FEB 16 2016
abhi shelat
FFT, Median
merge-sort \((A, p, r)\)

if \(p < r\)

\[ q \leftarrow \lfloor (p + r)/2 \rfloor \]

merge-sort \((A, p, q)\)

merge-sort \((A, q + 1, r)\)

merge \((A, p, q, r)\)
Karatsuba(ab, cd)

Base case: return b*d if inputs are 1-digit

ac = Karatsuba(a,c)
bd = Karatsuba(b,d)
t = Karatsuba((a+b),(c+d))
mid = t - ac - bd

RETURN ac*1002 + mid*100 + bd

3T(n/2) + 2n

4n

3n
Closest(P, SX, SY)

Let q be the middle-element of SX
Divide P into Left, Right according to q
delta, r, j = MIN(Closest(Left, LX, LY) Closest(Right, RX, RY))

Mohawk = { Scan SY, add pts that are delta from q.x }
For each point x in Mohawk (in order):
    Compute distance to its next 15 neighbors
    Update delta, r, j if any pair (x, y) is < delta

Return (delta, r, j)
arbit+(A[1...n])

base case if |A|<=2, ...

(lg,minl,maxl) = arbit(left(A))
(rg,minr,maxr) = arbit(right(A))

return max{maxr-minl,lg,rg},
min{minl, minr},
max{maxl, maxr}
\[
R = \begin{bmatrix}
AE + BG & AF + BH \\
CE + DG & CF + DH
\end{bmatrix} = P_1 + P_2
\]

\[
T = P_3 + P_4 = P_5 + P_1 - P_3 - P_7
\]

\[
P_1 = A(F - H)
\]
\[
P_2 = (A + B)H
\]
\[
P_3 = (C + D)E
\]
\[
P_4 = D(G - E)
\]
\[
P_5 = (A + D)(E + H)
\]
\[
P_6 = (B - D)(G + H)
\]
\[
P_7 = (A - C)(E + F)
\]
FFT($f = a[1, ..., n]$)

Base case if $n \leq 2$

$E[...] ← \text{FFT}(A_e)$    // eval $A_e$ on $n/2$ roots of unity
$O[...] ← \text{FFT}(A_o)$    // eval $A_o$ on $n/2$ roots of unity

combine results using equation:

\[
A(ω_i, n) = A_e(ω_i^2, n) + ω_i, n A_o(ω_i^2, n)
\]

\[
A(ω_i, n) = A_e(ω_i \mod n/2, n/2) + ω_i, n A_o(ω_i \mod n/2, n/2)
\]

Return $n$ resulting values.
Fast Fourier Transform 2

© Jim Hatch Illustration / www.khulsey.com
FFT

**input:** $a_0, a_1, a_2, \ldots, a_{n-1}$

$$A(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$

**output:** evaluate polynomial $A$ at (any) $n$ different points.

$n$ points on a curve

$n$ roots of unity
$A(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$

Brute force method to evaluate $A$ at $n$ points:
\[ A(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} \]
\[ = a_0 + a_2 x^2 + a_4 x^4 + \cdots + a_{n-2} x^{n-2} + a_1 x + a_3 x^3 + a_5 x^5 + \cdots + a_{n-1} x^{n-1} \]

\[ A_e(x) = a_0 + a_2 x + a_4 x^2 + \cdots + a_n x^{(n-2)/2} \]
\[ A_o(x) = a_1 + a_3 x + a_5 x^2 + \cdots + a_{n-1} x^{(n-2)/2} \]

\[ A(x) = A_e(x^2) + \frac{1}{2} x A_o(x^2) \]

**Divide & Conquer**
\[ \text{FFT}(f=a[1,...,n]) \]

Evaluates degree n poly on the \( n \)th roots of unity

\[- E \leftarrow \text{FFT}(A_e) \quad \text{// } E[1...n/2] \]
\[- O \leftarrow \text{FFT}(A_0) \quad \text{// } O[1...n/2] \]

then compute

\[- A(x) = A_e(x^2) + xA_0(x^2) \quad \text{for } n \text{ poly} \]

\[ T(n) = 2T(\frac{n}{2}) + \Theta(n) \]
Last remaining issue: Which points to use?

Roots of unity should have $n$ solutions
what are they?

$x^n = 1$

Need points that have $\log(n)$ square roots

$\mathbb{Z}^*_p$
\[ x^n = 1 \]

the n solutions are:

\[ \{1, e^{2\pi i/n}, e^{2\pi i2/n}, e^{2\pi i3/n}, \ldots, e^{2\pi i(n-1)/n}\} \]

because \( e^{2\pi i} = 1 \) Euler identity

\[ \left[ e^{2\pi i \left( \frac{3}{4} \right)} \right]^A = (e^{2\pi i})^j = 1^j = 1 \]
\[ x^n = 1 \]

the n solutions are:

consider \( e^{2\pi ij/n} \) for \( j=0,1,2,3,\ldots,n-1 \)

\[
\left[ e^{(2\pi i/n)j} \right]^n = \left[ e^{(2\pi i/n)n} \right]^j = [e^{2\pi i}]^j = 1^j
\]

\( e^{2\pi ij/n} = \omega_{j,n} \) is an \( n \)th root of unity

\[ \omega_{0,n}, \omega_{2,n}, \ldots, \omega_{n-1,n} \]
What is this number?

\[ e^{2\pi ij/n} = \omega_{j,n} \] is an \( n \)th root of unity
Taylor series expansion

of a function $f$ around point $a$

$$f(y) = f(a) + \frac{f'(a)}{1!}(y - a) + \frac{f''(a)}{2!}(y - a)^2 + \frac{f'''(a)}{3!}(y - a)^2 + \cdots$$

$e^x = \text{around 0}$
What is this number?

\[ e^{2\pi ij/n} = \omega_{j,n} \] is an \( n \)th root of unity

\[ e^{ix} = \cos(x) + i \sin(x) \]

\[ e^{2\pi ij/n} = \cos(2\pi j/n) + i \sin(2\pi j/n) \]
$e^{2\pi i j/n} = \omega_{j,n}$ is an $n$th root of unity

$\omega_{0,n}, \omega_{2,n}, \ldots, \omega_{n-1,n}$

Let's compute $\omega_{1,8}$

\[
\omega_{1,8} = \cos \left( \frac{2\pi}{8} \right) + i \sin \left( \frac{2\pi}{8} \right)
\]
\[
= \cos \left( \frac{\pi}{4} \right) + i \sin \left( \frac{\pi}{4} \right)
\]
\[
= \frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}}
\]
Compute all 8 roots of unity \( n = 8 \)

\[ -\frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}} \]

Then graph them.
roots of unity

\[ x^n = 1 \]

should have n solutions

\[ e^{2\pi ij/n} = \cos\left(\frac{2\pi j}{n}\right) + i\sin\left(\frac{2\pi j}{n}\right) \]
squaring the $n^{\text{th}}$ roots of unity

$x^n = 1$

\[ \omega_{1,8} = (\frac{1}{\sqrt{2}} + \frac{i}{2})^2 = i \]

\[ \omega_{3,8} = \left( \frac{-1}{\sqrt{2}} + \frac{i}{2} \right)^2 = -i \]
Thm: Squaring an $n^{th}$ root produces an $n/2^{th}$ root.

example: \[\omega_{1,8} = \left(\frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}}\right)\]

\[\omega_{1,8}^2 = \left(\frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}}\right)^2 = \left(\frac{1}{\sqrt{2}}\right)^2 + 2\left(\frac{1}{\sqrt{2}} \cdot \frac{i}{\sqrt{2}}\right) + \left(\frac{i}{\sqrt{2}}\right)^2\]

\[= 1/2 + i - 1/2\]
\[= i\]
squaring the $n^{\text{th}}$ roots of unity

\( x^n = 1 \)

\( x^{n/2} = 1 \)
Thm: Squaring an $n^{\text{th}}$ root produces an $n/2^{\text{th}}$ root.

$$\{1, e^{2\pi i (1/n)}, e^{2\pi i (2/n)}, e^{2\pi i (3/n)}, \ldots, e^{2\pi i (n/2)/n}, e^{2\pi i (n/2+1)/n}, \ldots, e^{2\pi i (n-1)/n}\}$$

Square $n^{\text{th}}$ roots

\[
e^{\frac{2\pi i}{n}}, e^{\frac{2\pi i}{n/2}}, e^{\frac{2\pi i}{n/2}}, e^{\frac{2\pi i (n+2)/n}{n}}, e^{\frac{2\pi i (1/n)}{n/2}}
\]
Thm: Squaring an $n^{\text{th}}$ root produces an $n/2^{\text{th}}$ root.

\[ \left\{ 1, e^{2\pi i (1/n)}, e^{2\pi i (2/n)}, e^{2\pi i (3/n)}, \ldots, e^{2\pi i (n/2)/n}, e^{2\pi i (n/2+1)/n}, \ldots, e^{2\pi i (n-1)/n} \right\} \]

\[
\begin{pmatrix}
1 & e^{2\pi i (1/(n/2))} & e^{2\pi i (2/(n/2))} & e^{2\pi i (3/(n/2))} \\
& e^{2\pi i ((n/2)+1/(n/2))} \\
& & e^{2\pi i (1+1/(n/2))} \\
& & & = 1 \cdot e^{2\pi i (1/(n/2))}
\end{pmatrix}
\]
If $n=16$

- $A_{e, A_0}$
- $\omega_{1,8}$
- $\Omega_{e, \Omega}$
- $\Omega_{e, \Omega, 1,00}$

At the 16th roots of unity $A_{e, A_0}$

- $\Omega_{e, \Omega}$
- $\Omega_{e, \Omega, 1,00}$

- $2^n$ roots of unity

(Bad case)
\[ A(x) = A_e(x^2) + xA_o(x^2) \]
evaluate at a root of unity

\[ A(\omega_{i,n}) = A_e(\omega_{i,n}^2) + \omega_{i,n}A_o(\omega_{i,n}^2) \]
\[ \text{n}^{\text{th}} \text{ root of unity} \]
\[ \text{n}/2^{\text{th}} \text{ root of unity} \]
\[ \text{n}/2^{\text{th}} \text{ root of unity} \]
\[ \text{FFT}(f=a[1,...,n]) \]

Evaluates degree n poly on the \( n \)th roots of unity

\[ E \leftarrow \text{FFT}(A_e) \]  // eval \( A_e \) of degree \( \frac{n}{2} \) on the \( \frac{n}{2} \)th roots of unity

\[ 0 \leftarrow \text{FFT}(A_o) \]  // “

Combine these points to produce \( A \) eval @ \( n \)th roots

\( A(w_0,n), \ldots, A(w_{n-1},n) \) using the equation

\[ A(w_i,n) = A_e(w_i,n^2) + w_i^n \cdot A_o(w_i,n^2) \]
FFT($f=a[1,\ldots,n]$)

Base case if $n\leq 2$

$E[\ldots] \leftarrow \text{FFT}(A_e)$  // eval $A_e$ on $n/2$ roots of unity
$O[\ldots] \leftarrow \text{FFT}(A_o)$  // eval $A_o$ on $n/2$ roots of unity

combine results using equation:

$A(\omega_i,n) = A_e(\omega_i^2,n) + \omega_i,n A_o(\omega_i^2,n)$

$A(\omega_i,n) = A_e(\omega_i \mod n/2, \frac{n}{2}) + \omega_i,n A_o(\omega_i \mod n/2, \frac{n}{2})$

Return $n$ resulting values.
\[ 1 \cdot x^3 + 7 \cdot x^2 + 8 \cdot x + 9 \]

For \( x = 10 \)

\[
A(x) \cdot B(x) = C(x)
\]

And then

\[ \text{return } C(10) \]
\[ A(x) = a_3 x^3 + a_2 x^2 + a_1 x + a_0 \]
\[ B(x) = b_3 x^3 + b_2 x^2 + b_1 x + b_0 \]

\[ C(x) = a_3 b_3 x^6 + (a_3 b_2 + a_2 b_3) x^5 + (a_3 b_1 + a_2 b_2 + a_1 b_3) x^4 + (a_3 b_0 + a_2 b_1 + a_1 b_2 + a_0 b_3) x^3 + (a_2 b_0 + a_1 b_1 + a_0 b_2) x^2 + (a_1 b_0 + a_0 b_1) x + a_0 b_0 \]
\[ y = x + 1 \]
\[ y = 2x + 1 \]
$y = x + 1$

$y = 2x + 1$
\[ A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + 0x^4 + 0x^5 + 0x^6 + 0x^7 \]

\[ B(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + 0x^4 + 0x^5 + 0x^6 + 0x^7 \]
\[ A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + 0x^4 + 0x^5 + 0x^6 + 0x^7 \]

\[ B(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + 0x^4 + 0x^5 + 0x^6 + 0x^7 \]
\[ A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + 0 x^4 + 0 x^5 + 0 x^6 + 0 x^7 \]

\[ B(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + 0 x^4 + 0 x^5 + 0 x^6 + 0 x^7 \]

\[
\begin{align*}
A(\omega_0) & \quad A(\omega_1) & \quad A(\omega_2) & \quad \ldots & \quad A(\omega_7) \quad \text{FFT} \\
B(\omega_0) & \quad B(\omega_1) & \quad B(\omega_2) & \quad \ldots & \quad B(\omega_7) \quad \text{FFT} \\
C(\omega_0) & \quad C(\omega_1) & \quad \ldots & \quad \ldots & \quad C(\omega_8) \quad \text{multiply}
\end{align*}
\]
\[ A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + 0 x^4 + 0 x^5 + 0 x^6 + 0 x^7 \]

\[ B(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + 0 x^4 + 0 x^5 + 0 x^6 + 0 x^7 \]

\[
\begin{array}{cccccccc}
A(\omega_0) & A(\omega_1) & A(\omega_2) & \ldots & A(\omega_7) & \text{FFT} \\
B(\omega_0) & B(\omega_1) & B(\omega_2) & \ldots & B(\omega_7) & \text{FFT} \\
C(\omega_0) & C(\omega_1) & C(\omega_2) & \ldots & C(\omega_7) \\
\end{array}
\]
\[ A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + 0 x^4 + 0 x^5 + 0 x^6 + 0 x^7 \]

\[ B(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + 0 x^4 + 0 x^5 + 0 x^6 + 0 x^7 \]

\[ C(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_7 x^7 \]
application to mult

\[ \Theta(n^{\log_2 3}) \]
application to mult

\[ T(n) = 3T(n/2) + 6O(n) \]

\[ \Theta(n^{\log_2 3}) \]
Multiplying n-bit numbers

Schönhage–Strassen ‘71 \( O(n \log n \log \log n) \)

Fürer ‘07 \( O(n \log(n)2^{\log^*(n)}) \)
Abstract. Schönhage-Strassen’s algorithm is one of the best known algorithms for multiplying large integers. Implementing it efficiently is of utmost importance, since many other algorithms rely on it as a subroutine. We present here an improved implementation, based on the one distributed within the GMP library. The following ideas and techniques were used or tried: faster arithmetic modulo $2^n + 1$, improved cache locality, Mersenne transforms, Chinese Remainder Reconstruction, the $\sqrt{N}$ trick, Harley’s and Granlund’s tricks, improved tuning. We also discuss some ideas we plan to try in the future.

Introduction

Since Schönhage and Strassen have shown in 1971 how to multiply two $N$-bit integers in $O(N \log N \log \log N)$ time [21], several authors showed how to reduce other operations — inverse, division, square root, gcd, base conversion, elementary functions — to multiplication, possibly with log $N$ multiplicative factors [5, 8, 17, 18, 20, 23]. It has now become common practice to express complexities in terms of the cost $M(N)$ to multiply two $N$-bit numbers, and many researchers tried hard to get the best possible constants in front of $M(N)$ for the above-mentioned operations (see for example [6, 16]).

Strangely, much less effort was made for decreasing the implicit constant in $M(N)$ itself, although any gain on that constant will give a similar gain on all multiplication-based operations. Some authors reported on implementations of large integer arithmetic for specific hardware or as part of a number-theoretic project [2, 10]. In this article we concentrate on the question of an optimized implementation of Schönhage-Strassen’s algorithm on a classical workstation.
Applications of FFT
Conclusion

$O \left( n \log n \right)$

time

$O \left( \frac{n}{\log n} \right)$

data items
String matching with *

Looking for all occurrences of

\[ \text{GGC*GAG*C*GC} \]

where I don't care what the * symbol is.

\[ 0(4B \cdot m) \]

\[ 10^9, 10^6 \sim 10^{15} \]

\[ 10^9, \log (10^9) = 10^{10} \]
Median
Problem: given a list of \( n \) elements, find the element of rank \( n/2 \) (half are larger, half are smaller).
Problem: given a list of $n$ elements, find the element of rank $n/2$. (half are larger, half are smaller)

First solution: sort and pluck.

$O(n \log n)$
**Problem:** given a list of $n$ elements, find the element of rank $i$.

**Key Insight:**
- we do not have to “fully” sort.
- semi sort can suffice.
pick first element
partition list about this one
see where we stand
review: how to partition a list
review: how to partition a list

GOAL: start with THIS LIST and END with THAT LIST

less than greater than
review: how to partition a list
review: how to partition a list

is greater than p.
review: how to partition a list
review: how to partition a list
review: how to partition a list

\[ \Theta(n) \]

less than

greater than.
partitioning a list about an element takes linear time.
select \((i, A[1, \ldots, n])\)

Base case if \(|A| \leq 2\)

\(p \leftarrow \text{partition}(A)\) so that all elements are either \(\leq p\).

if \((i = p)\) return \(A[p]\).

else \((i < p)\) select \((i, A[0 \ldots p-1])\)

else select \((i-p-1, A[p \ldots n])\)
select \((i, A[1, \ldots, n])\)

handle base case.

partition list about first element

if pivot \(p\) is position \(i\), return pivot

else if pivot \(p\) is in position \(> i\)

else

select \((i, A[1, \ldots, p - 1])\)

else select \(((i - p - 1), A[p + 1, \ldots, n])\)

\[T(n) = \Theta(n) + T(\frac{n}{2}) + O(n) \implies \Theta(n)\]
Assume our partition always splits list into two equal parts.
handle base case.
partition list about first element
  if pivot is position $i$, return pivot
else if pivot is in position $> i$
else
  select ($i, A[1, \ldots, p - 1]$)
else
  select ($i - p - 1, A[p + 1, \ldots, n]$)
Assume our partition always splits list into two equal parts.

Handle base case.

Partition list about first element

If pivot is position $i$, return pivot.

Else if pivot is in position $> i$

Else select $(i, A[1, \ldots, p - 1])$

Else select $((i - p - 1), A[p + 1, \ldots, n])$

\[
T(n) = T(n/2) + O(n)
\]

\[
\Theta(n)
\]
problem: what if we always pick bad partitions?
problem: what if we always pick bad partitions?
problem: what if we always pick bad partitions?
problem: what if we always pick bad partitions?

\[ T(n) = T(n-5) + \Theta(n) = \Theta(n^2) \]

\[ \approx T(n-5) + \Theta(n) \]
select \( (i, A[1, \ldots, n]) \)

handle base case.
partition list about first element
if pivot is position \( i \), return pivot
else if pivot is in position \( > i \) select \( (i, A[1, \ldots, p - 1]) \)
else select \( ((i - p - 1), A[p + 1, \ldots, n]) \)
select \((i, A[1, \ldots, n])\)

handle base case.
partition list about first element
if pivot is position \(i\), return pivot
else if pivot is in position \(> i\) 
else 
select \((i, A[1, \ldots, p - 1])\)
else select \(((i - p - 1), A[p + 1, \ldots, n])\)

\[
T(n) = T(n - 1) + O(n)
\]

\[
\Theta(n^2)
\]
Needed:

a good partition element

partition \((A[1, \ldots, n])\)
Needed:

a good partition element

partition \((A[1, \ldots, n])\) produce an element where
30% smaller, 30% larger
solution: bootstrap

image: gucci
image: d&g
image: mark nason
partition \((A[1, \ldots, n])\)
partition \((A[1, \ldots, n])\)
partition \((A[1, \ldots, n])\)

\[ M = \]

compute the medians of each group

how big is this list \([\frac{n}{5}]\)

use \(\text{Select}(\frac{n}{10}, M)\) to pick \(p_r\)

our partition element.
partition \((A[1, \ldots, n])\)

median of each group

form a smaller list

\(B[1, \ldots, \lceil n/5 \rceil]\)

select \(\lfloor n/5 \rfloor /2, B[1, \ldots, \lfloor n/5 \rfloor]\)

use the median of this smaller list as the partition element
partition \((A[1, \ldots, n])\)
partition \((A[1, \ldots, n])\)

divide list into groups of 5 elements
find median of each small list
gather all medians
call select(...) on this sublist to find median
return the result
partition \((A[1, \ldots, n])\)

- Divide list into groups of 5 elements \(\Theta(n)\)
- Find median of each small list \(\Theta(n)\)
- Gather all medians \(\Theta(n/5)\)
- Call select(...) on this sublist to find median \(S\left(\frac{n}{5}\right)\)
- Return the result

\[
P(n) = S(\lceil n/5 \rceil) + O(n)
\]
a nice property of our partition
a nice property of our partition
a nice property of our partition
SWITCH TO A BIGGER EXAMPLE

\[ 3 \left( \frac{1}{2} \left\lceil \frac{n}{3} \right\rceil - 2 \right) \]

Every other column has 3 elements that are smaller than \( p \).

Might be able to find elements.

Our partition \( p \) is larger than all of these elements.
a nice property of our partition
a nice property of our partition

\[ 3 \left( \left\lceil \frac{1}{2} \left\lceil \frac{n}{5} \right\rceil \right\rceil - 2 \right) \]

\[ \geq \frac{3n}{10} - 6 \]

\[ \sim 30 \% \]

this implies there are at most \( \frac{7n}{10} + 6 \) numbers larger than \( \star \) /smaller
a nice property of our partition
\[ P_2 \leq \frac{7n}{10} + 6 \leq \frac{7n}{10} + 6 \]

This is the result of partition.
select \((i, A[1, \ldots, n])\)
select \( (i, A[1, \ldots, n]) \)

handle base case for small list
else pivot = FindPartitionValue(A,n) \( \longrightarrow P(n) = S\left(\frac{n}{5}\right) + \Theta(n) \)

partition list about pivot \( \longrightarrow \Theta(n) \)
if pivot is position i, return pivot
else if pivot is in position > i
else select \((i, A[1, \ldots, p-1])\) \( S\left(\frac{7n}{10} + b\right) \)
else select \((i-p-1), A[p+1, \ldots, n]\) \( S\left(\frac{n}{5} - 1\right) \)

\[ S(n) = S\left(\frac{n}{5}\right) + S\left(\frac{7n}{10} + b\right) + \Theta(n) = \Theta(n) \]
FindPartition \((A[1, \ldots, n])\)

- divide list into groups of 5 elements
- find median of each small list
- gather all medians
- call select(...) on this sublist to find median
- return the result

\[
P(n) = S(\lfloor n/5 \rfloor) + O(n)
\]
select \((i, A[1, \ldots, n])\)

handle base case for small list
else pivot = FindPartitionValue(A,n)
partition list about pivot
if pivot is position \(i\), return pivot
else if pivot is in position > \(i\)
else
select

\[ S(n) = S\left(\lceil n/5 \rceil \right) + O(n) + S\left(\frac{7n}{10} + 6 \right) \]
select \((i, A[1, \ldots, n])\)

- handle base case for small list
- else pivot = FindPartitionValue(A,n)
- partition list about pivot
- if pivot is position \(i\), return pivot
- else if pivot is in position \(> i\)
- else

\[
S(n) = S\left(\left\lfloor \frac{n}{5} \right\rfloor \right) + O(n) + S\left(\frac{7n}{10} + 6\right)
\]

\(\Theta(n)\)
Stairs(n)
    if n<=1 return 1
    return Stairs(n-1) + Stairs(n-2)
Stairs(n) if n<=1 return 1
    ret Stairs(n-1) + Stairs(n-2)

Stairs(5)

Stairs(4)  Stairs(3)

Stairs(3)  Stairs(2)  Stairs(2)  Stairs(1)

Stairs(2)  Stairs(1)  Stairs(1)  Stairs(0)  Stairs(1)  Stairs(0)
initialize memory M

Stairs(n)
    if n<=1 then return 1
    if n is in M, return M[n]
    answer = Stairs(i-1)+ Stairs(i-2)
    M[n] = answer
    return answer
Stairs(n)
    if n<=1 then return 1
    if n is in M, return M[n]
    answer = Stairs(i-1)+ Stairs(i-2)
    M[n] = answer
    return answer

Stairs(5)
Stairs(n)

    stair[0]=1
    stair[1]=1

    for i=2 to n
        stair[i] = stair[i-1]+stair[i-2]
    
    return stair[i]
initialize memory $M$

$\text{Stairs}(n)$
Stairs(n)

  if n<=1 then return 1
  if n is in M, return M[n]
  answer = Stairs(i-1)+ Stairs(i-2)
  M[n] = answer
  return answer
Stairs(n)

    stair[0]=1
    stair[1]=1

    for i=2 to n
        stair[i] = stair[i-1]+stair[i-2]
    
    return stair[i]
Dynamic Programming
two big ideas
two big ideas
recursive structure
+ memoizing
wood cutting

Quarter Sawn Log

Regular Sawn Log

http://www.amishhandcraftedheirlooms.com/quarter-sawn-oak.htm
Spot price for lumber
Spot price for lumber

1”  2”  3”  4”  5”  6”  7”  8”
Log cutter dilemma

input to the problem: \( n, (p_1, \ldots, p_n) \)

goal:
Observation
Solution equation
Approach

\[0 \quad \ldots \quad i\]
BestLogs \( n, (p_1, \ldots, p_n) \)

\[
\text{if } n \leq 0 \text{ return } 0
\]
BestLogs\left(n,(p_1,\ldots,p_n)\right)

\begin{align*}
\text{if } n & \leq 0 \text{ return } 0 \\
\text{for } i=1 \text{ to } n \\
\text{    } \text{Best}[i] &= \max_{k=1}^{i} \{ p_k + \text{Best}[i-k] \}
\end{align*}
The actual cuts?
BestLogs($n, (p_1, \ldots, p_n))$

if $n \leq 0$ return $0$

for $i = 1$ to $n$

Best[$i$] = max$_{k=1 \ldots i}$ \{ $p_k + \text{Best}[i - k]$ \}