def fastcount(words):
    d = {}
    for w in words:
        if w in d:
            d[w] += 1
        else:
            d[w] = 1
    return sorted(d.items())
Q is for …

- QR code
  - Black and white and read all over
- Quicksort
  - Sort of choice before Timsort?
- QWERTY
  - When bad ideas persist
Christine Alvarado

• Teaching Professor, UCSD
• PhD Computer Science, MIT
• Her work is in designing CS curriculum that is more accessible and more appealing to all
• LogiSketch – draw and simulate digital circuits

“It’s important to choose your own path, and try not to compare yourself to others. You have your own unique circumstance, so what others do or don’t do shouldn’t really affect your life.”
Announcements

• Assignment 4 GuessWord due today!
• APT-5 due Thur, March 24
  • Recommend to do before Exam 3
• Lab 8 Friday, do prelab
• Assignment 5 due March 29
• Exam 2 regrades by March 23
• Exam 2 booster – 8 pts - Take by March 23
• Mid-Semester Survey going out
  • How are we doing, How are UTAs doing, etc
  • 2 extra pts on Exam 3 if 75% of you fill it out!
Exam 3 – in person – Tues, March 22

- Exam is in class on paper – 10:15am
  - Need pen or pencil
- See materials under 3/22 date
  - Exam 3 Reference sheet - part of exam
- Covers
  - Topics: sets, parallel lists, dictionaries, sorting, tuples, (No images)
  - APTs through APT5
  - Labs through Lab 8
  - Assignments through Assignment 4
  - Sakai Quizzes through 3/17
PFTD

• Dictionaries
  • More Practice
  • Fast!

• Family APT

• Clever GuessWord
Finish up Jotto

• Last time I made an error:
  • Didn’t test updateWordList
• Bad code below. What does this do?

```python
def updateWordList(words, numInCommon, userword):
    return [w for w in words if commonCount(w, userword)]
```

Not a Boolean but an integer
Not flagged as an error!!
Treating positive numbers as True and 0 as false
YOU DO NOT NEED TO KNOW THIS!
Finish up Jotto

• Last time I made an error:
  • Didn’t test updateWordList
• Bad code below. What does this do?

```python
def updateWordList(words, numInCommon, userword):
    return [w for w in words if commonCount(w, userword)]
```

• Not a Boolean but an integer
• Not flagged as an error!!
• Treating positive numbers as True and 0 as false
• YOU DO NOT NEED TO KNOW THIS!
Finish up Jotto (2)

• Correct code for `updateWordList`

```python
def updateWordList(words, numInCommon, userword):
    return [w for w in words
            if commonCount(w, userword) == numInCommon]
```

• Now run Jotto. Computer wins a lot!

Guess a word with 5 letters: `beach`
next word is: `femur`
num letters in common is: 1
words remaining: 2534

words remaining: 2
next word is: `beach`
num letters in common is: 5
words remaining: 1
I won! :D
Dictionary Iteration (unordered!)

- Iterate through keys:
  - `for k in d:
  - for k in d.keys():`

- Iterate through pairs:
  - `for (k,v) in d.items():`
  - `for k,v in d.items():`
Sorting a list from dictionary - sorted()

d = {'k': 3, 'h': 8, 'a': 12, 'd': 5}

x = sorted(d.keys())
y = sorted(d.values())
z = sorted(d.items())
Sorting a list from dictionary - sorted()

d = {'k': 3, 'h': 8, 'a': 12, 'd': 5}

x = sorted(d.keys())  
x is ['a', 'd', 'h', 'k']

y = sorted(d.values())  
y is [3, 5, 8, 12]

z = sorted(d.items())  
z is [('a', 12), ('d', 5), ('h', 8), ('k', 3)]
WordFrequencies
Dictionary Example

• Let’s see an example that compares using a dictionary vs not using a dictionary
slowcount function
Short Code and Long Time

• See module WordFrequencies.py
  • Find # times each word in a list of words occurs
  • We have tuple/pair: word and word-frequency

```python
37  def slowcount(words):
38    pairs = [(w,words.count(w)) for w in set(words)]
39    return sorted(pairs)
```

• Think: How many times is `words.count(w)` called?
  • Why is `set(words)` used in list comprehension?
WordFrequencies with Dictionary

• If start with a million words, then...
• We look at a million words to count # "cats"
  • Then a million words to count # "dogs"
  • Could update with parallel lists, but still slow!
  • Look at each word once: dictionary!

• Key idea: use word as the "key" to find occurrences, update as needed
  • Syntax similar to `counter[k] += 1`
Using fastcount

• Update count if we've seen word before
  • Otherwise it's the first time, occurs once

```python
def fastcount(words):
    d = {}
    for w in words:
        if w in d:
            d[w] += 1
        else:
            d[w] = 1
    return sorted(d.items())
```
Using fastcount

• Update count if we've seen word before
  • Otherwise it's the first time, occurs once

```python
def fastcount(words):
    d = {}
    for w in words:
        if w in d:
            d[w] += 1
        else:
            d[w] = 1
    return sorted(d.items())
```
Let’s run them and compare them!

• Run with Melville and observe time

• Run with Hawthorne and observe time
Let’s run them and compare them!

- Run with Melville and observe time
  - slowcount about 0.76 seconds
  - fastcount about 0.00 seconds

- Run with Hawthorne and observe time
  - slowcount about 14.6 seconds
  - fastcount about 0.03 seconds
APT Family

APT: Family

Problem Statement

You have two lists: parents and children. The ith element in parents is the parent of the ith element in children. Count the number of grandchildren (the children of a person's children) for the person in the person variable.

Hint: Consider making a helper function that returns a list of a person's children.
Step 1: work an example by hand

```python
parents = ['Junhua', 'Anshul', 'Junhua', 'Anshul', 'Kerry']
children = ['Anshul', 'Jordan', 'Kerry', 'Paul', 'Kai']
person = 'Junhua'

Returns 3
```
Step 1: work an example by hand

parents = ['Junhua', 'Anshul', 'Junhua', 'Anshul', 'Kerry']
children = ['Anshul', 'Jordan', 'Kerry', 'Paul', 'Kai']
person = 'Junhua'

Returns 3

- First find the children of Junhua
  - Loop over parents list
    - If name is Junhua add corresponding child to list
      - How do I do that? I need an index (parallel lists)
    - Kids are ['Anshul', 'Kerry']
  - For each kid:
    - Loop over parents list:
      - If name is kid’s name add their child to the list
        » How do I do that? I need an index (parallel lists)
      - ‘Anshul’s kids -> ‘Jordan’ and ‘Paul’
      - Kerry’s kids -> ‘Kai’
  - Return 3
Step 1: work an example by hand

```python
parents = ['Junhua', 'Anshul', 'Junhua', 'Anshul', 'Kerry']
children = ['Anshul', 'Jordan', 'Kerry', 'Paul', 'Kai']
person = 'Junhua'
```

Returns 3

Notice anything?

- First find the children of Junhua
  - Loop over parents list
    - If name is *Junhua* add corresponding child to list
      - How do I do that? I need an index (parallel lists)
    - Kids are ['Anshul', 'Kerry']
  - For each kid:
    - Loop over parents list:
      - If name is *kid’s name* add their child to the list
        » How do I do that? I need an index (parallel lists)
    - ‘Anshul’s kids -> ‘Jordan’ and ‘Paul’
    - Kerry’s kids -> ‘Kai’
  - Return 3

They are the same!

Write a helper function!
Helper function

def childrenOf(parents, children, name):
    <missing code to traverse parallel lists>
    return list of name’s children
How to traverse parallel lists?

parents: ['Junhua', 'Anshul', 'Junhua', 'Anshul', 'Kerry']
children: ['Anshul', 'Jordan', 'Kerry', 'Paul', 'Kai']

0 1 2 3 4
How to traverse parallel lists?

parents: ['Junhua', 'Anshul', 'Junhua', 'Anshul', 'Kerry']
children: ['Anshul', 'Jordan', 'Kerry', 'Paul', 'Kai']

Iterate over the list – need a loop!
Need to access same position in each list
- need an index

Use a while loop with an index!
How to traverse parallel lists?

parents: ['Junhua', 'Anshul', 'Junhua', 'Anshul', 'Kerry']
children: ['Anshul', 'Jordan', 'Kerry', 'Paul', 'Kai']

index = 0
while index < len(parents):
    <do something>
    index += 1  # update index
Assignment 5 - How to play
Guess Word Cleverly

• Make it hard for the player to win!

• One way: Try hard words to guess?
  • "jazziest", "joking", "bowwowing"

• Another Way: Keep changing the word, sortof 😊
Clever GuessWord

• Current GuessWord: Pick random secret word
  • User starts guessing

• Can you change secret word?
  • Yes, but must have letters in same place you have told user
    • Change consistent with all guesses
  • Make the user work harder to guess!
Programming A Clever Game

• Instead of guessing a word, you're guessing a group, category, or equivalence class of words
  Ex: _ _ _ _ _ _ and user guesses 'a'

• "asked", "adult", "aided", ... "axiom"
  • 209 words 'a' as first letter and the only 'a'
• "baked", "cacti", "false", ... "walls"
  • 665 words 'a' as second letter and the only 'a'
• "beets", "humor", ... "spoof"
  • 2,431 words with no 'a'

• What should our secret word be? "asked", "baked" or "beets"?
Programming A Clever Game

• Instead of guessing a word, you're guessing a group, category, or equivalence class of words
  Ex: _ _ _ _ _ _ and user guesses 'a'

• ["asked", "adult", "aided", ... "axiom"]
  • 209 words 'a' as first letter and the only 'a'

• ["baked", "cacti", "false", ... "walls"]
  • 665 words 'a' as second letter and the only 'a'

• ["beets", "humor", ... "spoof"]
  • 2,431 words with no 'a'

• What should our secret word be? "asked", "baked" or "beets"?
  Tell user there is no ‘a’
Sometimes there will be letters

- The letter “u” has been guessed and is the 2nd letter
  Ex: _ u _ _ _ _ and user guesses ‘r’

- ["ruddy", "rummy", "rungs", … "rusty"]
  - 5 words start with “ru” and no other “r” or “u”

- ["burch", "burly", "burns", … "turns"]
  - 17 words only ‘u’ as second letter and only ‘r’ third letter

- ["bucks", "bucky", … "tufts"]
  - 98 words with only “u” second letter and no ‘r’

- What should our secret word be? "ruddy", "burch" or "bucks“?
Sometimes there will be letters

- The letter “u” has been guessed and is the 2nd letter
  Ex: _ u _ _ _ and user guesses ‘r’

- ["ruddy", "rummy", "rungs", ... "rusty"]
  - 5 words start with “ru” and no other “r” or “u”
- ["burch", "burly", "burns", ... "turns"]
  - 17 words only ‘u’ as second letter and only ‘r’ third letter
- ["bucks", "bucky", ... "tufts"]
  - 98 words with only “u” second letter and no ‘r’

What should our secret word be? "ruddy" , "burch" or "bucks“?

Tell user there is no ‘r’
More Details on Game

- Current secret 8-letter word at random is `catalyst`
  - User guesses 'a', what should computer do?
  - Print `__ a __ a ___ ___` and continue?
More Details on Game

• Current secret 8-letter word at random is *catalyst*
  • User guesses 'a', what should computer do?
  • Print _ a _ a _ _ _ _ and continue?

No!
Try to change the word!
Best choice may be to tell the user there is no ‘a’
More Details on Game

• Current secret 8-letter word at random is *catalyst*
  • User guesses 'a', what should computer do?
  • Print \_ a \_ a \_ \_ \_ \_ \_ \_ and continue?

• Look at all groups of words and decide on a new word that is more likely to stump player
  • Why *designed* better choice than *tradeoff*?
  • 3,475 words with no 'a', 498 with 'a' 3\textsuperscript{rd} letter

Pick category with largest number of words!
Creating Groups/Categories

- For each of 7,070 words (8 letters), given word and ‘a’, find its group, represented by a template
- Use dictionary
  - Template is KEY, the VALUE is a list of matching words
- Choose biggest list
- Repeat
- # words smaller over time

<table>
<thead>
<tr>
<th>Group/Template</th>
<th>Size of Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>_ a _ _ _ _ _ _ _</td>
<td>587</td>
</tr>
<tr>
<td>_ a _ _ _ _ _ _ _ _</td>
<td>63</td>
</tr>
<tr>
<td>_ _ a _ _ _ _ _ _ _ _</td>
<td>498</td>
</tr>
<tr>
<td>_ _ _ a _ _ _ _ _ _ _ _ _</td>
<td>406</td>
</tr>
<tr>
<td>_ _ _ _ _ _ _ _ _ _ _ _ _</td>
<td>3,475</td>
</tr>
</tbody>
</table>
Changes to Regular GuessWord

• List of words from which secret word chosen
  • Initially this is all words of specified length
    • User will specify the length of the word to guess
  • After each guess, word list is a new subset
• Keep some functions, modify some, write new ones
• *Changes go in another function* to minimize changes to working program
  • Minimizing changes helps minimize introducing bugs into a working program
Play a game

• _______

• Secret word is:
  • flamer

• User guesses:
  • a

• Possible words:
  • 6166
Play a game

- __________
- Secret word is:
  - flamer
- User guesses:
  - a
- Possible words:
  - 6166

You build a dictionary for all the possible places an a can be in a word

Keys in dictionary
23 keys
Play a game

• _______
• Secret word is:  
  • flamer
• User guesses:  
  • a
• Possible words:  
  • 6166

Each value in dictionary is a list of words

These are the length of each value/list
Consider “__ __ a _ a” : 11

• Means “__ __ a _ a” is key in dictionary
• The value is a list of 11 words
  • have “a’ in 4th and 6th position

“__ __ a _ a”

['cicada', 'errata', 'guiana', 'guyana', 'ithaca', 'lusaka', 'nevada', 'ottawa', 'sonata', 'tirana', 'urbana']
Consider “__ __ a __ a” : 11

• Means “__ __ a __ a” is key in dictionary
• The value is a list of 11 words
  • have “a” in 4th and 6th position

“__ __ a __ a”

[key in dictionary]

['cicada', 'errata', 'guiana', 'guyana', 'ithaca', 'lusaka', 'nevada', 'ottawa', 'sonata', 'tirana', 'urbana']

[value in dictionary]
Play a game

- _______
- Secret word is: 
  - flamer
- User guesses:
  - a
- Possible words:
  - 6166
- Tell user: NO ‘a’

Pick new secret word, any letter without ‘a’
Play a game

- ________
- Secret word is:
  - mounds
- User guesses:
  - o
- Possible words:
  - 3441
Play a game

- _______
- Secret word is:
  - mounds
- User guesses:
  - o
- Possible words:
  - 3441

Note: None of these lists have the letter ‘a’ in them. We removed all words that have ‘a’ from our list of words.
Play a game

• __________
• Secret word is:
  • mounds
• User guesses:
  • o
• Possible words:
  • 3441
• Tell user no ‘o’

Pick new secret word, any letter without ‘o’
Play a game

• _ _ _ _ _ _
• Secret word is:
  • *burkes*
• User guesses:
  • u
• Possible words:
  • 2105
Play a game

- _______
- Secret word is: burkes
- User guesses: u
- Possible words: 2105
- Tell user no ‘u’

Largest category

<table>
<thead>
<tr>
<th></th>
<th>1441</th>
</tr>
</thead>
<tbody>
<tr>
<td>_______</td>
<td></td>
</tr>
<tr>
<td>______u</td>
<td>2</td>
</tr>
<tr>
<td>_______u</td>
<td>36</td>
</tr>
<tr>
<td>_______u</td>
<td>84</td>
</tr>
<tr>
<td>_______u_u</td>
<td>1</td>
</tr>
<tr>
<td>____<em><strong>u</strong></em></td>
<td>107</td>
</tr>
<tr>
<td>___<strong><strong>u</strong></strong></td>
<td>362</td>
</tr>
<tr>
<td>____<em><strong>u</strong></em></td>
<td>13</td>
</tr>
<tr>
<td>_____<strong>u</strong></td>
<td>11</td>
</tr>
<tr>
<td>_____<strong>uu</strong></td>
<td>37</td>
</tr>
<tr>
<td>______<em>u</em></td>
<td>5</td>
</tr>
<tr>
<td>______<em>u</em></td>
<td>5</td>
</tr>
<tr>
<td>______<em>u</em></td>
<td>5</td>
</tr>
<tr>
<td>_______u</td>
<td>1</td>
</tr>
</tbody>
</table>
Play a game

- ________

- Secret word is:
  - wilted

- User guesses:
  - i

- Possible words:
  - 1441
Play a game

- ______
- Secret word is: wilted
- User guesses: i
- Possible words: 1441
- Tell user no ‘i’

Largest category

Pick new secret word, any letter without ‘i’
Play a game

- ________
- Secret word is: served
- User guesses:
  - e
- Possible words:
  - 503
Play a game

- _______
- Secret word is: served
- User guesses: e
- Possible words: 503
- Tell user ‘e’ in these two places

Pick new secret word with ‘e’ in 2\textsuperscript{nd} and 5\textsuperscript{th} positions
Play a game

• _e__e_
• Secret word is:
  • tested
• User guesses:
  • s
• Possible words:
  • 160
Play a game

• _e__e_
• Secret word is: 
  • tested
• User guesses: 
  • s
• Possible words: 
  • 160
• Tell user no ‘s’

Largest category

Pick new secret word with no ‘s’ in it
Play a game

• _e__e_

• Secret word is:
  • kepler

• User guesses:
  • r

• Possible words:
  • 100
Play a game

•  _e___e_

• Secret word is:
  •  kepler

• User guesses:
  •  r

• Possible words:
  •  100

• Tell user no ‘r’
Play a game

• _e__e_

• Secret word is:
  • wedded

• User guesses:
  • d

• Possible words:
  • 45
Play a game

- _e__e_
- Secret word is:  
  - wedded
- User guesses:  
  - d
- Possible words:  
  - 45
- Tell user last letter is ‘d’

Largest category

Pick new secret word with ‘d’ as last letter
Play a game

• _ e _ e d
• Secret word is:
  • belted
• User guesses:
  • |  
• Possible words:
  • 20
Play a game

• _e__e d
• Secret word is:
  • belted
• User guesses:
  • l
• Possible words:
  • 20
  • Tell user no ‘l’

Pick new secret word with no ‘l’ in it

Largest category

_e__ed : 10
_el__ed : 4
_elled : 5
le__ed : 1
Play a game

• _e__e_d
• Secret word is:
  • vented
• User guesses:
  • t
• Possible words:
  • 4

_possible : 4
_possible : 1
_ettled : 4
_te_ted : 1
Play a game

• _e_ _e d
• Secret word is:
  • *vented*
• User guesses:
  • t
• Possible words:
  • 4
  • Tell user no ‘t’

That is 10 tries, Game Over!

It is really hard to win!

Largest category

<table>
<thead>
<tr>
<th>Word</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>e_ _ed</td>
<td>4</td>
</tr>
<tr>
<td>e_ ted</td>
<td>1</td>
</tr>
<tr>
<td>etted</td>
<td>4</td>
</tr>
<tr>
<td>te_ ted</td>
<td>1</td>
</tr>
</tbody>
</table>
Greedy Algorithms

• “Choosing largest group” –> greedy algorithm
  • Make a locally optimal decision that works in the long run
  • Choose largest group to make game last …

• Greed as in “it chooses the best current choice every time, which results in getting the best overall result”

• Canonical example? Change with coins
  • Minimize # coins given for change: 57 cents
Making change for 57 cents

• When choose next coin, always pick biggest
• With half-dollar coins

• With quarters and no half dollars
Making change for 57 cents

- When choose next coin, always pick biggest
- With half-dollar coins

- With quarters and no half dollars
When greedy doesn't work

• What if no nickels? Making change for 31 cents:
When greedy doesn't work

- What if no nickels? Making change for 31 cents:

- Can we do better? Yes!
Problem Solving

• Given Brodhead University. They have a basketball team.
• Data on players and how they did when playing against another team.

• List of lists named datalist
  • Each list has
    • school opponent name
    • player name
    • Points player scored
    • Whether game was ‘won’ or ‘lost’
Example: lists of 20 lists
data_list =

[ ['Duke', 'Bolton', '2', 'lost'],
  ['NCSU', 'Stone', '12', 'won'],
  ['Duke', 'Kreitz', '3', 'lost'],
  ['Duke', 'Pura', '6', 'lost'],
  ['GT', 'Dolgin', '4', 'lost'],
  ['WFU', 'Laveman', '20', 'won'],
  ['ECU', 'Parlin', '15', 'won'],
  ['UNC', 'Stone', '17', 'won'],
  ['UNC', 'Dolgin', '12', 'won'],
  ['UNC', 'Kreitz', '5', 'won'],
  ['Duke', 'Stone', '16', 'lost'],
  ['Duke', 'Laveman', '13', 'lost'],
  ['NCSU', 'Kreitz', '8', 'won'],
  ['NCSU', 'Dolgin', '18', 'won'],
  ['NCSU', 'Parlin', '13', 'won'],
  ['GT', 'Bolton', '7', 'lost'],
  ['GT', 'Stone', '9', 'lost'],
  ['WFU', 'Parlin', '14', 'won'],
  ['ECU', 'Laveman', '16', 'won'],
  ['ECU', 'Pura', '15', 'won'] ]
1) Write function `dictPlayerToNumGamesPlayedIn`

Build a dictionary of players mapped to number of games they have played in.

```python
def dictPlayerToNumGamesPlayedIn(datalist):
    # Function definition

    # With previous example, player 'Laveman' would be mapped to 3 games
```

With previous example, player ‘Laveman’ would be mapped to 3 games.
Woto-3 Players and Games Played in
Write function `dictPlayerToNumGamesPlayedIn`

def dictPlayerToNumGamesPlayedIn(datalist):
    d = {}
    for line in datalist:
        player = line[1]
        if player in d:
            d[player] += 1
        else:
            d[player] = 1
    return d

When each item needs its own count, build a dictionary

This is a counting dictionary
ANOTHER WAY: Write function `dictPlayerToNumGamesPlayedIn`

def dictPlayerToNumGamesPlayedIn(datalist):
    d = {}
    for line in datalist:
        player = line[1]
        if player not in d:
            d[player] = 0
            d[player] += 1
    return d
2) Write function
playersPlayedInNumGames(number, datalist)
Calculate list of players who played in 3 or more games, give (player name, number of games played in), sort by player name

[('Dolgin', 3), ('Kreitz', 3), ('Laveman', 3), ('Parlin', 3), ('Stone', 4)]

[ ['Duke', 'Bolton', '2', 'lost'],
  ['NCSU', 'Stone', '12', 'won'],
  ['Duke', 'Kreitz', '3', 'lost'],
  ['Duke', 'Pura', '6', 'lost'],
  ['GT', 'Dolgin', '4', 'lost'],
  ['WFU', 'Laveman', '20', 'won'],
  ['ECU', 'Parlin', '15', 'won'],
  ['UNC', 'Stone', '17', 'won'],
  ['UNC', 'Dolgin', '12', 'won'],
  ['UNC', 'Kreitz', '5', 'won'],
  ['Duke', 'Stone', '16', 'lost'],
  ['Duke', 'Laveman', '13', 'lost'],
  ['NCSU', 'Kreitz', '8', 'won'],
  ['NCSU', 'Dolgin', '18', 'won'],
  ['NCSU', 'Parlin', '13', 'won'],
  ['GT', 'Bolton', '7', 'lost'],
  ['GT', 'Stone', '9', 'lost'],
  ['WFU', 'Parlin', '14', 'won'],
  ['ECU', 'Laveman', '16', 'won'],
  ['ECU', 'Pura', '15', 'won'] ]
2) Write function playersPlayedInNumGames(number, datalist)

def playersPlayedInNumGames(number, datalist):
    d = dictPlayerToNumGamesPlayedIn(datalist)
    # build a list of tuples
    answer = [ ]
    for player in d.keys():
        if d[player] >= number:
            answer.append((player, d[player]))
    return sorted(answer)
2) Write function `playersPlayedInNumGames(number, datalist)`

```python
def playersPlayedInNumGames(number, datalist):
    d = dictPlayerToNumGamesPlayedIn(datalist)
    # build a list of tuples
    answer = []
    for (player, count) in d.items():
        if count >= number:
            answer.append( (player, count) )
    return sorted(answer)
```
ANOTHER WAY  2) Write function playersPlayedInNumGames(number, datalist)

Another way using a list comprehension!
However, this is putting a lot in one long line.
It may be better to break it up into steps as the previous two slides do. Less chance to make a mistake.

def playersPlayedInNumGames(number, datalist):
    d = dictPlayerToNumGamesPlayedIn(datalist)
    # build a list of tuples
    return sorted([ (player, count) for (player,count) in d.items() if count >= number] )
You should be able to:

• Build a dictionary
• Use a dictionary to help solve a problem