Programming Assignment 1: Brief Overview

First: Array Sorting

- Sorts an input array
- Input
  - First line specifies number of array elements
  - Second line specifies the array
- Output
  - Odds in ascending order, Evens in descending order
- Example:

Input: [1 8 12 3 5 8 2 4 -5]

Sorted Array: [-5 1 3 5 12 8 8 4 2]

Second: Hash Table

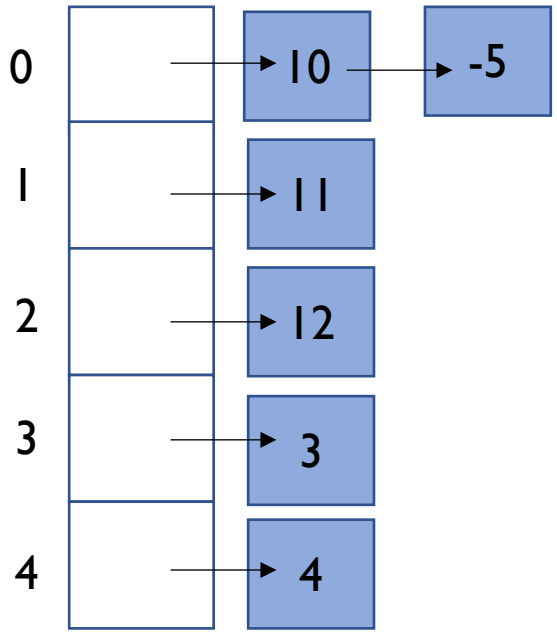
- Hash Table with 10,000 buckets
- Given hash table operations, manipulate hash table
  - i <value> : insert into hash table
    - prints "inserted" if value successfully inserted into hash table
    - prints "duplicate" if value already exists in hash table
  - s <value> : search hash table
    - prints "present" if value is in hash table
    - prints "absent" if value is NOT in hash table
- Hash is <value> modulo <# of buckets>
- Don't forget about negative values
  - A negative value might yield you a negative hash value
  - Ex. -5 % 2 = -1, there is no such index as -1

Second: Hash Table (Example)

- Example with 5 buckets

example_file.txt	output
i 10	inserted
i 11	inserted
i 3	inserted
i 4	inserted
i 12	inserted
s 10	present
i 10	duplicate
s 4	present
i -5	inserted
s 5	absent

Hash Table



Third: Bit Function

- Take a value X and perform bit operations on it
- Operations:
  - set <n> <v> : Set the nth bit of X to v and print the value
  - get <n> <v> : Get the value of the nth bit and print it (eg. 1 or 0)
  - comp <n> <v> : Modify the nth bit to its complement (eg. 1 if 0 or 0 if 1) and print the resulting X value
- Input
  - First line specifies the initial value X
  - Rest of lines specifies the bit operations to carry out
- Only use bitwise and assignment operators, no arithmetic
- Example:

example_file.txt	state	output
• 5	• 5 (0101)	• <none>
• get 0 0	• 0101	• 1
• comp 0 0	• 0101 -> 0100 = 4	• 4
• set 1 1	• 0100 -> 0110 = 6	• 6

Fourth: One Shot Learning

- Carry out One-Shot Learning in C to predict the price of an unknown house
- Input:
  - File of training data with prices and attributes for a house in each row
  - File of attributes unknown houses
- Output:
  - The predicted prices for each of the unknown houses

The Problem

- Someone out there is pricing houses based on several attributes
  - Ex. # of bedrooms, total size of house, # of baths, etc.
  - $x_1, x_2, x_3, x_4, \ldots.$
- All these attributes contribute to the final house price
- However all attributes may not contribute to the final price the same way
  - Not all attributes have the same weight on the price
- Would look more like the following:
  - $price = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + \ldots$
  - Each attribute x_i has some weight w_i
  - Note: x_0 is implicitly 1 as w_0 would basically serve as a constant

The Problem

- Given some houses we do know the attributes and prices of, can we predict what the potential price of an unknown house?
- Example:
  - Recall: $price = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + \ldots$

House Price	Price Constant	# bedrooms	# sq ft	# bathrooms
350,000	1	4	3000	1
200,000	1	2	2000	1
577,000	1	5	8000	3
400,000	1	3	5000	2
300,000	1	3	4000	2
?	1	3	5000	3

House Price		Price Constant		# bedrooms		# sq ft		# bathrooms
350,000	=	w_0 1	+	w_1 4	+	w_2 3000	+	w_3 1
200,000	=	w_0 1	+	w_1 2	+	w_2 2000	+	w_3 1
577,000	=	w_0 1	+	w_1 5	+	w_2 8000	+	w_3 3
400,000	=	w_0 1	+	w_1 3	+	w_2 5000	+	w_3 2
300,000	=	w_0 1	+	w_1 3	+	w_2 4000	+	w_3 2
?	=	w_0 1	+	w_1 3	+	w_2 5000	+	w_3 3

- Can we learn weights $w_0,\ w_1,\ w_2, and\ w_3$ to predict the unknown house price?

The Problem Generalized

- Lets solve the weights with what we know

House Price	Price Constant	# bedrooms	# sq ft	# bathrooms
350,000	1	4	3000	1
200,000	1	2	2000	1
577,000	1	5	8000	3
400,000	1	3	5000	2
300,000	1	3	4000	2

- Represent attributes and prices as matrices
- Let matrix X be a matrix of all the attributes we know
  - Each row corresponds to the attributes of a house
  - The ith column corresponds to the ith attribute
- Let W be a vector of all weights
- Let Y be the house prices

$$\begin{bmatrix} 1 & 4 & 3000 & 1 \\ 1 & 2 & 2000 & 1 \\ 1 & 5 & 8000 & 3 \\ 1 & 3 & 5000 & 2 \\ 1 & 3 & 4000 & 2 \end{bmatrix} \quad \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} \quad = \quad \begin{bmatrix} 350{,}000 \\ 200{,}000 \\ 577{,}000 \\ 400{,}000 \\ 300{,}000 \end{bmatrix}$$

X W Y

Solution

- We have the equation $XW = Y$
- We know X and Y
- We need to solve with for W
- We can do this using the following equation:
  - $W = (X^T X)^{-1} X^T Y$
- Method of Least Squares
  - Do not worry to much about the details
  - If interested look here: https://en.wikipedia.org/wiki/Linear_least_squares
- All you need to know is that it will approximate W
- Once we know W we can use it to predict the price of any house

Things you need to know

- Solving for the weights:
  - $W = (X^T X)^{-1} X^T Y$
- Predicting Price of Unknown House:
  - $Y = XW$
- Matrix Multiplication
  - https://www.mathsisfun.com/algebra/matrix-multiplying.html
  - https://www.khanacademy.org/math/precalculus/x9e81a4f98389efdf:matrices/x9e81a4f98389efdf:multiplying-matrices-by-matrices/v/matrix-multiplication-intro
- Transposing A Matrix
  - https://mathinsight.org/matrix_transpose
  - https://www.khanacademy.org/math/linear-algebra/matrix-transformations/matrix-transpose/v/linear-algebra-transpose-of-a-matrix
- Inverting a Matrix
  - https://www.mathsisfun.com/algebra/matrix-inverse-row-operations-gauss-jordan.html
  - https://www.khanacademy.org/math/algebra-home/alg-matrices/alg-determinants-and-inverses-of-large-matrices/v/inverting-matrices-part-3