Perfect Hash Functions

In most general applications, we cannot know exactly what set of key values will need to be hashed until the hash function and table have been designed and put to use.

At that point, changing the hash function or changing the size of the table will be extremely expensive since either would require re-hashing every key.

A *perfect hash function* is one that maps the set of actual key values to the table without any collisions.

A *minimal perfect hash function* does so using a table that has only as many slots as there are key values to be hashed.

If the set of keys IS known in advance, it is possible to construct a specialized hash function that is perfect, perhaps even minimal perfect.

Algorithms for constructing perfect hash functions tend to be tedious, but a number are known.

Cichelli's Method

This is used primarily when it is necessary to hash a relatively small collection of keys, such as the set of reserved words for a programming language.

The basic formula is:

```
h(S) = S.length() + g(S[0]) + g(S[S.length()-1])
```

where `g()` is constructed using Cichelli's algorithm so that h() will return a different hash value for each word in the set.

The algorithm has three phases:

- computation of the letter frequencies in the words
- ordering the words
- searching

Cichelli's Method

Suppose we need to hash the words in the list below:

```
calliope
clio
erato
euterpe
melpomene
polyhymnia
terpsichore
thalia
urania
```

Determine the frequency with which each first and last letter occurs:

letter:	e	a	c	o	t	m	p	u
freq:	6	3	2	2	2	1	1	1

Score the words by summing the frequencies of their first and last letters, and then sort them in that order:

calliope	8
clio	4
erato	8
euterpe	12
melpomene	7
polyhymnia	4
terpsichore	8
thalia	5
urania	4

```
euterpe
calliope
erato
terpsichore
melpomene
thalia
clio
polyhymnia
urania
```

Cichelli's Method

Finally, consider the words in order and define `g(x)` for each possible first and last letter in such a way that each of the words will have a distinct hash value:

word	g_value assigned	h(word)	table slot	
euterpe	e-->0	7	7	ok
calliope	c-->0	8	8	ok
erato	o-->0	5	5	ok
terpsichore	t-->0	11	2	ok
melpomene	m-->0	9	0	ok
thalia	a-->0	6	6	ok
clio	none	4	4	ok
polyhymnia	p-->0	10	1	ok
urania	u-->0	6	6	reject
	u-->1	7	7	reject
	u-->2	8	8	reject
	u-->3	9	0	reject
	u-->4	10	1	reject

Cichelli's Method

Cichelli's method imposes a limit on the search at this point (we're assuming it's 5 steps), and so we back up to the previous word and redefine the mapping there:

word	g_value assigned	h(word)	table slot
polyhymnia	p-->0	10	1 reject
	p-->1	11	2 reject
	p-->2	12	3
urania	u-->0	6	6 reject
	u-->1	7	7 reject
	u-->2	8	8 reject
	u-->3	9	0 reject
	u-->4	10	1 ok

So, if we define `g()` as determined above, then `h()` will be a minimal perfect hash function on the given set of words.

The primary difficulty is the cost, because the search phase can degenerate to exponential performance, and so it is only practical for small sets of words.