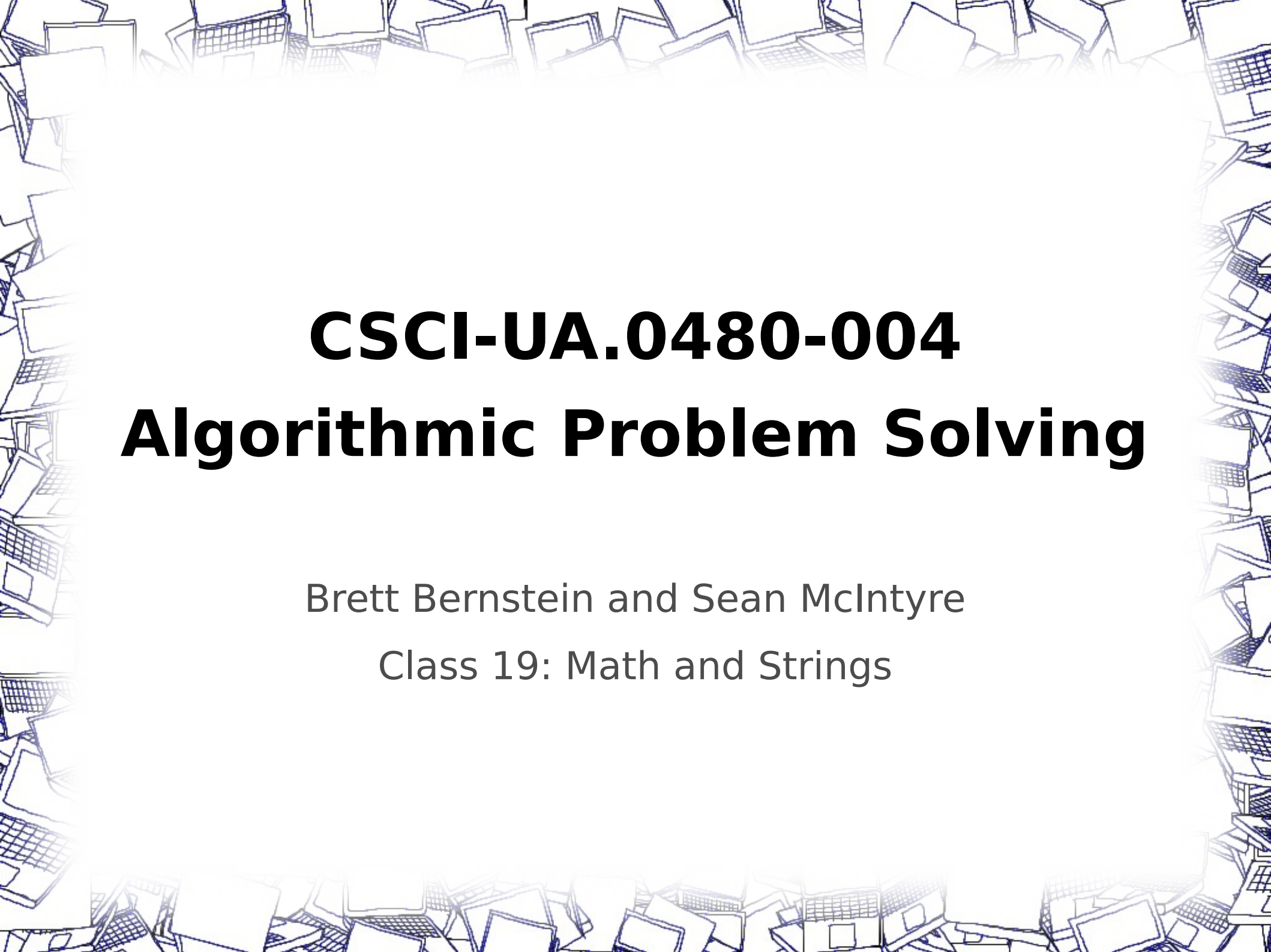


# CSCI-UA.0480-004
# Algorithmic Problem Solving

Brett Bernstein and Sean McIntyre

Class 19: Math and Strings

# Stacking Boxes

- Read Exercise 1

# Stacking Boxes

- Solution
  - Turns out there are only a relatively few possible configurations for the boxes
  - Determine all possible positions, then complete search on those positions
  - In particular, there can only be 1, 2, 3, or 4 stacks of boxes

# Stacking Boxes

- Solution, 1-stack
  - 1000 possible positions for a singular stack
  - For each position on the number line, compute the cost of moving all boxes to the position
  - Take the minimum cost for a 1-stack solution
  - Runtime: O($N$)

# Stacking Boxes

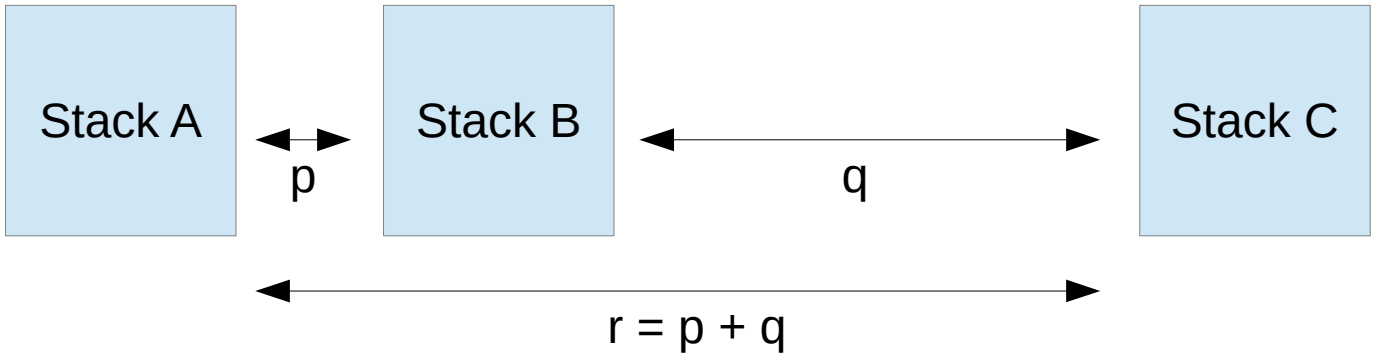
- Solution, 2-stacks
  - Use the Sieve of Eratosthenes to compute all primes up to 1000
  - For each start position $i$,
    - Compute the cost of moving all boxes to $i$ and $i+p$ for all possible primes $p$ (such that $i+p < 1000$)
    - Boxes less than $(2i+p)/2$ move to $i$; otherwise to $i+p$
    - Precompute the sum of boxes from 0 to position $x$, use to compute the number of boxes in an interval

# Stacking Boxes

- Solution, 2-stacks
  - Keep track of the minimum 2-stack solution
  - Runtime: O($N$ * $N$ / log($N$))

# Stacking Boxes

- Solution, 3-stacks
  - For all 3-stack solutions, one pair of boxes must be distance 2 from one another



# Stacking Boxes

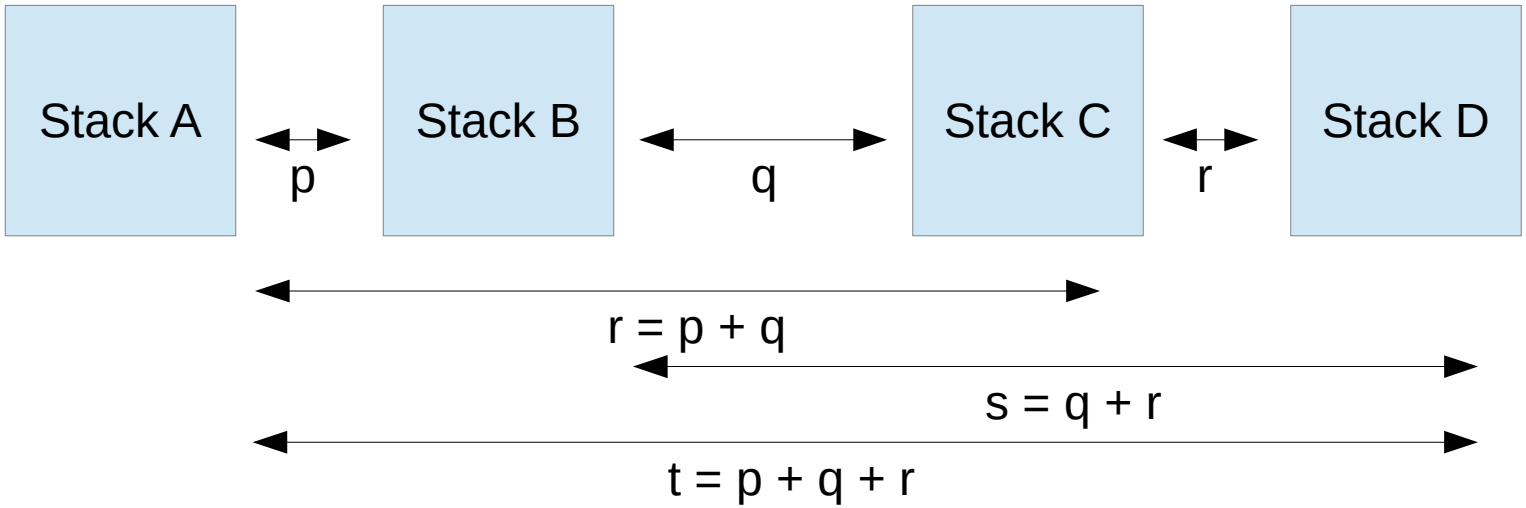
- Solution, 3-stacks
  - For all 3-stack solutions, one pair of boxes must be distance 2 from one another
  - Proof by contradiction:
    - Let $p$, $q$, $r$ be primes such that $p$ != 2, $q$ != 2, and $r = p+q$. Then $p$ and $q$ are both odd, so $2|(p+q)$ and (p+q) > 4. But then $r$ cannot be prime.
    - So either $p$ or $q$ must be 2.

# Stacking Boxes

- Solution, 3-stacks
  - So, for each start position *i*,
    - For each prime *p* such that *p*+2 is prime,
      - Compute the cost of moving boxes to stacks at *i*, *i*+*p*, *i*+*p*+2
      - Compute the cost of moving boxes to stacks at *i*, *i*+2, *i*+2+*p*
  - Keep track of the minimum 3-stack solution
  - Runtime: O(*N* * *N* / log(*N*))
    - But very few possibilities

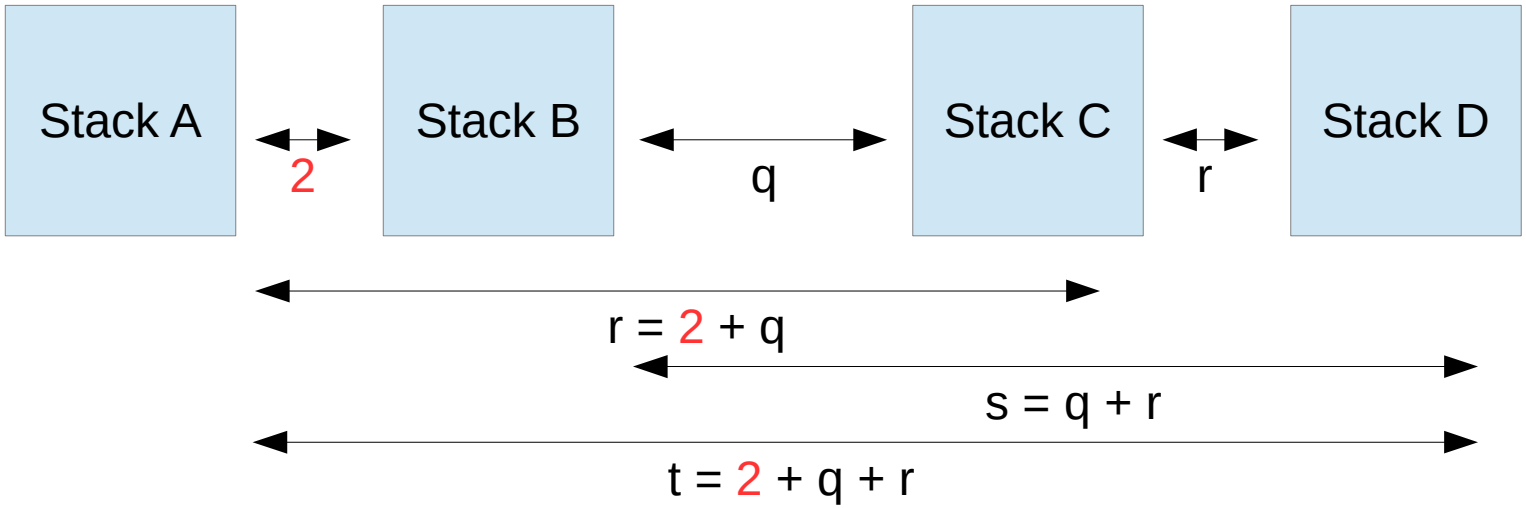
# Stacking Boxes

- Solution, 4-stacks
  - There is only one possible set of distances for 4-stack solutions



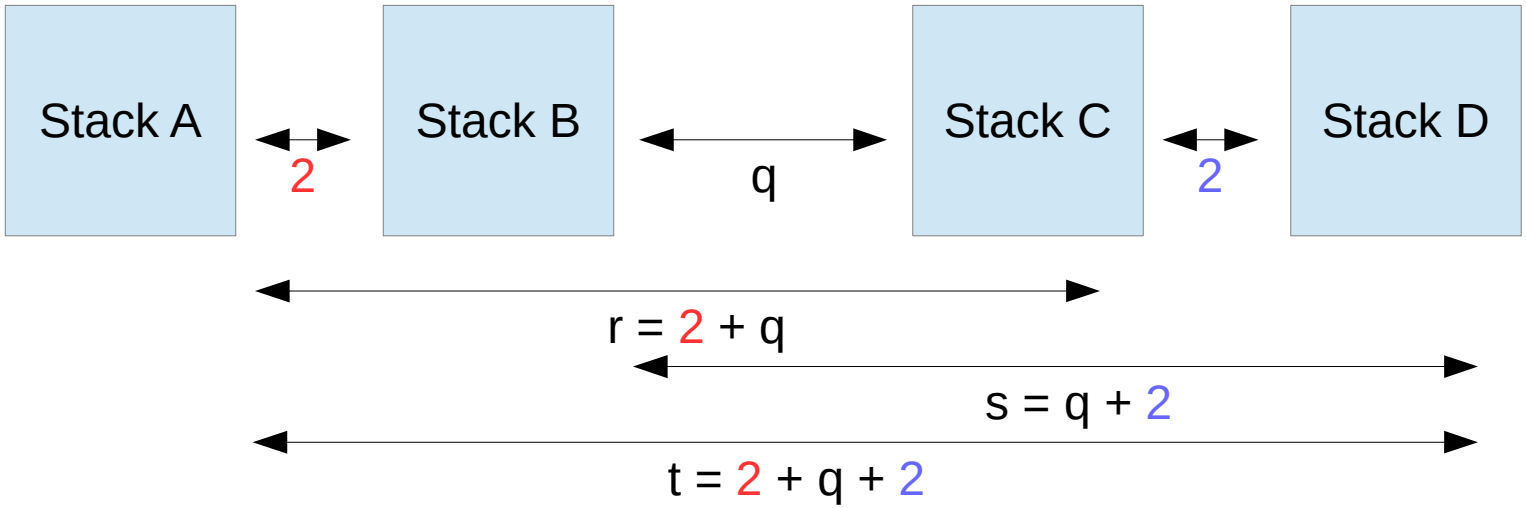
# Stacking Boxes

- Solution, 4-stacks
  - There is only one possible set of distances for 4-stack solutions



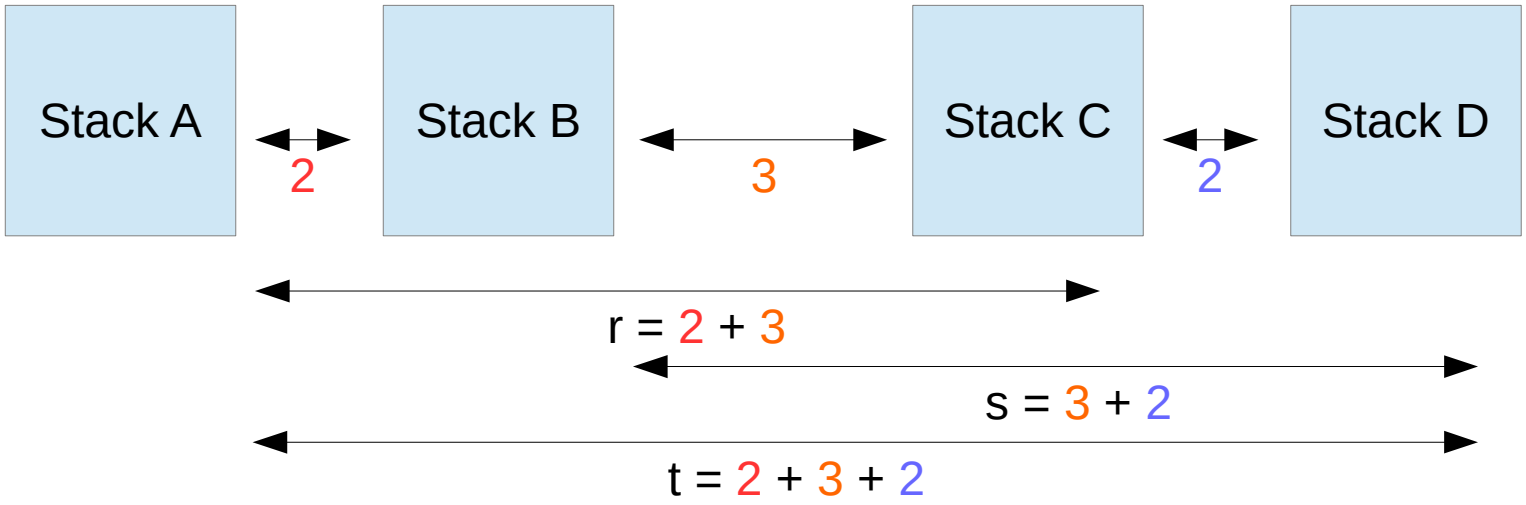
# Stacking Boxes

- Solution, 4-stacks
  - There is only one possible set of distances for 4-stack solutions



# Stacking Boxes

- Solution, 4-stacks
  - There is only one possible set of distances for 4-stack solutions



# Stacking Boxes

- Solution, 4-stacks
  - For each starting position *i,*
    - Compute the cost of moving boxes to *i*, *i*+2, *i*+5, and *i*+7
  - Keep track of the minimum 4-stack solution
  - Runtime: O(*N*)
  - Output the minimum of all solutions

# Cyberline

- Read the exercise

# Cyberline

```
public String lastCyberword(String cyberline) {
  String[] w =cyberline.replaceAll("-","")
        .replaceAll("[^a-zA-Z0-9]"," ")
        .split(" ") ;
  return w[w.length-1];
}
```

# Stammering Aliens

- Read the exercise

# Stammering Aliens

- Solution
  - Use a good hash function to hash the substrings and compare them
    - Too slow to store and pairwise compare strings
  - Hash function
    - hash(“babab”) = (2 * 31^4 + 1 * 31^3 + 1 * 31^2 + 1 * 31^1 + 1 * 3^0) % LARGE_PRIME = 1,877,826
    - 31 is a prime number greater than 26
      - Intuitively good for reducing collisions
    - LARGE_PRIME = 1,000,000,007

# Stammering Aliens

- Solution
  - Modulo arithmetic properties
    - (a+b) % p = ((a % p) + (b % p)) % p
    - (a – b) % p = ((a % p) – (b % p)) % p
    - (ab) % p = ((a % p) (b % p)) % p

# Stammering Aliens

- Solution
  - Sweep through the substring, computing all substring hashes of length L
    - Precompute 31^(L-1) % LARGE_PRIME using fast exponentiation
    - Remove the leftmost character from the hash by subtracting (char_value * 31^(L-1)) % LARGE_PRIME
    - Add the rightmost character to the hash by multiplying by 31 and adding char_value

# Stammering Aliens

- Solution
  - Maximum length of the string substring is strlen – nStrings
    - e.g., 4 cccccc
  - Try from max length down to 1
    - Too slow! :)
  - Binary search between 1 and max length
    - Turns out if you can find a solution of length L, you can find a solution of L-1 – this is a monotonic increasing function, so binary search