Algorithmic complexity: Speed of algorithms

Michael Ernst

CSE 140

University of Washington

How fast does your program run?

- Usually, this *does not matter*
- Correctness trumps speed

- Computer time is much cheaper than human time
- The cost of your program depends on:
  - Time to write and verify it
    - High cost: salaries
  - Time to run it
    - Low cost: electricity
- An inefficient program may give results faster

Sometimes, speed does matter

- Ridiculously inefficient algorithms
- Very large datasets

  Google:

  46 billion pages indexed (2011)

  3 billion searches per day (2012)

  = 150,000,000,000,000,000,000 pages searched per day

Example: Processing pairs

```
def make_pairs(list1, list2):
    """Return a list of pairs.
    Each pair is made of corresponding elements of list1 and list2.
    list1 and list2 must be of the same length."""
    …

assert make_pairs([100, 200], [101, 201]) == [[100, 101], [200, 201]]
```

- 2 nested loops vs. 1 loop
- Quadratic vs. linear time

Searching

```
def search(n, list):
    """Return index of value in list.
    The value must be in the list."""
    …
```

- Any list vs. a sorted list
- Linear vs. logarithmic time

Sorting

```
def sort(l):
    """Return a sorted version of the input list.
    The input list is not modified."""
    …

assert sort([3, 1, 4, 1, 5, 9, 2, 6, 5]) == [1, 1,
2, 3, 4, 5, 5, 6, 9]
```

- selection sort vs. quicksort
- 2 nested loops vs. recursive decomposition
- time: quadratic (n^2) vs. logarithmic (n log n)