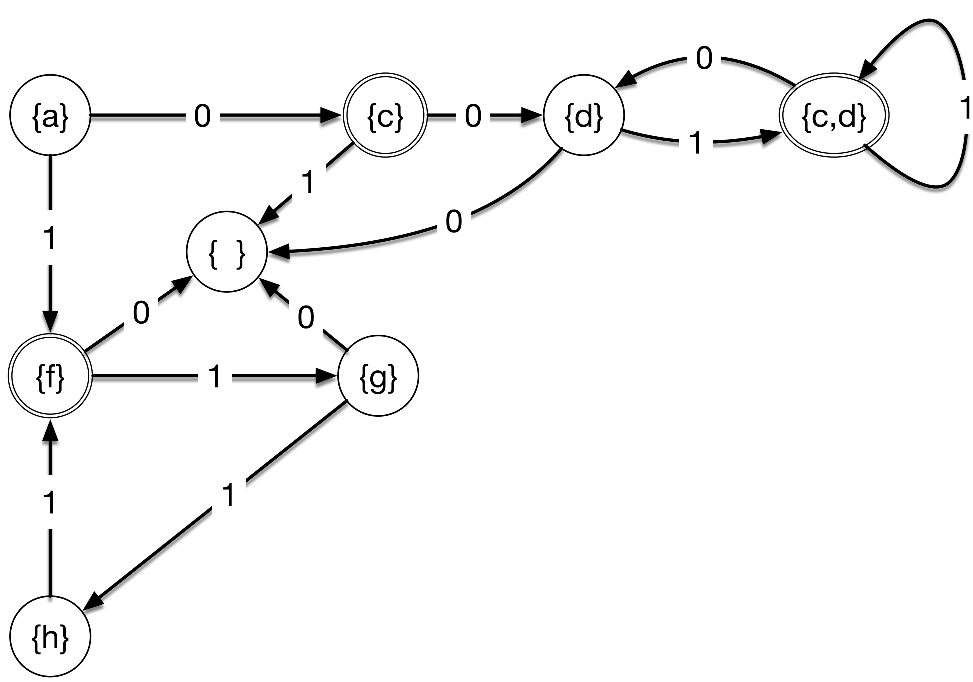
Homework 1 Solutions

Problem 1:

The easiest way is to just start with the starting state and find out the list of states which can be reached on a 0 or 1 symbol. For each line, the resulting set of states is then treated the same. This gives the following table

State	0	1
{a}	{c}	{f}
{c}	{d}	{ }
{d}	{ }	{c,d}
{c,d}	{d}	{c,d}
{f}	{ }	{g}
{g}	{ }	{h}
{h}	{ }	{f}

The accepting states are {c}, {c,d}, and {f}. We can draw a automaton:



The alternative is to take the ε-closure of all states, which means that we start with $\{a, b, e\}$.

Problem 2:

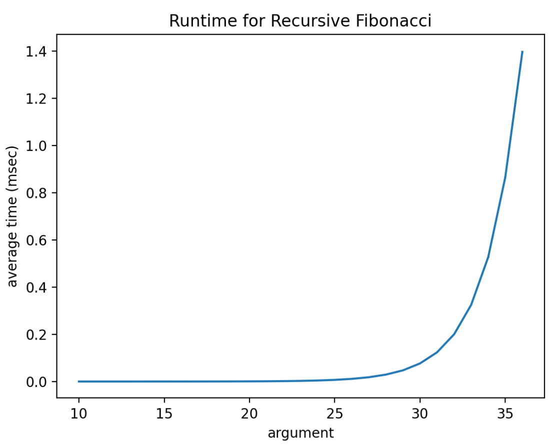
We can solve this in a single program, using confidence intervals of 5σ, using plt.errorbar. However, in order to actually see errorbars, we need to restrict ourselves to small arguments (below right).

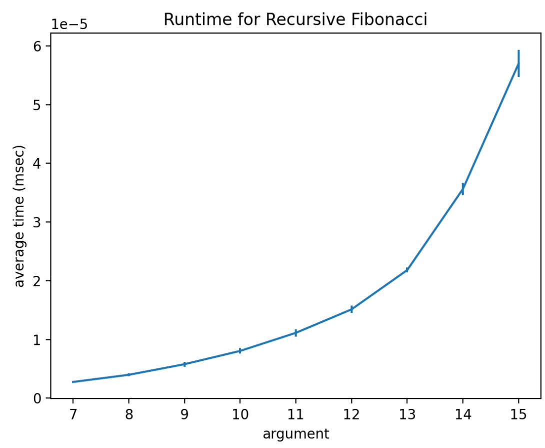
```
import time
import numpy as np
import matplotlib.pyplot as plt

def rec_fib(n):
    if n <= 1:
        return n
    else:
        return rec_fib(n-1)+rec_fib(n-2)


def timer(n):
    start = time.time()
    for _ in range(50):
        accu = rec_fib(n)
    end = time.time()
    return (end-start)/50

def main(n,m):
    xx, mus, sigmas = [], [], []
    for i in range(n,m+1):
        xx.append(i)
        counts = [timer(i) for _ in range(30)]
        mus.append(np.mean(counts))
        sigmas.append(5*np.std(counts)/np.sqrt(30))
    plt.errorbar(x=xx, y=mus, yerr=sigmas)
    plt.xlabel('argument')
    plt.ylabel('average time (msec)')
    plt.title('Runtime for Recursive Fibonacci')
    plt.show()
```





The recurrence relation is $r_n = r_{n-1} + r_{n-2} + 2$, as there are two recursive call, the first of which will create r_{n-1} recursive calls and the second one will create r_{n-2} recursive calls. For $n = 2$, $r_2 = 2 = f_1 + f_2 + f_3 - 2$. Assume that $r_n = f_{n-1} + f_n + f_{n+1} - 2$ is true for all $n \leq m$. Then

$$\begin{aligned} r_{m+1} &= 2 + r_m + r_{m-1} \\ &= 2 - 2 + f_{m-1} + f_m + f_{m+1} - 2 + f_{m-2} + f_{m-1} + f_m \\ &= -2 + f_m + f_{m+1} + f_{m+2} \end{aligned}$$

where in the last step we used the Fibonacci recursive formula.