Fibonacci Sequence

$f_0 = 0, \quad f_1 = 1$

$f_n = f_{n-1} + f_{n-2}, \quad n \geq 2$

Ex. Show that if $n \geq 3$, $f_n > \alpha^{n-2}$ where $\alpha = \frac{1+\sqrt{5}}{2}$

proof.

Base step: $\alpha < 2 = f_3$; $\alpha^2 = \frac{3+\sqrt{5}}{2} < 3 = f_4$

So $P(3)$ and $P(4)$ are true

Inductive step: Assume $P(j)$ is true, namely, $f_j > \alpha^{j-2}$ for $3 \leq j \leq k$, $k \geq 4$. We must show $P(k+1)$ is true. that is, that $f_{k+1} > \alpha^{k-1}$,

we have $f_{k+1} = f_k + f_{k-1} > \alpha^{k-2} + \alpha^{k-3}$

Note that $\alpha^2 = \alpha + 1$. This gives

$\alpha^{k-2} + \alpha^{k-3} = (\alpha+1)\alpha^{k-3} = \alpha^2 \alpha^{k-3} = \alpha^{k-1}$

therefore, $f_{k+1} > \alpha^{k-1}$, so $P(k+1)$ is True

By strong induction, the statement is true.

Binet's formula

$$f_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right), \quad n \geq 0$$

Cassini's Identity

$f_n^2 - f_{n+1} f_{n-1} = (-1)^{n-1}$ for all $n \geq 1$

proof. Basis step $f_1^2 - f_2 f_0 = 1 - 1 \cdot 0 = 1 = (-1)^{1-1}$

Inductive step. Assume that $f_k^2 - f_{k+1} f_{k-1} = (-1)^{k-1}$ for some $k \geq 1$

Then $f_{k+1}^2 - f_{k+2} f_k = f_{k+1}^2 - (f_{k+1} + f_k) f_k$

$= f_{k+1}^2 - f_{k+1} f_k - f_k^2$

$= f_{k+1}(f_k + f_{k-1}) - f_{k+1} f_k - f_k^2$

$= f_{k+1} f_k + f_{k+1} f_{k-1} - f_{k+1} f_k - f_k^2$

$= -(f_k^2 - f_{k+1} f_{k-1}) = (-1)^k$

By induction, the statement is true for all $n \geq 1$.

Climbing Stairs Problem.

- You are at the bottom of a staircase of n steps
- You can only climb 1 step or 2 steps at a time
- The goal is to find the total number of unique ways to reach the top of the staircase.

$t_1 = 1$, $t_2 = 2$ (1+1 or 2)

$t_n = t_{n-1} + t_{n-2}$, $n \geq 3$

we find that $t_n = f_{n+1}$ (satisfies the same recurrence)

Merge Sort

procedure mergesort ($L = a_1, \ldots, a_n$)

if $n > 1$ then

$m := \lfloor \frac{n}{2} \rfloor$

$L_1 := a_1, \ldots, a_m$

$L_2 := a_{m+1}, \ldots, a_n$

$L =$ merge(mergesort(L_1), mergesort(L_2))

Procedure merge (L_1, L_2 ; sorted lists)

$L :=$ empty list

while L_1 and L_2 are both nonempty

remove smaller of first elements of L_1 and L_2

put it at the right end of L

if one list is empty then

append all elements of another to the right end of L.

Running time : $O(n \log n)$ where n is the size of input array

$$T(n) = 2T(\tfrac{n}{2}) + n$$

$$\vdots$$

$$= 2^k T(\tfrac{n}{2^k}) + kn$$

$$= O(n \log n)$$

$2^k \geq n > 2^{k-1}$

$k = O(\log n)$

$2^k = O(n)$, $T(\frac{n}{2^k}) = \Theta(1)$