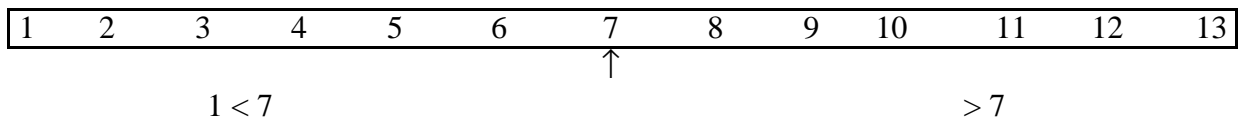


## Binäre Suche:



```

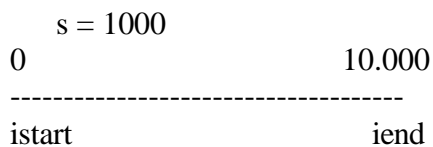
for ( ;istart <= iend; )
{
    imid = (istart + iend) / 2;
    if (f [imid] == s)
        return imid;
    else if (f [imid] >= s)
        iend = imid - 1;
    else istart = imid + 1;
}

```

Img\_0003.jpg return - 1;

## Interpolatorische Suche:

Idee



### *Schlüssel gleichverteilt*

Nächster Suchindex

$$\frac{\text{inext} - \text{istart}}{\text{iend} - \text{istart}} = \frac{s - f[\text{istart}]}{f[\text{iend}] - f[\text{istart}]}$$

“Optimales” Suchverfahren wäre gegeben, wenn Schlüsselposition (index) = Schlüssel gilt.  
Suche nach s:

Falls f [s] „leer“ → Suche erfolglos  
sonst Ergebniss f [s]

Vor.:

1. Integer Schlüssel  
(Schlüssel = Index)

Chaning / Closed hashing

B = 7

{2, 9, 3, 10, 16, 4}

separate chaining / closed hashing

		Nachfolger	Start
0	4	-1	-1
1		-1	-1
2	2	3	2
3	9	6	4
4	3	5	0
5	10	-1	-1
6	16	-1	-1

		Nachfolger
0	16	-1
1		-1
2	2	5
3	3	6
4	4	-1
5	9	0
6	10	1

open hashing

(Kollisionsbehandlung ausserhalb der Hashtabelle)

{2, 9, 3, 10, 16, 4}

0		
1		
2	2	Verkettete Liste
3	3	
4	4	
5		
6		

1. Offener „Container“ für  
Hashtabelle (sequentiell  
zu durchsuchen)

Alternativ:  
Pro Index einOverflow Container