CS 310

Assignment 408

Brandon Ingli

Problem 1. Draw the memo table and memok helper table for the RNA substructure problem using the RNA sequence CAGUCAUAGCAAUGCU, and indicate which letters match in the secondary structure.

Answer: The memo table appears as

			j											
			5 C	6 A	7 U	8 A	9 G	10 C	11 A	12 A	13 U	14 G	15 C	16 U
	0	–	0	0	0	0	0	0	0	0	0	0	0	0
	1	C	0	0	1	1	2	2	2	2	3	4	4	5
	2	A	0	0	1	1	1	2	2	2	3	3	4	5
	3	G	0	0	0	0	1	1	2	2	2	3	4	4
	4	U	0	0	0	1	1	1	2	2	2	3	3	3
	5	C	0	0	0	0	1	1	1	2	2	3	3	3
i	6	A		0	0	0	0	0	0	1	2	2	2	3
	7	U			0	0	0	0	1	1	1	1	2	3
	8	A				0	0	0	0	0	1	1	2	3
	9	G					0	0	0	0	0	0	2	2
	10	C						0	0	0	0	1	1	1
	11	A							0	0	0	0	0	1
	12	A								0	0	0	0	1

and the helper memo table appears as

			j											
			5 C	6 A	7 U	8 A	9 G	10 C	11 A	12 A	13 U	14 G	15 C	16 U
	0	–												
	1	C			2		1	3	4	4	2	1	3	2
	2	A			2			3	4	4	2	5	3	2
	3	G						3	4	4	6	5	3	
	4	U				4			4	4	6	5	9	6
	5	C					5			7	6	5		6
i	6	A								7	6		9	6
	7	U							7	7	8		9	8
	8	A									8		9	8
	9	G											9	
	10	C										10		11
	11	A												11
	12	A												12

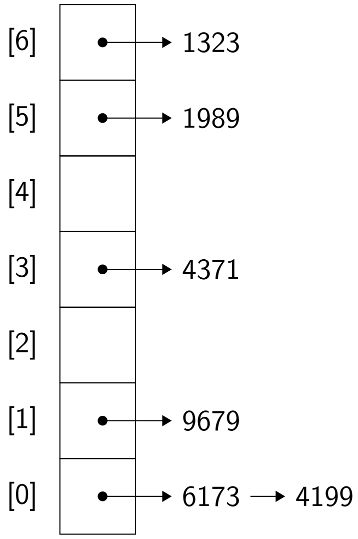
Any values not filled in can be assumed to be 0.

The following letters match up in the RNA Secondary Structure:

- (2 - A, 16 - U)
- (3 - G, 15 - C)
- (5 - C, 14 - G)
- (6 - A, 13 - U)
- (7 - U, 12 - A)

Problem 2. Given the input 4371, 1323, 6173, 4199, 9679, 1989 for an initially empty hash table, and a hash function $h(x) = 6 - (x \bmod 7)$, draw the resulting hash table. State and explain any assumptions you make, and explain your resulting table.

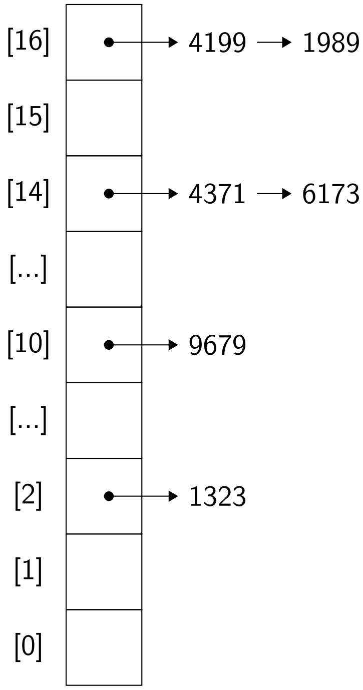
Answer:



From the given hash function, we are assuming a table of size 7, as well as open chaining to deal with collisions. The given hash function appears to suit the data quite well, as the members are quite evenly distributed. One collision occurred at index 0. This table is more than 70% full.

Problem 3. Rehash the table of problem 2, and show the table that results from the rehashing. Explain your process and results.

Answer: The next prime number more than twice the size of our current table is 17, so that will be the size of our new table. Keeping with the spirit of the previous hash function, the new function will be $h(x) = 16 - (x \bmod 17)$. After running each element of the previous table through the new hash function and placing them in the table, it appears as shown below. For the given data, there is one more collision in the rehash than in the original table.



Problem 4. Write a program to implement the given hash function and count the number of collisions that would occur in a hash table for words in the Unix dictionary. Approximate a load factor of 1, and read the words from standard input.

```
size_t hash(const string& key, size_t table_size)
{
   size_t hash_val {0};
   for (auto character : key)
   {
      hash_val = 37 * hash_val + static_cast(character);
   }
   return hash_val % table_size;
}
```

Answer: Please see that attached program. The number of collisions detected on my system with 102305 dictionary words was 37633.

Problem 5. Explain and justify your results. Be sure to explain what your table size is, and why you chose that value.

Answer: To approximate a load factor of 1 with 102305 words, I chose a table size of 102301, as this is the closest prime number to the number of elements to be placed in the table. About 37% of the words pushed onto the hash table experienced a collision, which can be attributed to the fact that the use of letters in the English language is not evenly distributed. The data is still somewhat spread out due to the hash function taking the length of the word as a major factor in the hash value. While this too is not very evenly distributed, the length in combination with letter choice helps in spreading the data over the table.