CS21: INTRODUCTION TO COMPUTER SCIENCE

Prof. Mathieson
Fall 2017
Swarthmore College
Outline Nov 13:

• Recap binary search and runtime: $O(\log(n))$
• Runtime worksheet (Handout 6)
• Begin: sorting

Notes

• Lab 8 due Saturday night (read BEFORE coming to lab!)
• Lab 9 due Monday after Thanksgiving
• Quiz 4 this Friday (let me know if you have conflicts)
Tips for Lab 8

• Read the entire lab before starting!
• Use string formatting to right-align strings or numbers

```python
lst = ["anya","christina","clarissa","michelle","pravadh","rachel","rye","scout","tristan"]

for name in lst:
    print("%20s" % name)

anya
cristina
clarissa
michelle
pravadh
rachel
rye
scout
tristan
```

• When reading files, so far we have used list accumulators, but you can also accumulate the data as one long string
• When in doubt, use an accumulator :)
Women and the LINC to Modern Computer Technology

A talk by Mary Allen Wilkes (The first person to use a personal computer in the home – and designer of the interactive operating system LAP6 for the LINC. Wikipedia)

Wednesday Nov. 15 at 5:00 p.m.    Science Center 101
Sponsored by Computer Science, WiCS, and Sigma Xi
All are welcome!
Binary Search
def binary_search(query, lst):
    low = 0
    high = len(lst) - 1
    count = 0

    # if low surpasses high (low > high), not found: STOP
    while low <= high:
        mid = (low+high)//2
        if query == lst[mid]:
            return mid, count  # return both the index and the count
        elif query < lst[mid]:
            high = mid - 1
        else:
            low = mid + 1
            count += 1

    return None, count  # return both the index and the count
Order of words in dictionary file: (/usr/share/dict/words)

Order on **Mac**: Not sorted! Need to convert to lower()

Order on **Linux**: Sorted! Don’t convert to lower()
Reading the dictionary file

- For Linux, don’t convert to lower! (already sorted)
- All upper case words come before all lower case words in Python as well

```python
def read_dictionary(filename):
    """Read a dictionary file and return a list of all the words."""
    word_file = open(filename, 'r')
    word_lst = []  # set up list accumulator

    # loop through each line (one word on each line)
    for line in word_file:
        word_lst.append(line.strip())

    word_file.close()
    return word_lst

>>> 'abba' < 'zip'
True
>>> 'abba' < 'Abba'
False
>>> 'Zip' < 'swarthmore'
True
```
Example of linear vs. binary

```python
def main():
    # get a list of all words in the dictionary
    dictionary = "/usr/share/dict/words"
    word_lst = read_dictionary(dictionary)

    n = int(input("Enter the number of words to use: "))
    subset = word_lst[:n]  # keep the first n words
    print("first word:", subset[0], "last word:", subset[-1])

    query = input("Enter a word to search for: ")

    print("\nlinear search:"
    index, count = linear_search(query, subset)
    print(query, "is at index", index, "with", count, "comparisons")

    print("\nbinary search:"
    index, count = binary_search(query, subset)
    print(query, "is at index", index, "with", count, "comparisons")

main()
```
# Linear vs. Binary: worst case

<table>
<thead>
<tr>
<th>n</th>
<th>Linear (# steps)</th>
<th>Binary (# steps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>100</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>1000</td>
<td>10</td>
</tr>
<tr>
<td>10000</td>
<td>10000</td>
<td>14</td>
</tr>
<tr>
<td>100000</td>
<td>100000</td>
<td>17</td>
</tr>
<tr>
<td>1000000</td>
<td>1000000</td>
<td>20</td>
</tr>
</tbody>
</table>

Below are the graphs illustrating the comparison:

- **Graph 1:** Shows a linear increase in steps for both linear and binary search methods as the input size increases.
- **Graph 2:** Illustrates a logarithmic increase in steps for binary search, with a linear trend for linear search, indicating the efficiency advantage of binary search for large datasets.
Handout 6 (Runtime)
1. for i in range(n):
   print(i)
   n steps $\rightarrow O(n)$

2. for i in range(100):
   print(i * n)
   100 steps
   Constant $\rightarrow O(1)$

3. for i in range(n):
   print(i)
   for j in range(n):
   steps: $2n$ $O(n)$
   print(j)

4. for i in range(n):
   print(i)
   for j in range(n):
   print(i * j)

5. for i in range(n):
   for j in range(i, n):
   print(i, j)

6. $O(n)$
7. $O(\log n)$
8. $O(1)$
9. $O(n \log n)$

\[ \frac{n^2}{2} \approx n^2 \]

\[ \frac{n^2}{2} \geq \frac{(n+1)(n+2)}{2} \]

\[ (n+1) \frac{n}{2} \geq \frac{O(n^2)}{} \]

\[ \sum_{i=1}^{n} i = \frac{n(n+1)}{2} \]

\[ \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \]

\[ \begin{array}{ccccccc}
0 & 1 & 2 & 3 & 4 & 5 \\
1 & x & x & x & x & x \\
2 & x & x & x & x & x \\
3 & x & x & x & x & x \\
4 & x & x & x & x & x \\
5 & x & x & x & x & x \\
\end{array} \]
Begin: Sorting
Sorting with cards

1) With a partner, set up a series of ~10 cards (out of order)

2) Try to come up with a **sorting algorithm** that only involves comparing and swapping elements

3) Check your algorithm with me or a ninja

4) Begin implementation in **sorts.py**

5) Here is our swap function from Week 6:

```python
def swap(i, j, lst):
    # This function swaps the i-th and j-th values of the lst.
    temp = lst[i]
    lst[i] = lst[j]
    lst[j] = temp
```