Anuj Kumar - 214297998

# EECS 3101 Assignment 2

## Problem 1: Slot machine design and analysis

(a) In the worst case **500** "cha-ching" are printed by the slot machine. Because the worst case from the casino's point of view is when A=B=C=1. Then the loop runs 500 times.

(b) When m=1, then the worst case happens all the time. Hence the probability is 1.

When m=2, the probability that A=1 is ½. Similarly the probability that B=1 is ½ and C=1 is ½. Hence the probability that A=B=C=1 is (½)^3 = 1/8.

So in general, the probability of worst case happening = (1/m)^3.

(c) In the best case**, zero** "cha-ching" is printed. The best case is when A is not equal to B.

(d) The probability that best case occurs is calculated as follows. Both A & B can take m slots each. So A & B can take a total of $m^2$combinations of values. Out of which, there are m ways in which A=B.
Hence the probability that A is not equal to B is given by, $1 - \frac{m}{m^2} = \frac{m-1}{m}$

(e) Let's calculate the expected money the gambler earns. Note that 2 dollar is paid to play.

Case 1: Jackpot! A=B=C=1

Money earned = 500 – 2 = $498.
Probability for this to happen = $1/m^3$ as shown in (b)

Case 2: A=B=C but not A=B=C=1

Money earned = 20 – 2 = $18
Probability that A=B=C = $\frac{m}{m^3} = 1/m^2$
But we have to exclude the first case (A=B=C=1).
Hence the probability of this case happening = $\frac{1}{m^2} - \frac{1}{m^3} = \frac{m-1}{m^3}$

Case 3: A=B but not A=B=C

Money earned = 5 – 2 = $3
Probability that A=B = $\frac{1}{m}$
But we have to exclude the case that A=B=C

Hence the probability of this case = $\frac{1}{m} - \frac{1}{m^2} = \frac{m-1}{m^2}$

Case 4: None of the above (A!=B)

Money earned = -2$

Probability of this case = $\frac{m-1}{m}$ as done shown in (c)

Expected money earned by the gambler = $498 * \frac{1}{m^3} + 18 * \frac{m-1}{m^3} + 3 * \frac{(m-1)}{m^2} - 2 * \frac{m-1}{m}$

$= \frac{498+18m-18+3m^2-3m-2m^3+2m^2}{m^3} = \frac{480+15m+5m^2-2m^3}{m^3}$

Hence the expected money earned by the casino is the negative of the above quantity,

$= \frac{2m^3-5m^2-15m-480}{m^3}$

(f) Casino's owner expects to earn $0.8 to $1 per play.

We below list the expected money earned by the casino for different values of m:

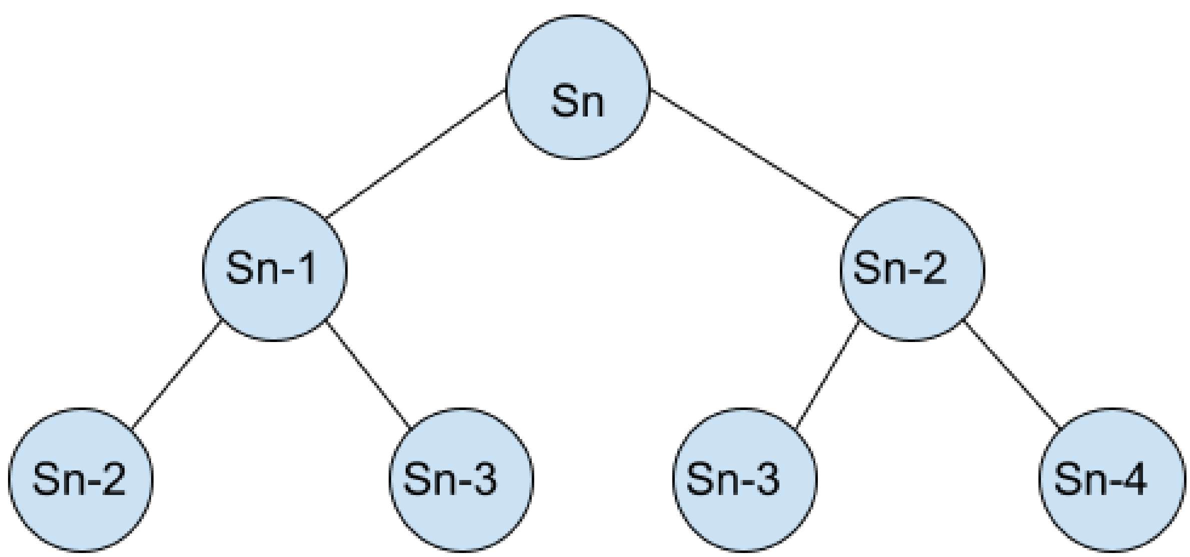
| m | Expected earnings ($) |
|---|---|
| 1 | -498.000 |
| 2 | -64.250 |
| 3 | -19.111 |
| 4 | -7.688 |
| 5 | -3.440 |
| 6 | -1.472 |
| 7 | -0.420 |
| 8 | 0.203 |
| 9 | 0.601 |
| 10 | 0.870 |

| m | Expected earnings ($) |
|---|---|
| 11 | 1.061 |
| 12 | 1.201 |

It is clear from the above table that **m = 10** for the expected earning of the casino is between $0.8 and $1 per play.

## Problem 2: Computing numbers in a sequence

(a) The recursion tree is shown below;




Since $S_n = 2S_{n-1} + 3S_{n-2}$, To calculate Sn, we have to calculate Sn-1 and Sn-2. This is shown as the two children of the root node. The recursive tree at depth k has $2^k$nodes. The height of the tree is at least n/2. So the time complexity is $\Omega(2^n)$.

(b) Consider the matrix multiplication,

$$\begin{bmatrix} 2 & 3 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2x + 3y \\ x \end{bmatrix}$$

$$\begin{bmatrix} 2 & 3 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} S_{n-1} \\ S_{n-2} \end{bmatrix} = \begin{bmatrix} 2S_{n-1} + 3S_{n-2} \\ S_{n-1} \end{bmatrix} = \begin{bmatrix} S_n \\ S_{n-1} \end{bmatrix}$$

(By using the fact that $2S_{n-1} + 3S_{n-2} = S_n$)

We have $S_1 = 1$ & $S_2 = 2$. So $S_3$ can be obtained by matrix multiplication as follows.

$\begin{bmatrix} 2 & 3 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} S_2 \\ S_1 \end{bmatrix} = \begin{bmatrix} S_3 \\ S_2 \end{bmatrix}$ - Equation 1

Then $S_4$ can be obtained by matrix multiplication as
$\begin{bmatrix} 2 & 3 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} S_3 \\ S_2 \end{bmatrix} = \begin{bmatrix} S_4 \\ S_3 \end{bmatrix}$ – Equation 2

But using Equation 1 to substitute for $\begin{bmatrix} S_3 \\ S_2 \end{bmatrix}$ in Equation 2,

We obtain $\begin{bmatrix} 2 & 3 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 2 & 3 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} S_2 \\ S_1 \end{bmatrix} = \begin{bmatrix} S_4 \\ S_3 \end{bmatrix}$ (Let's call M=$\begin{bmatrix} 2 & 3 \\ 1 & 0 \end{bmatrix}$ and note that $\begin{bmatrix} S_2 \\ S_1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ )

So recursively, we can get $S_n$ using matrix multiplication as,

$$\begin{bmatrix} 2 & 3 \\ 1 & 0 \end{bmatrix}^{n-2} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} S_n \\ S_{n-1} \end{bmatrix}$$

So to find Sn, we just have to repeatedly multiply the matrix M, n-2 times. This can be done in $O(logn)$time.

The following is a pseudo code of the algorithm to find the $n^{th}$power of a matrix A.

Function matrix_power(A, n):

1. If n == 1 then return A
2. If n is odd, then
3.     k = floor(n/2)
4.     temp = matrix_power(A, k)
5.     return (temp * temp * A)

```
if n is even, then
    k = n/2
    temp = matrix_power(A, k)
    return (temp*temp)
```

So in order to find Sn, we call matrix power(M, n-2) and then multiply the result with $\begin{bmatrix} 2 \\ 1 \end{bmatrix}$ and return the first component of the 2 dimensional vector output.

(c) Note that the algorithm matrix power to find the nth power of a matrix is based on the divide and conquer approach, because we subdivide the problem into finding the solutions to smaller problems. The recursive relation can be written as

$$T(n) = T(n/2) + O(1)$$

This is because to find the matrix power of n, we find the matrix power of n/2 recursively, and the additional operations cost only a constant number of operations (since the dimension of the matrix is 2x2).

So by the Master theorem, the overall time complexity is $T(n) = \Theta(\log n)$