put will do one of three possible actions:
1. update the value of an existing entry;
2. add a new entry but not change top;
3. add a new entry and change top and possibly add new levels.

```
        B                          O             @T

        B     F  G                 O              T  U

        B  C  F  G        K     M  O     Q        T  U  W     Z

     A  B  C  F  G  I  J  K  L  M  O  P  Q  R  S  T  U  W  X  Z
     4  3  1  8  6  7  0  2  9  0  8  4  6  3  5  6  3  4  2  0
```

Remember any entry in the top level list could be top.

```
        B                          O             @T

        B     F  G                 O              T  U

        B  C  F  G        K     M  O     Q        T  U  W     Z

     A  B  C  F  G  I  J *K  L  M  O  P  Q  R  S  T  U  W  X  Z
     4  3  1  8  6  7  0  2  9  0  8  4  6  3  5  6  3  4  2  0
```

1. put(K, 3) calls rFind(@, K) and gets back a bottom entry (*).

```
        B                          O             @T

        B     F  G                 O              T  U

        B  C  F  G        K     M  O     Q        T  U  W     Z

     A  B  C  F  G  I  J *K  L  M  O  P  Q  R  S  T  U  W  X  Z
     4  3  1  8  6  7  0  3  9  0  8  4  6  3  5  6  3  4  2  0

1. put(K, 3) sets its value.
```

```
        B                          O             @T

        B     F  G                 O              T  U

        B  C  F  G        K     M  O     Q        T  U  W     Z

     A  B  C  F  G  I  J  K  L  M  O  P  Q  R  S  T  U  W  X  Z
     4  3  1  8  6  7  0  3  9  0  8  4  6  3  5  6  3  4  2  0
```

2. put(N, 6) calls rFind(*, N) and gets back null.

```
        B                             O             @T

        B     F  G                    O              T  U

        B  C  F  G        K     M  N  O     Q        T  U  W     Z

     A  B  C  F  G  I  J  K  L  M  N  O  P  Q  R  S  T  U  W  X  Z
     4  3  1  8  6  7  0  2  9  0  6  8  4  6  3  5  6  3  4  2  0
```

2. put(N, 6) calls rAdd(*, N, 6) and gets back null because N did not three heads in a row (see rAdd.pdf).

```
        B                             O             @T

        B     F  G                    O              T  U

        B  C  F  G        K     M  N  O     Q        T  U  W     Z

     A  B  C  F  G  I  J  K  L  M  N  O  P  Q  R  S  T  U  W  X  Z
     4  3  1  8  6  7  0  2  9  0  6  8  4  6  3  5  6  3  4  2  0
```

3. put(D, 7) calls rFind(*, D) and gets back null.

```
        B    +D                          O             @T

        B     D  F  G                    O              T  U

        B  C  D  F  G        K     M  N  O     Q        T  U  W     Z

     A  B  C  D  F  G  I  J  K  L  M  N  O  P  Q  R  S  T  U  W  X  Z
     4  3  1  7  8  6  7  0  2  9  0  6  8  4  6  3  5  6  3  4  2  0
```

3. put(D, 7) calls rAdd(*, D, 7) and gets back a new Entry (+) because D was lucky and threw three heads in a row.

```
        B    @D                          O              T

        B     D  F  G                    O              T  U

        B  C  D  F  G        K     M  N  O     Q        T  U  W     Z

     A  B  C  D  F  G  I  J  K  L  M  N  O  P  Q  R  S  T  U  W  X  Z
     4  3  1  7  8  6  7  0  2  9  0  6  8  4  6  3  5  6  3  4  2  0

3. put(D, 7) makes this new Entry the top.  put starts flipping a coin.
```

```
             @D

        B     D                          O              T

        B     D  F  G                    O              T  U

        B  C  D  F  G        K     M  N  O     Q        T  U  W     Z

     A  B  C  D  F  G  I  J  K  L  M  N  O  P  Q  R  S  T  U  W  X  Z
     4  3  1  7  8  6  7  0  2  9  0  6  8  4  6  3  5  6  3  4  2  0
```

3. put(D, 7) flips heads, so it creates a new Entry whose value is the old top and makes that Entry the new top.

```
             @D

              D

        B     D                          O              T

        B     D  F  G                    O              T  U

        B  C  D  F  G        K     M  N  O     Q        T  U  W     Z

     A  B  C  D  F  G  I  J  K  L  M  N  O  P  Q  R  S  T  U  W  X  Z
     4  3  1  7  8  6  7  0  2  9  0  6  8  4  6  3  5  6  3  4  2  0
```

3. put(D, 7) flips heads, so it creates a new Entry whose value is the old top a nd makes that Entry the new top.

```
             @D

              D

        B     D                          O              T

        B     D  F  G                    O              T  U

        B  C  D  F  G        K     M  N  O     Q        T  U  W     Z

     A  B  C  D  F  G  I  J  K  L  M  N  O  P  Q  R  S  T  U  W  X  Z
     4  3  1  7  8  6  7  0  2  9  0  6  8  4  6  3  5  6  3  4  2  0
```

3. D’s luck finally runs out: put(D, 7) flips tails.