**Chapter 19**

***R-19.2*** Suppose two teams, the Anteaters and the Bears, have a long rivalry in basketball. Suppose further that in any given game, the Anteaters will beat the Bears with Probability 2/3, independent of any other games that they play. Give a bound on the probability that, in spite of this, the Bears will win a majority of n games that they play.

Answer:

From the given problem there are n games. The probability that Anteaters win a game is 2/3 and the probability that Bears win a game is 1/3.

Let X: where bears win a game

X ~ B (n, 1/3)

Bears will win a majority of n games means it should win more than n/2 games. Then, the probability that bears will win majority of n games is

P (x > (n/2)) = 1 – P (x ≤ (n/2)) = 1 – F(n/2)

Let’s find the probability bound where bears win majority of games.

And for Bears to win majority of the games, they need to win 50% of the matches.

Using the Chernoff’s bound:

Let be the random independent trials for Games being chosen.

Hence,

i.e.

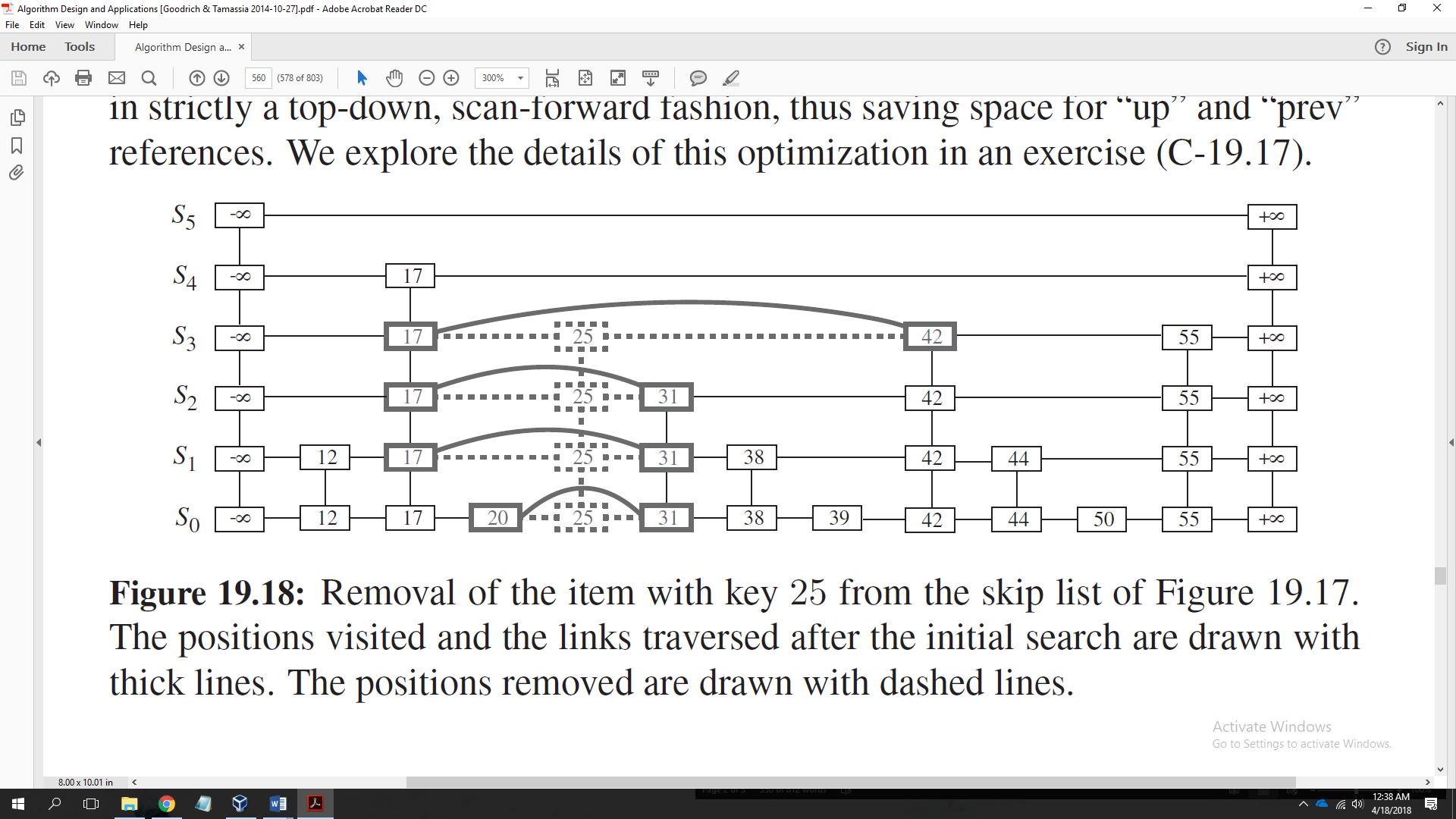
is the mean probability of each independent probability of event

To find the upper bound that no. of games bears Win=0.5n is

Thus, the upper bound on probability for using Chernoff Bound is < 0.9n

***R-19.13*** Draw an example skip list resulting from performing the following sequence of operations on the skip list in Figure 19.18: remove (38), insert (8, x), insert (24, y), remove (55). Assume the coin flips for the first insertion yield two heads followed by tails, and those for the second insertion yield three heads followed by tails.

Answer:



Following is the path highlighted for removing 38

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S­5** | **- ∞** |  |  |  |  |  |  |  |  |  |  | **+ ∞** |
| **S4** | **- ∞** |  | **17** |  |  |  |  | **42** |  |  | **55** | **+ ∞** |
| **S3** | **- ∞** |  | **17** |  |  |  |  | **42** |  |  | **55** | **+ ∞** |
| **S2** | **- ∞** |  | **17** |  | **31** |  |  | **42** |  |  | **55** | **+ ∞** |
| **S1** | **- ∞** | **12** | **17** |  | **31** | **38** |  | **42** | **44** |  | **55** | **+ ∞** |
| **S0** | **- ∞** | **12** | **17** | **20** | **31** | **38** | **39** | **42** | **44** | **50** | **55** | **+ ∞** |

Insert 8

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S5** | **- ∞** |  |  |  |  |  |  |  |  |  |  | **+ ∞** |
| **S4** | **- ∞** |  |  | **17** |  |  |  | **42** |  |  | **55** | **+ ∞** |
| **S3** | **- ∞** |  |  | **17** |  |  |  | **42** |  |  | **55** | **+ ∞** |
| **S2** | **- ∞** | **8** |  | **17** |  | **31** |  | **42** |  |  | **55** | **+ ∞** |
| **S1** | **- ∞** | **8** | **12** | **17** |  | **31** |  | **42** | **44** |  | **55** | **+ ∞** |
| **S0** | **- ∞** | **8** | **12** | **17** | **20** | **31** | **39** | **42** | **44** | **50** | **55** | **+ ∞** |

Insert 24

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S5** | **- ∞** |  |  |  |  |  |  |  |  |  |  |  | **+ ∞** |
| **S4** | **- ∞** |  |  | **17** |  | **24** |  |  | **42** |  |  | **55** | **+ ∞** |
| **S3** | **- ∞** |  |  | **17** |  | **24** |  |  | **42** |  |  | **55** | **+ ∞** |
| **S2** | **- ∞** | **8** |  | **17** |  | **24** | **31** |  | **42** |  |  | **55** | **+ ∞** |
| **S1** | **- ∞** | **8** | **12** | **17** |  | **24** | **31** |  | **42** | **44** |  | **55** | **+ ∞** |
| **S0** | **- ∞** | **8** | **12** | **17** | **20** | **24** | **31** | **39** | **42** | **44** | **50** | **55** | **+ ∞** |

Following is the path highlighted to delete 55

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S5** | **- ∞** |  |  |  |  |  |  |  |  |  |  |  | **+ ∞** |
| **S4** | **- ∞** |  |  | **17** |  | **24** |  |  | **42** |  |  | **55** | **+ ∞** |
| **S3** | **- ∞** |  |  | **17** |  | **24** |  |  | **42** |  |  | **55** | **+ ∞** |
| **S2** | **- ∞** | **8** |  | **17** |  | **24** | **31** |  | **42** |  |  | **55** | **+ ∞** |
| **S1** | **- ∞** | **8** | **12** | **17** |  | **24** | **31** |  | **42** | **44** |  | **55** | **+ ∞** |
| **S0** | **- ∞** | **8** | **12** | **17** | **20** | **24** | **31** | **39** | **42** | **44** | **50** | **55** | **+ ∞** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S5** | **- ∞** |  |  |  |  |  |  |  |  |  |  | **+ ∞** |
| **S4** | **- ∞** |  |  | **17** |  | **24** |  |  | **42** |  |  | **+ ∞** |
| **S3** | **- ∞** |  |  | **17** |  | **24** |  |  | **42** |  |  | **+ ∞** |
| **S2** | **- ∞** | **8** |  | **17** |  | **24** | **31** |  | **42** |  |  | **+ ∞** |
| **S1** | **- ∞** | **8** | **12** | **17** |  | **24** | **31** |  | **42** | **44** |  | **+ ∞** |
| **S0** | **- ∞** | **8** | **12** | **17** | **20** | **24** | **31** | **39** | **42** | **44** | **50** | **+ ∞** |

The 1st insertion produced 2 heads, result in inserting 8 to S 1 and S 2. The 2nd insertion produced 3 heads, results in inserting 24 at 3 more additional levels.

***C-19.7*** Suppose that there is a collection of 3n distinct coupons, n of which are colored red and

2n of which are colored blue. Suppose that each time you go to a ticket window to get a coupon, the clerk first randomly decides, with probability 1/2, whether he will give you a red coupon or blue coupon and then he chooses a coupon uniformly at random from among the coupons that are that color. What is the expected number of times that you must visit the ticket window to get all 3n coupons?

Answer:

We solve the problem with the two-coupon collector method.

We have collection of 3n distinct coupons, n of which are colored red and 2n of which are colored blue.

Assuming X to be a random variable representing the number of times that we need to visit

the ticket window before we get all n coupons. We can write

X as X = X1 + X2 + ・ ・ ・ + Xn,

where Xi is the number of trips we should make to the ticket window in order to go from having i − 1 distinct coupons to having i distinct coupons.

By linearity of expectation,

E[X] = E[X1] + E[X2] + … + E[Xn]

= + + … +

= + + …. +

= n

= n Hn

where Hn is the nth harmonic number, which, as we have observed elsewhere, can

be approximated as ln n ≤ Hn ≤ ln n+1. In other words, the expected number of

times that we need to visit the ticket window in order to get at least one instance of

each of n coupons is nHn.

Assume that, the colored tickets are distributed from two separate windows.

Because we have n red colored tickets and 2nd blue colored tickets, we would need nHn trips to the red window and 2nH2n trips to the blue window to get all 3n coupons.

The Probability of choosing either red OR blue =

Hence, from the equations, expected number of trips to get all the blue coupons = 4nH2n.

And because the probability of getting a red coupon is 1/2, the expected number of red tickets will be 2nH2n.

Because nHn < 2nH2n, you will get enough red tickets to get all n in these trips as well. Hence, the expected number to get all 3n tickets is 4nH2n.

**Chapter 25**

***R-25.8*** Compute the product of the binary numbers (01101000)2 and (10001011)2 using the algorithm given in the book.

Answer:

**Theorem 25.8:** Given two *n*-bit integers *P* and *Q*, we can compute the product

*R* = *P ・ Q* using *O* (*n* log *n*) arithmetic operations.

Here, the arithmetic operations are done in the number system that is used to

define the primitive *n*th roots of unity required by the FFT algorithm. For instance,

if we can do all the arithmetic in *Zt*, for a prime *t* = *cn* + 1, for a small integer

constant, *c*, such that *t* can be stored in a single word on our computer, then we can

perform each arithmetic operation in *O* (1) time in the RAM model.

In some cases, we cannot assume that arithmetic involving reasonably sized

words can be done in constant time, however. In such scenarios, we must pay

constant time for every bit operation. In this model it is still possible to use the

FFT to multiply two *n*-bit integers, but the details are somewhat more complicated

and the running time increases to *O* (*n* log *n* log log *n*). We omit the details for this

approach here.

01101000 × 10001011 = 11100001111000

***C-25.3***Given degree-n polynomials p (x) and q (x), describe a method for multiplying the derivatives of p(x) and q (x), that is, p (x)·q(x), using O (n log n) arithmetic operations.

Answer:

For polynomial multiplication, if p(x) and q(x) are polynomials of degree-bound n, we say that their product r(x) is a polynomial of degree-bound 2n - 1 such that r(x) = p(x)q(x) for all x in the underlying field. You have probably multiplied polynomials before, by multiplying each term in p(x) by each term in q(x) and combining terms with equal powers. For example, we can multiply p(x) = 6x3 + 7x2 - 10x + 9 and q(x) = -2x3 + 4x - 5 as follows:

6x3 + 7x2 - 10x + 9

- 2x3 + 4x - 5

-------------------------

- 30x3 - 35x2 + 50x - 45

24x4 + 28x3 - 40x2 + 36x

- 12x6 - 14x5 + 20x4 - 18x3

---------------------------------------------

- 12x6 - 14x5 + 44x4 - 20x3 - 75x2 + 86x - 45  
  
Note that degree(C) = degree(A) + degree(B), implying

degree-bound(C) = degree-bound(A) + degree-bound(B) - 1

degree-bound(A) + degree-bound(B).  
We shall nevertheless speak of the degree-bound of C as being the sum of the degree-bounds of A and B, since if a polynomial has degree-bound k it also has degree-bound k + 1.