Using Introduction to the Theory of Computation, Chapter 2
Outline

- gaussian multiplication
- binary search
intuition about master theorem

\[
T(n) = \begin{cases} 
  c & n = 1 \\
  a_2 T(\lfloor n/b \rfloor) + a_1 T(\lfloor n/b \rfloor) + f(n) & n > 1 
\end{cases}
\]

Cost recursive calls:
\[
a_1 + a_2 = q
\]

Cost splitting + recombining:
\[
(b_k)^l \rightarrow \ell (b_k)^l
\]
\[
\sum_{i=0}^{k-1} a_i (b_k^{-i})^l
\]
\[
c \cdot a^k = a \log_b n \cdot c
\]
\[
a \log_b n \cdot c
\]
intuition about master theorem

\[ T(n) = \begin{cases} 
  c & \text{if } n = 1 \\
  a_2 T(\lfloor n/b \rfloor) + a_1 T(\lfloor n/b \rfloor) + f(n) & \text{if } n > 1 
\end{cases} \]

\[ \sum_{i=0}^{k-1} \left( \frac{a}{b^l} \right)^i \cdot (f(k))^l = n^l \sum_{i=0}^{k-1} \left( \frac{a}{b^l} \right)^i \]

\[ a = b^l \Rightarrow n^l \cdot \frac{k}{b^l} = n^l \log_b n \]

If \( a > b^l \), then...
Gauss’s trick

\[ xy = 2^n x_1 y_1 + x_0 y_0 + 2^{n/2} ((x_1 + x_0)(y_1 + y_0) - x_1 y_1 - x_0 y_0) \]

Master theorem

- \( a = \begin{cases} 3^2 \end{cases} \)
- \( b = 2 \)
- \( l = \begin{cases} 1 \end{cases} \)

**Apply Master Theorem**

- \( a \geq b^l \rightarrow \text{yes} \)
- \( 3 \times 2 \rightarrow \text{defers} \)

Complexity \( \Theta(n \log^3 n) \) should be better than \( O(n \log^2 n) \)
Gauss’s payoff
lose one multiplication

1. divide each factor (roughly) in half
2. sum the halves
3. multiply the sum and the halves Gauss-wise
4. combine the products with shifts and adds
recursive binary search

def recBinSearch(x, A, b, e):
    if b == e:
        if x <= A[b]:
            return b
        else:
            return e + 1
    else:
        m = (b + e) // 2 # midpoint
        if x <= A[m]:
            return recBinSearch(x, A, b, m)
        else:
            return recBinSearch(x, A, m+1, e)

What correct?
conditions, pre- and post-

pre conditions

- $x$ and elements of $A$ are comparable
- $e$ and $b$ are valid indices, $b \leq e$
- $A[b..e]$ is sorted non-decreasing

RecBinSearch($x, A, b, e$) terminates and returns index $p$

- $b \leq p \leq e + 1$
- $b < p \Rightarrow A[p - 1] < x$
- $p \leq e \Rightarrow x \leq A[p]$

(except for boundaries, returns $p$ so that $A[p - 1] < x \leq A[p]$)
precondition $\Rightarrow$ termination and postcondition

Proof: induction on $n = e - b + 1$

Base case, $n = 1$: Terminates because there are no loops or further calls, returns $x \leq A[b = p] \iff p = b = e$ is returned. $x > A[b = p - 1] \iff p = b + 1$ returned, so postcondition satisfied. Notice that the choice forces if-and-only-if.

Induction step: Assume $n > 1$ and that the postcondition is satisfied for inputs of size $1 \leq k < n$ that satisfy the precondition. Call RecBinSearch($A,x,b,e$) when $n = e - b + 1 > 1$. Since $b < e$ in this case, the test on line 1 fails, and line 7 executes. Exercise: $b \leq m < e$ in this case. There are two cases, according to whether $x \leq A[m]$ or $x > A[m]$.
Case 1: \( x \leq A[m] \)

must show \( 1 \leq k < n = e+1 - b \)

\( IH: P(k), 1 \leq k < n = e+1 - b \)

\( m \geq b \)
gives this

\( p > m \)
gives this

- Show that \( IH \) applies to \( RBS(x,A,b,m) \)
- Translate the postcondition to \( RBS(x,A,b,m) \)
  - \( RBS \) returns \( p \) such that
    1. \( b \leq p \leq m+1 \)
    2. \( b < p \Rightarrow A[p-1] < x \)
    3. \( p \leq m \Rightarrow x \leq A[p] \)
- Show that \( RBS(x,A,b,e) \) satisfies postcondition
  0. \( RBS \) returns \( p \) such that
  1. \( b \leq p \leq e \), since \( IH \) says \( b \leq p \leq m+1 < e+1 \),
  2. \( b < p \Rightarrow A[p-1] < x \), \( \checkmark \) directly, hence \( x \leq e \)
  3. \( p \leq e \), \( x \leq A[p] \)
     we know \( p \leq e \), must show \( x \leq A[p] \)
     \( \downarrow \)
     \( A[p] \)
    \( \downarrow \) case \( p \leq m \), then \( x \leq A[p] \) by \( IH \)
    \( \downarrow \) case \( p = m+1 \), by Case \( x \leq A[m] \)
    \( \checkmark \) list sorted, we know \( x \leq A[m] \leq A[m+1] \)
Case 2: $x > A[m]$

Show that IH applies to $\text{RBS}(x,A,m+1,e)$

Translate postcondition to $\text{RBS}(x,A,m+1,e)$

RBS returns $p$ such that

- $m+1 \leq p \leq e+1$
- $m+1 < p \implies A[p-1] < x$
- $p \leq e \implies x \leq A[p]$

Show that $\text{RBS}(x,A,b,e)$

by IH, RBS certainly returns some $p$ and

- $b \leq b + 1 \leq m' \leq p \leq e + 1$, by IH
- $b < p$ (always true, since $p \geq m+1 \geq b+1 > b$
- $m+1 < p$, by IH, $A[p-1] < x$
Case 2: $x > A[m]$

- Show that IH applies to $RBS(x, A, m+1, e)$
- Translate postcondition to $RBS(x, A, m+1, e)$

- Show that $RBS(x, A, b, e)$

  - $\ldots \text{ otherwise } p = m+1$, so $A[p-1] = A[m] < x$ (by Case 2)
  - $p \leq e \Rightarrow A[p] \geq x$

(by IH)
Case 2: $x > A[m]$

Conclude, by complete induction, that $RBS(x, A, b, e)$ with preconditions satisfy postconditions.

- Show that IH applies to $RBS(x, A, m+1, e)$
- Translate postcondition to $RBS(x, A, m+1, e)$
- Show that $RBS(x, A, b, e)$
what could go wrong?

- $m = \left\lfloor \frac{e+b}{2.0} \right\rfloor$
- $x < A[m]$

Either prove correct, or find a counter-example

- $b \leq m < e$
- $m = e$
  
  - $b = 0$
  
  - $e = 1$
  
  - $\left\lfloor \frac{1}{2} \right\rfloor = 1$

- $A = \begin{bmatrix} 2 & 3 \\ 0 & 1 \end{bmatrix}$

  - $\text{RBS}(2,A,0,1)$
  
  - wrong index?
recursive and iterative
mergesort

MergeSort(A,b,e):
1. if b == e: return
2. m = (b + e) / 2  # integer division
3. MergeSort(A,b,m)
4. MergeSort(A,m+1,e)
   # merge sorted A[b..m] and A[m+1..e] back into A[b..e]
5. for i = b,...,e: B[i] = A[i]
6. c = b
7. d = m+1
8. for i = b,...,e:
   9. if d > e or (c <= m and B[c] < B[d]):
      10. A[i] = B[c]
      11. c = c + 1
   else: # d <= e and (c > m or B[c] >= B[d])
      13. d = d + 1
conditions, pre- and post-

- $b$ and $e$ are natural numbers, $0 \leq b \leq e < \text{len}(A)$.
- elements of $A$ are comparable

- $A'[b..e]$ contains the same elements as $A[b..e]$, but sorted in non-decreasing order (use notation $A'$ for $A$ after calling MergeSort($A,b,e$)). All other elements of $A'$ are unchanged.
Proof of correctness of MergeSort(A,b,e) by induction on $n = e - b + 1$ for all arrays of size $n$, 
(precondition+execution)$\Rightarrow$(termination+postcondition)

Base case, $1 = e - b + 1$: Assume MergeSort(A,b,e) is called with $\text{len}(A) = 1$ preconditions satisfied. Then $0 \leq e \leq b \leq 0$, so $e == b$, and the algorithm terminates with a (trivially) sorted $A'$, satisfying the precondition.

Induction step: Assume $n \in \mathbb{N}$, $n > 1$, and for all natural numbers $k$, $1 \leq k < n$, that MergeSort on all arrays of size $k$ that satisfy the precondition and run will terminate and satisfy the postcondition. Assume MergSort(A,b,e) is executed and $n = e - b + 1$. 
The test on line 1 fails, and $m$ is set to $(b + e)//2$, strictly less than $e$ (exercise).

Does the IH apply to MergeSort($A, b, m$) and MergeSort($A, m+1, e$)? Translate the IH into postconditions for MergeSort($A, b, m$) and MergeSort($A, m+1, e$).

Now we need iterative correctness for the merge...
iterative correctness
partial correctness plus termination

- Preconditions plus termination imply the postcondition. Probably needs a loop invariant

- termination — construct a decreasing sequence in $\mathbb{N}$.