• **Homework 2**
  • Due Friday at 2pm
• **Midterm: Tuesday, March 8, 2016**
Given infinitely many coins with denominations in the set \{50, 25, 10, 5, 1\}, give an algorithm for making change for the amount \(A\) that uses the least number of coins.
Sort the coins in descending order, and use as many as possible of each coin before moving to the next coin. Now we prove this is optimal. Let $50 = c_1 > \cdots > c_5 = 1$ be the coin values. Assume we have created change for the amount $A$ only using coins strictly smaller than $c_i$, where $A \geq c_i$. If some of the coins used total to $c_i$, then we can replace them with $c_i$ and obtain a better result. Otherwise we have the following cases:

(a) $c_i = 50$ and a quarter, and 3 dimes are used: replace with a 50 cent piece and a nickel.

(b) $c_i = 25$ and 3 dimes are used: replace with a quarter and a nickel.
• Divide and conquer:
  • Break a problem into multiple smaller subproblems
  • Solve these subproblems
  • Combine the results
• Very popular:
  • Sorting (quicksort, mergesort)
  • Searching (binary search)
• Given a list of \( n \leq 30 \) signed integers and a 64-bit integer \( L \), determine if some sublist (not necessarily contiguous) sums up to the \( L \).
  
  • Split list in half, compute all possible sums that can occur in each half
  • \( O(2^{n/2} \log(2^{n/2})) = O(n^{2^{n/2}}) \)

• Multiply two big integers: \( a, b \) of length \( 2m \) bits
  
  • Note: \( a = a_h \times 2^m + a_l, b = b_h \times 2^m + b_l \)
  • Therefore \( ab = a_h b_h 2^{2m} + (a_h b_l + a_l b_h) 2^m + a_l b_l \)
  • Note that \( a_h b_l + a_l b_h = (a_h + a_l)(b_h + b_l) - a_h b_h - a_l b_l \)
  • Thus, \( ab \) can be computed recursively using 3 multiplications with operands half as big
  • Running time: \( O(n^{\log 3}) \), where \( \log 3 \approx 1.58 \)
  • Karatsuba’s algorithm
• Standard use: search for an element in a sorted array
  • Arrays.binarySearch, Collections.binarySearch in Java
  • lower_bound, upper_bound, binary_search in C++
• Recall that Java’s binarySearch doesn’t guarantee that the first occurrence of an element will be returned (when searching a list with duplicate elements)
• Still a popular interview problem!
• Useful application of binary search: remove one dimension of a complex problem
• **Example:** you are given a list of events that occur, and the time at which it occurs during your trip through the dessert
  • Spring a leak, causing you to lose gas at 1 unit per minute (multiple gas leaks accumulate)
  • Find a mechanic (fixes all leaks)
  • Find a gas station (fills tank)
  • Reach end of trip
• What is the minimum gas tank size required for the journey?
Let L be an array of integers.

**Exercise:** Compute the maximum value in L using divide and conquer.

**Exercise:** Assuming L is sorted in increasing order, implement binary search to find the first occurrence of k in L, and a second implementation to find the last occurrence.
Let L be an array of integers.

**Exercise**: Given k, compute the kth (0-based) smallest element of L.

**Exercise**: Assuming L has nonnegative entries, partition L into k contiguous subintervals so that the maximum sum over any of the subintervals is minimized.
**Exercise:** Suppose you have access to a function \( \text{double } f(\text{double } x) \) taking inputs from the interval \([0, 100]\). It is given that \( f \) (which may be discontinuous) is increasing on \([0, c]\) and decreasing on \([c, 100]\) for some unknown \( c \in [0, 100] \). Find \( c \) accurate to 6 digits after the decimal point.
Competitive Programming 3.3, 3.4