CS211 ALGORITHMS & DATA STRUCTURES II

LAB 7

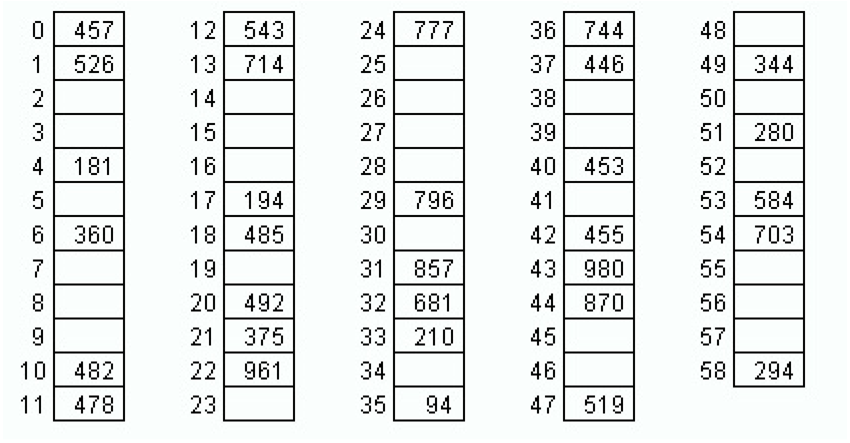
Dr. Phil Maguire

HASHING

Pen and paper exercise

Insert the values 444, 669 and 880 into the following hash table using the following algorithms. The double has function returns a value between 1 and 15.

i) Linear Probing
ii) Quadratic Probing
iii) Double Hashing



0	457	12	543	24	777	36	744	48	
1	526	13	714	25		37	446	49	344
2		14		26		38		50	
3		15		27		39		51	280
4	181	16		28		40	453	52	
5		17	194	29	796	41		53	584
6	360	18	485	30		42	455	54	703
7		19		31	857	43	980	55	
8		20	492	32	681	44	870	56	
9		21	375	33	210	45		57	
10	482	22	961	34		46		58	294
11	478	23		35	94	47	519		

Programming exercise

Download the files Hashing.java and dictionary.txt. There are around 216.5K words in the file and the objective of the lab is to put these words into a hash table with the minimum number of collisions.

Compile the file Hashing.java. Look through the code carefully to get a feel for what's going on. There are three methods for resolving collisions. However, the methods that compute the primary and secondary hash keys are not properly implemented. Once all the words are loaded into the hash table, the user is asked to input a word to search for. The program exits when the user types ‘quit’.

Objective 1:

To get the system working properly you need to create a decent hash function. Modify the method getHashKey( ) so that it creates a unique number for each word and then hashes it into the appropriate range. Use the modPow( ) method if you are using modulo with large powers. The hash function should always produce a value between 0 and the hash table size. Test to see if the number of collisions has been reduced.

Objective 2:

If you want to use double hashing then you need a double hash function. Modify the method getDoubleHashKey( ) so that it is based on more characters than just the last one. Remember, the important features of a secondary hash function are that it is different to the primary hash function (so that different items colliding into a particular slot will follow different jumps) and never produces a 0 (so that it doesn’t keep checking the same slot forever). The number that is returned by the function is the size of the jumps that will be taken in case of a collision. Test to see if the number of collisions has been reduced. For full marks, get the number of collisions below 90,000, while keeping the load factor above 0.5.

Advanced Programming exercise

See how low you can get the number of collisions by trying different hash table sizes and different open addressing systems. The more unique your hash functions, the fewer the number of collisions that will result. Whoever gets the lowest number of collisions with a load factor of at least 0.9 will earn a bonus 2% CA.