Administrivia:

HW0 due Wed (or Thu)

HW1 out today
due Tue
- groups of ≤ 3 — true collaborations
- collaborators + sources

Office Hours

Fast Fourier Transforms

↳ Convolutions

$A[0...n]$

$B[0...n]$

$A*B[0...2n]$ → $(A*B)[k] = \sum_{i+j=k} A[i] \cdot B[j]$

Given two sets $X = \{x_1 ... x_n\}$ $Y = \{y_1, ... y_n\}$ pos. ints.

$$X+Y = \{x+y \mid x \in X \text{ and } y \in Y\}$$

How big is $X+Y$? How many elements?

Easy: $O(n^2)$ expected time via hashing
$O(n^2 \log n)$ time via BBST

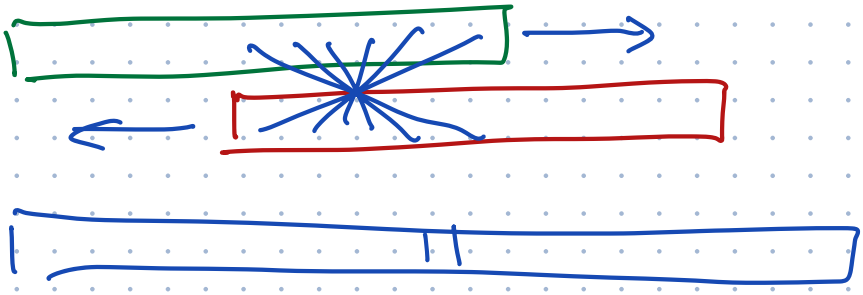
Think of x and Y as bit vectors: $X[i] = [i \in X]$

↑ Iverson bracket

$[P] = !!P$

$$(X+Y)[k] = [k \in X+Y]$$
$$= \bigvee_{i+j=k} (X[i] \wedge Y[j])$$
$$= [(X*Y)[k] > 0]$$

$O(U \log U)$ time where $U = \max(X \cup Y)$



PRIMVSDIGNITASINTAMTENVISCIENTIANONPOTEST
ESSERESENIMSVNTPARVAEPROPEINSINGVLISLITTERIS
ATQVEINTERPVNCTIONIBUSVERBORVMOCCVPATAE

sequence of letters ⟶ sequence of words
= nothing
= word + sequence of words

What is the first word?
↳ it's a word ⟶ Assume oracle IsWord(X[1..k])
↳ rest of string is splittable into words ⟶ recursion!

Try every prefix

"~~systematische~~ Tattivieren"

NOWHERE IS THIS CLEARER...

backtracking = recursive brute force

```
SPLITTABLE(A[1..n]):
  if n = 0
      return TRUE
  for i ← 1 to n
      if ISWORD(A[1..i])
          if SPLITTABLE(A[i+1..n])
              return TRUE
  return FALSE
```

$O(2^n)$ time in worst case

$$Splittable(i) = \begin{cases} \text{TRUE} & \text{if } i > n \\ \bigvee_{j=i}^{n} \big(\text{IsWord}(i,j) \wedge Splittable(j+1)\big) & \text{otherwise} \end{cases}$$

└ TRUE iff A[i..n] can be split into words.

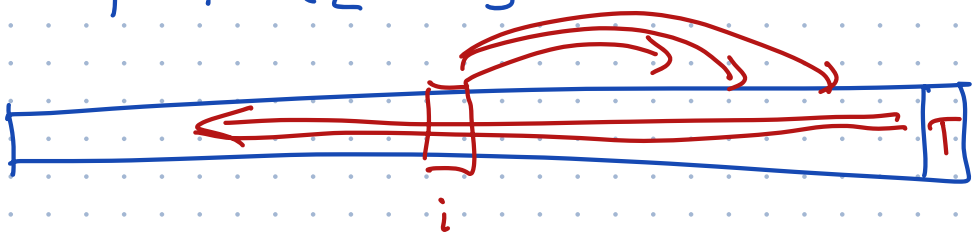
```
⟨⟨Is the suffix A[i..n] Splittable?⟩⟩
SPLITTABLE(i):
  if i > n
      return TRUE
  for j ← i to n
      if ISWORD(i, j)
          if SPLITTABLE(j + 1)
              return TRUE
  return FALSE
```

Memo(r)ize into data structure

Input: i

Output: Splitable(i) or ???

Array: SplitTable[1..n+1]



Memoized recurrence fills table from right to left

Easier/faster to do that ourselves

↓

```
FastSplittable(A[1..n]):
  SplitTable[n+1] ← True
  for i ← n down to 1
      SplitTable[i] ← False
      for j ← i to n
          if IsWord(i, j) and SplitTable[j+1]
              SplitTable[i] ← True
  return SplitTable[1]
```

Dynamic Programming

DP is <u>not</u> about nested for loops + table

but smart recursion