Storing and retrieving of data

• Simple scenario
  – Store item in data structure
  – Fetch item later in data structure

• Operations
  – Insert
  – Find

• For simplicity: data is string (e.g., name)
  – If name is not stored, find returns null
Storing data in arrays

• Arrays/tables are convenient data structures
  – O(1) access time to elements
• Ordering in arrays may be difficult
  – E.g., insertion requires shifting of other elements

Storing data in arrays

• Alternate approach:
  – Do not “compact” array
  – Assign spot to each element over entire array
Storing data in arrays

• How to determine spot for each element?

- Table for all possible values too large
- Need to leave space for future entries
- Need to handle multiple names in same spot
Hash table

• Index for element computed by hash function
  – E.g., maps string to integer
  – Range matches size of allocated table
  – Does not need to maintain ordering

• Example:

Hash function

• How to map String to integer?
Hash function

• How to map String to integer?
• Simple example in range 0...size-1:

```java
private int hash(String s) {
    char[] c = s.toCharArray();
    int h=0;
    for (int i=0; i< c.length; i++) {
        h+=Character.getNumericValue(c[i]);
    }
    return h%size;
}
```

Collisions

• What to do when multiple items have the same hash value?
Collisions

• What to do when multiple items have the same hash value?

• Collision resolution techniques
  – Store multiple items in same place (next lecture)
  – Probing (check items following hash index)
    • Linear probing
    • Quadratic probing
    • Etc.

Linear Probing

• Insertion
  – Compute hash value
  – Check if table cell is empty at hash value
  – If not, keep checking next cell until empty spot found
  – Insert

• Find
  – Compute hash value
  – Check if table cell at hash value matches search term
    • If table cell empty, return null
    • If cell entry does not match, continue searching until match or null
  – Return match or null
Code

• Hash table:

```java
public class HashTable {
    private String[] table;
    private int size;

    public HashTable(int n) {
        table = new String[n];
        size = n;
    }
```

• Insert:

```java
public void insert(String s) {
    int h = hash(s); // compute hash of item
    while (table[h]!=null) { // find empty spot in table
        h=(h+1)%size; // if occupied, increment by 1
    }
    table[h]=s; // enter item into table
}
```
Code

• Find:

```java
public String find(String s) {
    int h = hash(s); // compute hash of item
    while (table[h] != null && !table[h].equals(s)) {
        // find matching item or null
        h = (h + 1) % size; // if no match, increment by 1
    }
    return table[h]; // return item or null
}
```

Next Steps

• Lecture on Wednesday
• Homework 6
  – Deadline Wednesday 11 p.m. (hard deadline Thursday 11 p.m.).