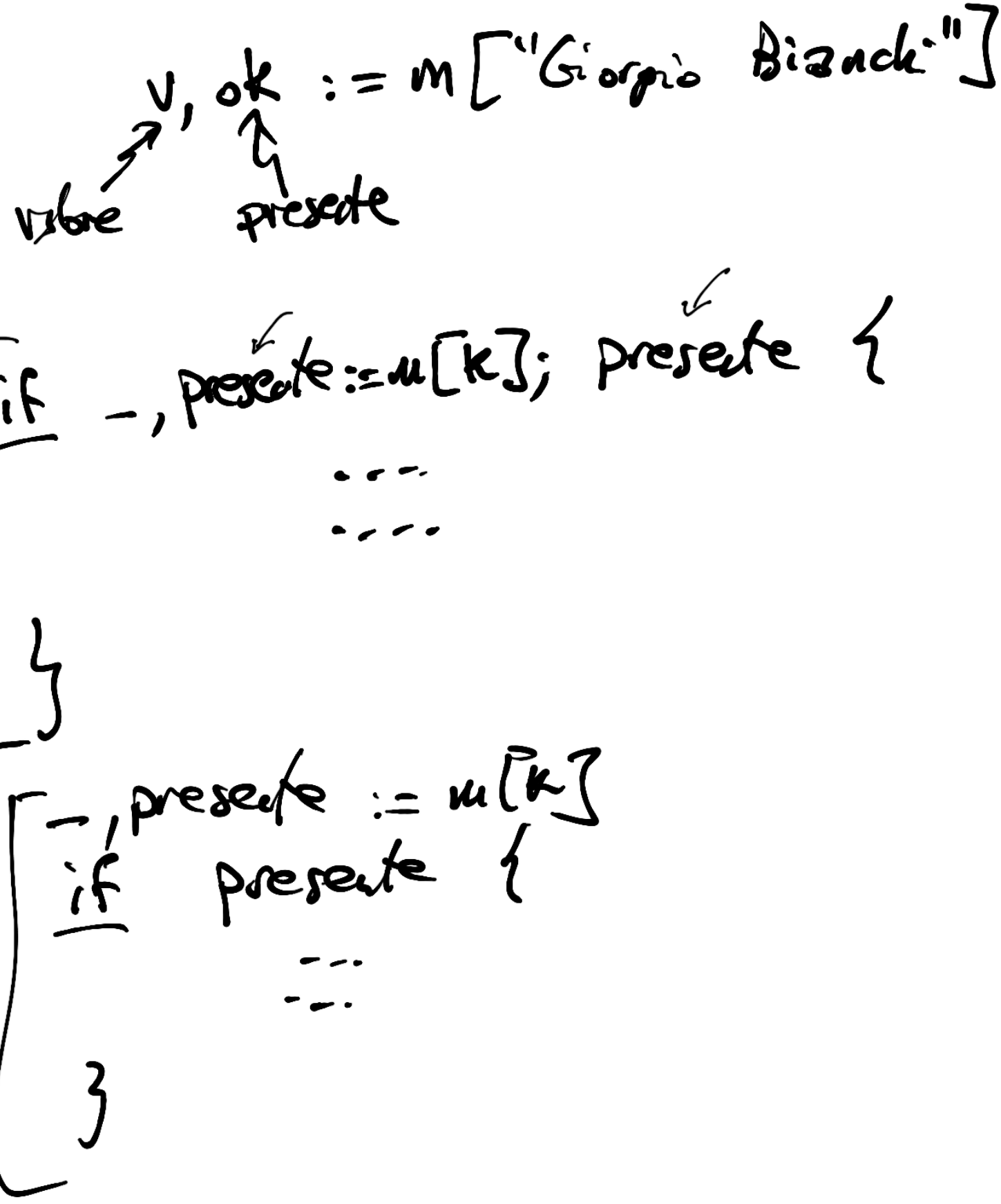
dove ad ogni chiave è associato un solo valore

---

map[T_1] T_2

- Associazione Nome → Anno di Nascita

```
var m map[string]int
m = make(map[string]int)
m["Paolo Baldi"] = 1968
m["Giovanni Rutti"] = 1984
m["Giorgio Vanni"] = 1985
len(m)
m["Giorgio Vanni"] = 1970
fmt.Println(m["Paolo Baldi"])
fmt.Println(m["Giorgio Bianchi"])
```


v, ok := m["Giorgio Bianchi"]
valore
presente
if _, presente := m[k]; presente {
...
}
_, presente := m[k]
if presente {
...
}

```
for k, v := range m {

    ...

}
```

```
delete(m, "Paolo Boldi")
```

```
var m map[string]int
m = map[string]int{"Paolo Boldi": 1968,
    "Incredibile Hulk": 1940,
    "Giorgio Correnti": 1947}
```

```
type data struct {
    g, m, a int
}

var p map[string] data
p= make (map [string ] data)
p["Paolo Boldi "]= data {29, 11, 1968}
p ["Giorgio Bianchi"] = data {1, 1, 1960}
        :
```

SCRIVETE UNA FUNZIONE CHE DATA UNA map[string]data E UN ANNO, RESTITUISCA LA SLICE DELLE PERSONE NATE IN QUELL'ANNO

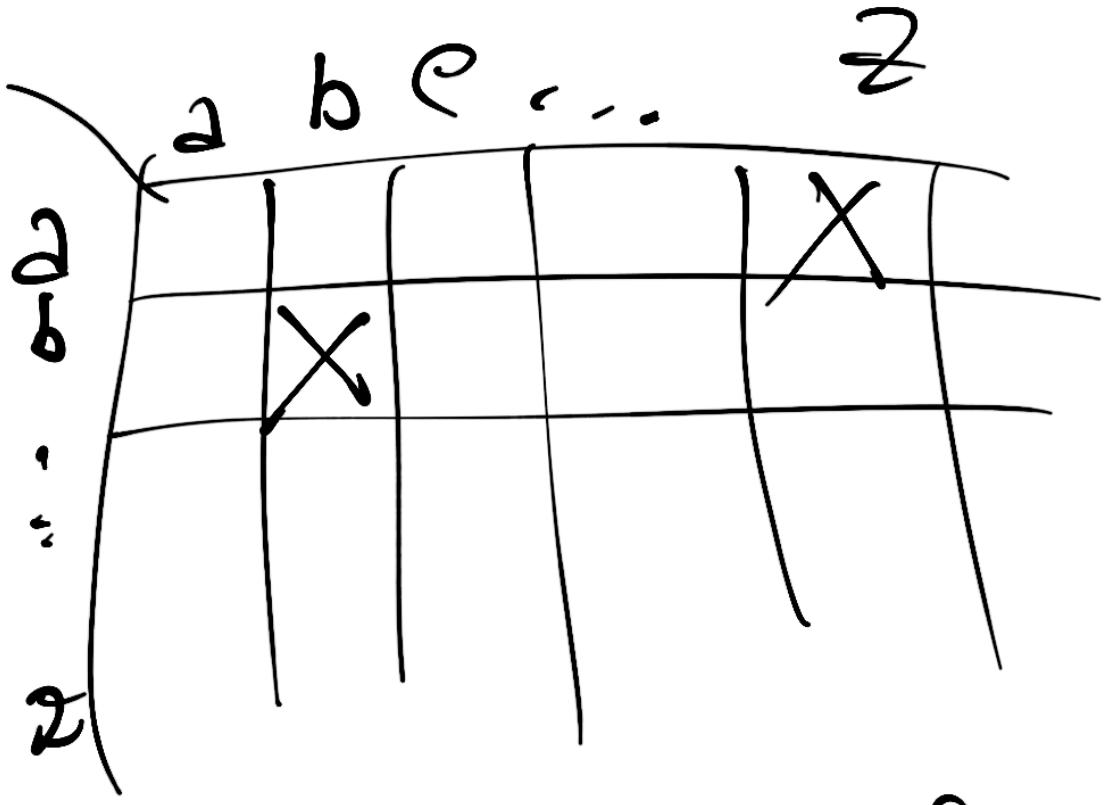
```
func natiNellAnno(m map[string]dat2,
    anno int) []string {
    var ris []string
    for nome, nascita := range m {
        if nascita.2 == anno {
            ris = append(ris, nome)
        }
    }
    return ris
}

m[nome].d
fmt.Printf("%d/%d/%d",
    m[nome].g, m[nome].m,
    m[nome].2)
```

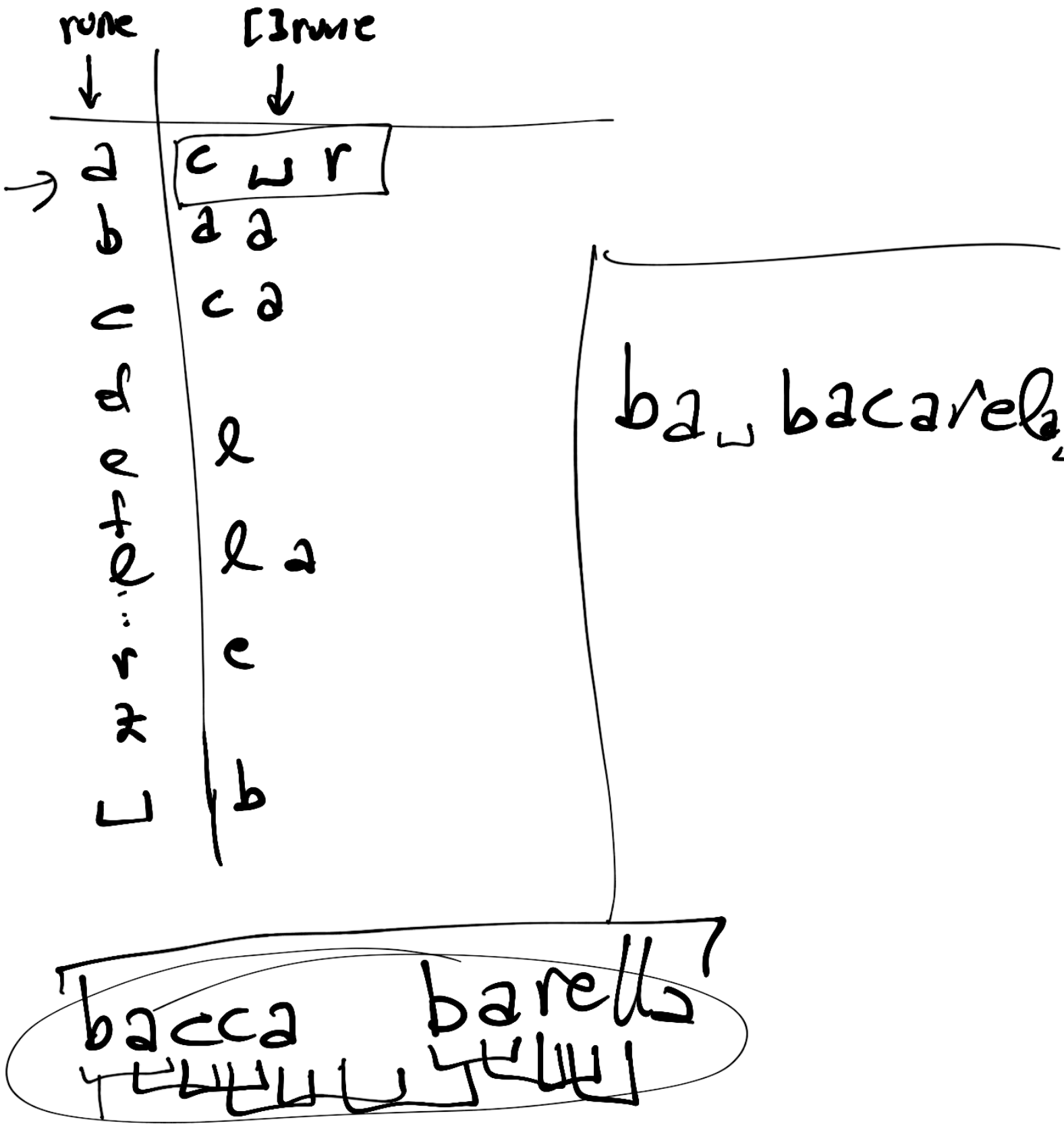
GENERATORE MARKOVIANO DI TESTI

ptzra

metempsicosi



bef


rune
[]rune
a
c ␣ r
b
a a
c
c a
d
e
l
f
l a
r
e
z
␣
b
ba␣ bacarella
bacca
barella

```
type markov map[rune][]rune
//Associa ad ogni runa le rune che
//possono seguirla, eventualmente con
//ripetizioni

func aggiorna(s string, m markov){
    var prev := ' '
    for _, c := range s {
        m[prev]=append(m[prev], c)
        prev=c
    }
}
```

./markov 17° < divinaCommedia.txt

```
func main() {
    var m markov
    m = make(map[rune][3rune)
      = make(markov)
      = markov{}
    scanner := bufio.NewScanner(
                    os.Stdin)
    for scanner.Scan(){
        line := scanner.Text()
        aggiorna(line, m)
    }
    n, _ := strconv.Atoi(os.Args[1])
    markovGen(m, n)
}
```

④ ③ ⑤

```
func markovGen(m markov, n int) {
    c := ' '
    for i:=0; i<n; i++ {
        k := len(m[c])
        j := rand.Intn(k)
        r := m[c][j]
        fmt.Printf("%c", r)
        c = r
    }
}
```

DICHIARATE DUE TIPI

- prescrizione
  - → nome di un farmaco
  - → dosaggio giornaliero
- paziente
  - → nome
  - → cognome
  - → lista di prescrizioni

```
type (
    prescrizione  struct {
                      farmaco   string
                      dosaggio  int
                  }

    paziente      struct {
                      nome      string
                      cognome   string
                      p         []prescrizione
                  }
)
```

SCRIVETE UNA FUNZIONE CHE
DATA UNA SLICE DI PAZIENTI
E UNA MAPPA CHE AD OGNI
FARMACO ASSOCIA IL DOSAGGIO
MASSIMO GIORNALIERO,
RESTITUISCE LA SLICE DEI
PAZIENTI CHE RISCHIANO
L'OVERDOSE

```
func overdose(paz []paziente,
    doseMax map[string]int) []paziente {

    var ris []paziente
→   for _, p := range paz {
→       for _, ricetta := range p.p {
            nomeFarmaco := ricetta.farmaco
            dose := ricetta.dosaggio
            max, ok := doseMax[nomeFarm]
            if ok && dose >= max {
                ris = append(ris, p)
                break
            }
        }
    }
    return ris
}
```

SCRIVETE UNA FUNZIONE
CHE DATA UNA SLICE
DI PAZIENTI E UNA
MAPPA CHE DI OGNI
FARMACO DICE QUALI
SONO I FARMACI
INCOMPATIBILI, STABILISCA
L'ELENCO DEI PAZIENTI
CHE MUOIONO