## CS 5633: Analysis of Algorithms

## Homework 3

**1. 4.5->1)**

1. T(n) = 2T(n/4)+1

For this recurrence, we have a = 2, b = 4, f(n) = 1, and thus we have that nlogba = nlog42 = n1/2

Since f(n) = O(nlogba -ε), where ε = 0.5, we can apply case 1 of the master theorem and conclude that the solution is T(n) = θ(n1/2)

1. T(n) = 2T(n/4)+n1/2

For this recurrence, we have a = 2, b = 4, f(n) = n1/2, and thus we have that nlogba = nlog42 = n1/2

Since f(n) = θ(nlogba ), we can apply case 2 of the master theorem and conclude that the solution is T(n) = θ(n1/2logn)

1. T(n) = 2T(n/4)+n

For this recurrence, we have a = 2, b = 4, f(n) = n, and thus we have that nlogba = nlog42 = n1/2

Since f(n) = Ω(nlogba +ε), where ε = 0.5, we can apply case 3 of the master theorem if we can show that the regularity condition holds for f(n).

For sufficiently large n, we have that af(n/b) = 2.n/4 = n/2 <= cf(n) for ½ <= c < 1. Consequently, by case 3 T(n) = θ(n)

1. T(n) = 2T(n/4)+n2

For this recurrence, we have a = 2, b = 4, f(n) = n2

Thus nlogba = nlog42 = n1/2

Since f(n) = Ω(nlogba +ε), where ε = 1.5, we can apply case 3 of the master theorem if we can show that the regularity condition holds for f(n). For sufficiently large n, we have that af(n/b)=2.n2/16 = n2/8 <= cf(n) for ⅛ <= c < 1. Consequently, by case 3 T(n) = θ(n2)

**4.5->4)** T(n) = 4T(n/2)+n2logn

For this recurrence, we have a = 4, b = 2, f(n) = n2logn, and thus we have that nlogba = nlog24 = n2

Though f(n) = n2logn is asymptotically larger than nlogba = n2, it is not polynomially larger.The ratio f(n)/nlogba  = logn is asymptotically less than nε for any positive constant ε. Consequently, the recurrence falls into the gap between case 2 and case 3. So, the master theorem doesn’t apply here.

The asymptotic upper bound = O(n2logn) as n2logn is larger than the subproblem.

**2.** As I have to divide a problem of size n into several subproblems of size n/3 and the dividing and combining will take O(log n) time,

T(n) = a.T(n/3) + logn

Comparing with T(n) = a.T(n/b) + f(n) we get b = 3 and f(n) = logn

To ensure T(n) = o(n2),

nlogba  = nlog3a

For a >= 3, nlog3a will be greater than f(n)

In that case, T(n) = θ(nlog3a) where a >= 3

T(n) to be o(n2),

nlog3a < n2

=> log3a < 2

=> a < 9

So, a can be a maximum of 8. So, I can divide the problem into 8 subproblems maximum.

**3. a)** Here, S = {1, 2, 3, 4, 5, 6}, set of all possible outcomes of a fair, 6-face die.

If X is the outcome of a single roll of a fair, 6-face die then P(X) = 1/6 for all values of S

Expected value, E[X] = 1.⅙ + 2.⅙ + 3.⅙ + 4.⅙ + 5.⅙ + 6.⅙

= 21.⅙

= 3.5

**b)** Here, S = {1, 2, 3, 4, 5, 6}, set of all possible outcomes of a fair, 6-face die.

If X is the outcome of a single roll of a fair, 6-face die then P(X) = 1/6 for all values of S

Expected value, E[X] = ∑i = 16 i. E[Xi]

Now, E[Xi] is ⅙ for one dice. For the sum of n dice, E[Xi] = n/6

So, E[X] = 21\*n/6 = 3.5n

**4.** Win $10 if 11

Win $1 if \_1

Lose $0.5 otherwise

Let Y be a Random Variable that denotes the gain.

Y(11) = 10, Y(one 1) = 1, Y(others) = -0.5

P(Y = 11) = 1/36, P(one 1) = 10/36, P(others) = 1 - (1/36+10/36) = 25/36

Expected value, E[X] = 10\*1/36 + 1\*10/36 - 0.5\*25/36 = 15/(2\*36) = 5/24