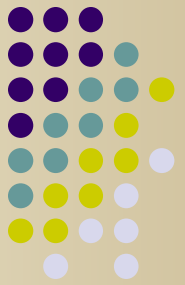
ENGR/CS 101 CS Session Lecture 11

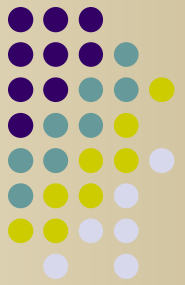


- No computers today
- Reminder: Homework 2 due on Wednesday. Submission system is now accepting assignments.

Outline

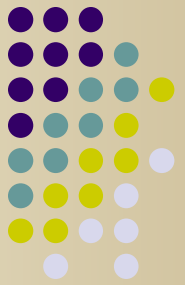
- Comparing algorithms
- Searching algorithms
- Sorting algorithms

Comparing Algorithms



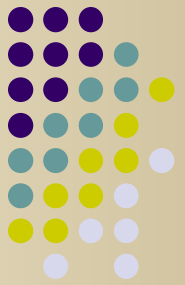
- Recall from first class: an ***algorithm*** is a series of well-defined steps that can be followed as a procedure.
- For most problems, there are several algorithms that may be used to solve the problem.
- We would like to know which ones are the better algorithms.

Comparing Algorithms



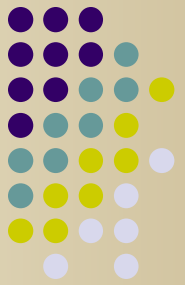
- Here is a simple example. Suppose we compute the sum of the integers between 1 and 100. I.e., sum = 1 + 2 + ... + 99 + 100 = 5050
- How many math operations are performed when we compute this sum?

Comparing Algorithms



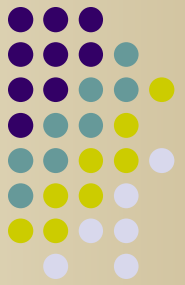
- Here is the way that the mathematician Gauss found to compute the same quantity: sum = (100 * (100+1))/2
- How many math operations are performed when we compute the sum this way?
- Aside: How do we know this is the right formula?

Comparing Algorithms



- We can generalize the example to computing the sum of the integers from 1 to **n** (an arbitrary integer greater than 1). I.e., sum = 1 + 2 + ... + n-1 + n
- How many math operations are performed for this algorithm?

Comparing Algorithms

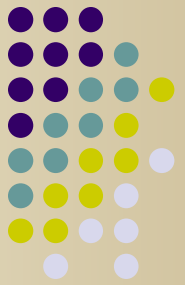


- The generalized formula is

  sum = (n * (n+1))/2

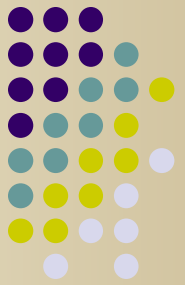
- How many math operations are performed to compute the sum with this formula?

Comparing Algorithms



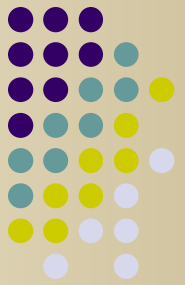
- If asked to compute the sum of the integers from 1 to **n**, which method would you choose?
- The first method depends on the size of **n**. The bigger **n** gets, the more math operations are needed. In particular, this method is directly proportional to the size of **n** and is said to be a ***linear*** time algorithm.

Comparing Algorithms



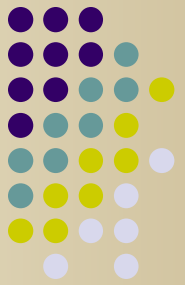
- The second method performs 3 math operations regardless of the size of **n**. Since the number of operations does not change, it is said to be a ***constant*** time algorithm.

Search Algorithms



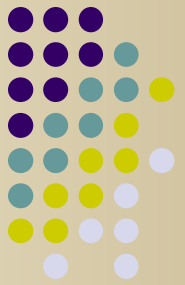
- Need 6 volunteers to stand at the front of the room.
- Each volunteer is given a card with a number not shown to the class.
- Need another student to find the person with the number 39 by asking to see a card using a systematic procedure. (I.e., choosing at random is **not** allowed.)
- What algorithm was used?

Search Algorithms



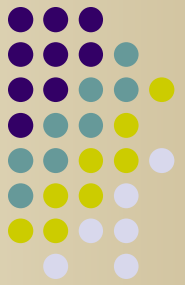
- Volunteers should exchange cards randomly, then stand in order from lowest to highest number, not showing their number to the class.
- Need another student to find the person with the number 39.
- What algorithm was used? Did we do better than last time?

Search Algorithms



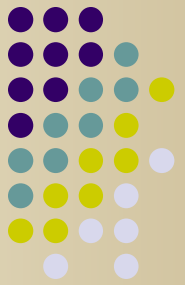
- In the first case, with the numbers in random order, the best algorithm is just to start at one end of the line asking to see a card until the target number is found. In the worst case, the target number is never found and we have asked every person in the line.
- Generalized to a line of **n** persons with **n** numbers, then the worst case is asking all **n** persons. This algorithm is called ***linear*** search.

Search Algorithms



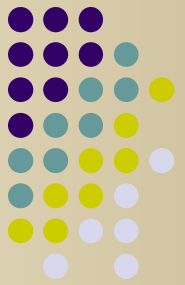
- In the second case, we know the numbers are in order, so the best algorithm is to start by asking the person in the middle to see her number. Then we eliminate half of the persons from having the number we seek. We can repeat this procedure with the half that might have the number, and so forth.
- This algorithm is called ***binary*** search. It is a ***logarithmic*** time algorithm, since it cuts down the search space in half at each step.

Sorting Algorithms



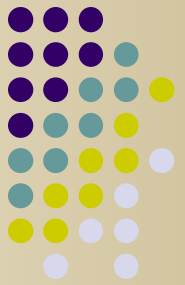
- Need another 6 volunteers to stand in the front of the room. (Can volunteer twice!)
- This time we want to rearrange the persons in order by height, shortest in front to tallest in back of the line. The catch is that the only "operations" are to compare the heights of two persons, and for two persons to trade places.

Sorting Algorithms



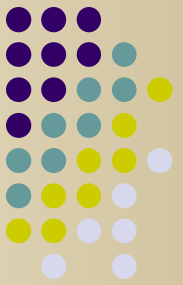
- Need a student to suggest an **algorithm** for getting the volunteers in height order. Again, choosing at random is not allowed. How many times are two persons' heights compared? How many times do two persons trade places?

Sorting Algorithms



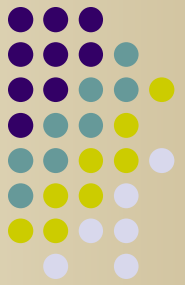
- Volunteers should go back to standing in random order.
- Need another student to suggest a ***different*** algorithm for getting the volunteers in height order. How many times are two persons' heights compared? How many times do two persons trade places?

Sorting Algorithms



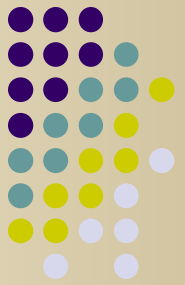
- There are myriads of sorting algorithms. Perhaps we developed one of the common ones.
- ***Selection Sort*** - find the shortest person and have her exchange places with the first person in the line; then find the next shortest person and have her exchange places with the second person in the line; repeat until everyone is in their correct place.

Sorting Algorithms



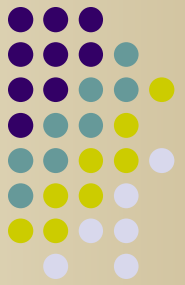
- ***Insertion Sort*** - start with the second person in line, if he is shorter than the person in front of him have them trade places. Repeat this comparison to the next person in front until everyone in front is shorter or the person has reached the front of the line. Repeat with the rest of the people in line.

Sorting Algorithms



- ***Bubble Sort*** - start with the first two persons in line, if they are not in the right order, exchange places, then move on to the second and third persons, and so forth to the end of the line. Repeat until the line is ordered.

Sorting Algorithms



- It turns out all of these algorithms are "slow" and have ***quadratic*** running time (proportional to $\mathbf{n^2}$). Faster sorting algorithms are more complex and are covered in CS 215.
- Analysis of more complex algorithms is covered in CS 315.