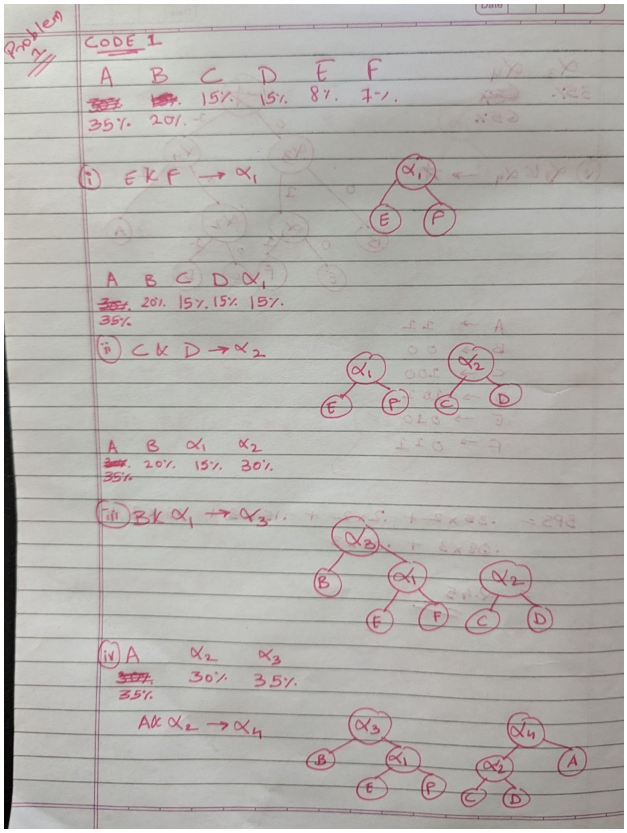
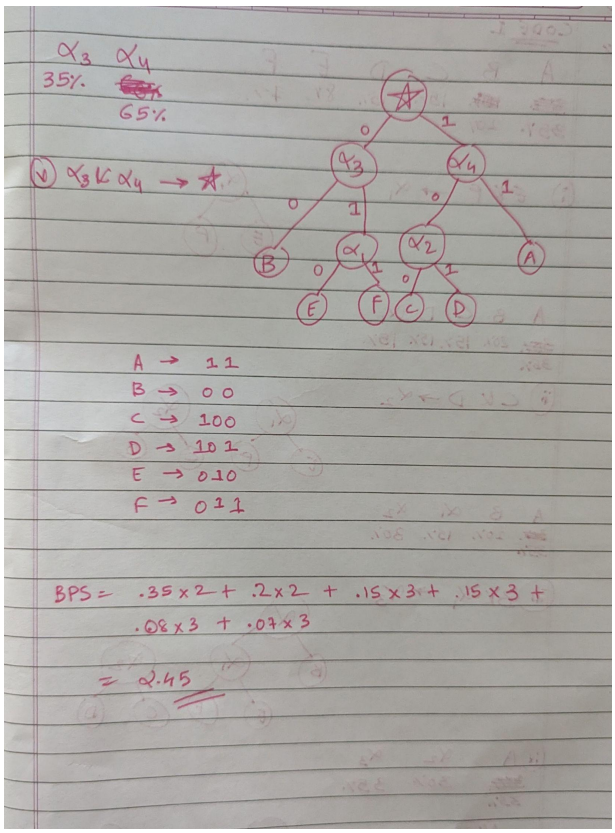
CSCI-665 Homework 3

Deepika Kini (dk1346)

Adith Shetty (as6127)

Problem 1

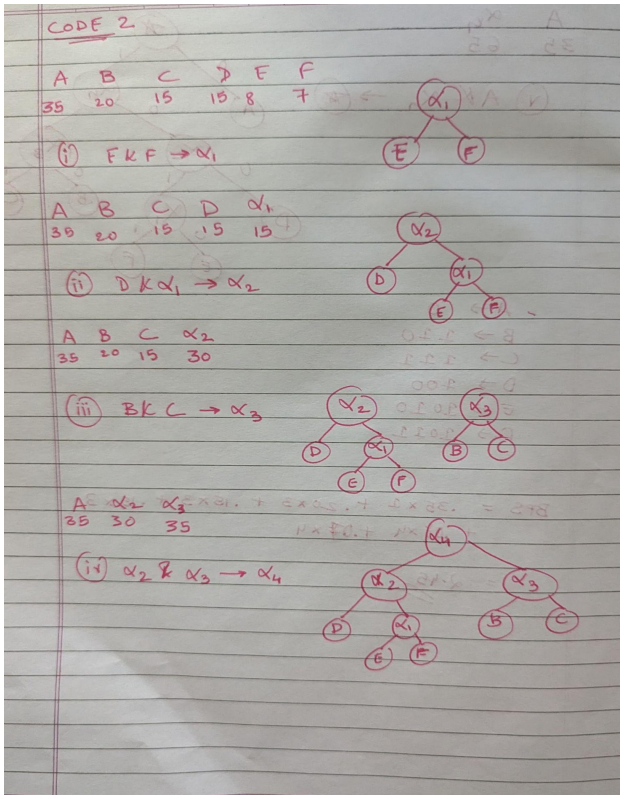


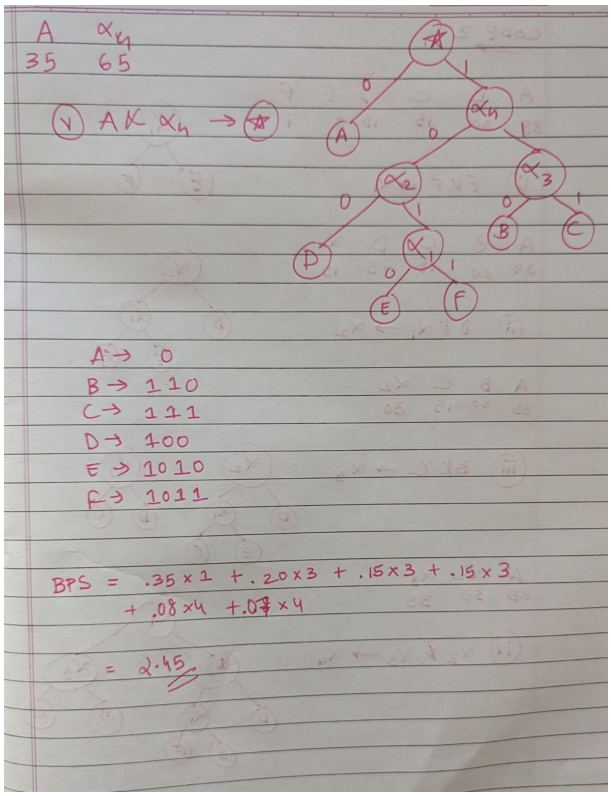


CSCI-665 Homework 3

Deepika Kini (dk1346)

Adith Shetty (as6127)



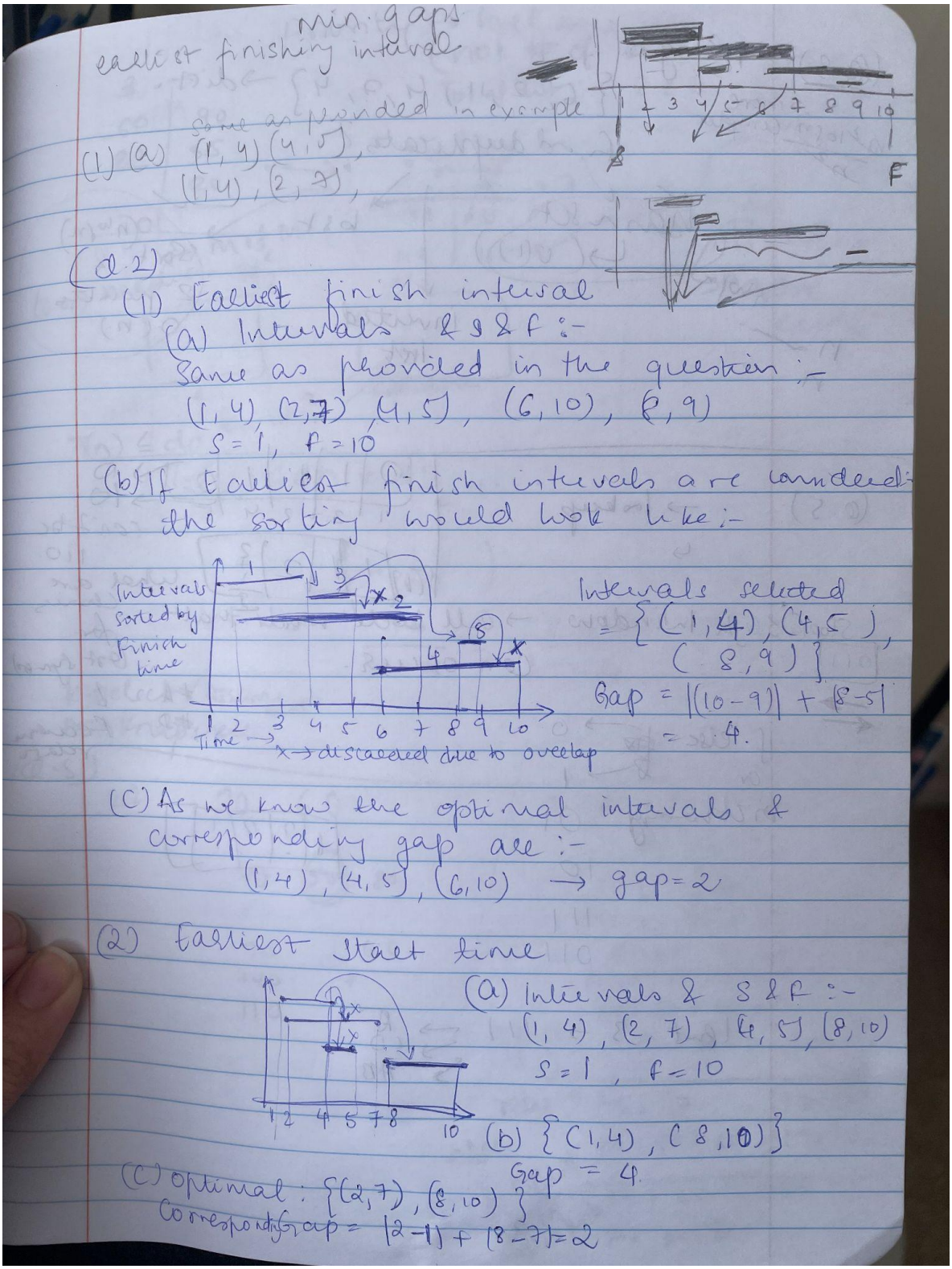


CSCI-665 Homework 3

Deepika Kini (dk1346)

Adith Shetty (as6127)

Problem2



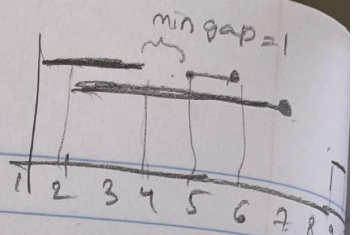
CSCI-665 Homework 3
Deepika Kini (dk1346)
Adith Shetty (as6127)

3.

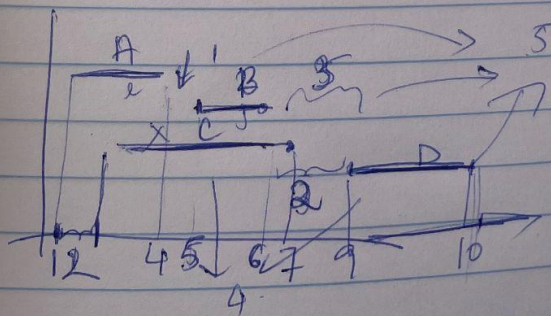
(a) (1, 4), (2, 7), (5, 6), (9, 10)
s = 1, f = 10.

(b) (1, 4), (5, 6), (9, 10)
Gap = 4

(c) optimum : (2, 7), (9, 10)
Gap = 3



Rough :-



A & B have least gap.
∴ C is discarded

j → B
i → A

CSCI-665 Homework 3
Deepika Kini (dk1346)
Adith Shetty (as6127)

Problem 3

Given are n distinct points in 2-dimensional space. Design an O(n2 log n) algorithm that determines whether any three of these points are evenly spaced, co-linear points. A Θ(n3) algorithm can earn at most 8 out of 18 marks.

BONUS: only can be earned after you successfully complete this problem in O(n2 log n) complexity. Additionally, design a solution that runs in expected O(n2) time. You may use a hash-table data structure for this part of the problem, and only for this part of the problem. Document the performance of this approach compared to your O(n2 log n) solution.

Design an O(n) algorithm that determines the number of youngest kids that end up sitting on the left side while balancing the boat as much as possible.

Hint: start out by asking yourself how you would solve this problem if the children were already sorted by increasing age. You can't afford to sort the data, however, as none of the O(n) sorts applies (you can't make any assumptions about the distribution of the age values).

O(n2 log n) solution :

The algorithm is as follows:

We use the concept of computing midpoints since it helps discover the colinearity and equidistance w.r.t. the other two points

Breaking down the steps:

We sort the 2-d array by y and then x values (O(nlogn))

We take in two point combinations and compute mid points. (O(n^2))

We check if these mid-points are present in the data array using binary search. The binary search on the 2 d array is done in such a way that it checks if x is already the mid-value. If yes, then it checks the y value (1st index) and decides to go to the left or right half. If x itself has not been found, depending on x value of point, we go to the left or right sub-array midpoints. (O(logn) for each midpoint)). The loop breaks if one mid-point is found.

This makes the algorithm O(n^2(logn)) since midpoint is searched for all n^2 combinations

CSCI-665 Homework 3
Deepika Kini (dk1346)
Adith Shetty (as6127)

O(n log n) :

Here a hashset is used to store points instead of binary search which easily finds if midpoint computed is present in the hashset. Adding and finding points in hashset is O(1) steps.
This makes it a O(n^2) algorithm.
The code is commented out in the uploaded data

For test cases

Binary search vs hashset

```
T1:
0.0002572080120444298    0.00012516698916442692

T2:
0.00023329199757426977 6.008398486301303e-05

T3:
0.035936458996729925     0.003771957999560982

T4:
0.0655184590141289       0.0016313329979311675

T5:
14.235483291995479       2.750484917021822
```

As we can see there is a vast difference between the two approaches

CSCI-665 Homework 3
Deepika Kini (dk1346)
Adith Shetty (as6127)

Problem 4

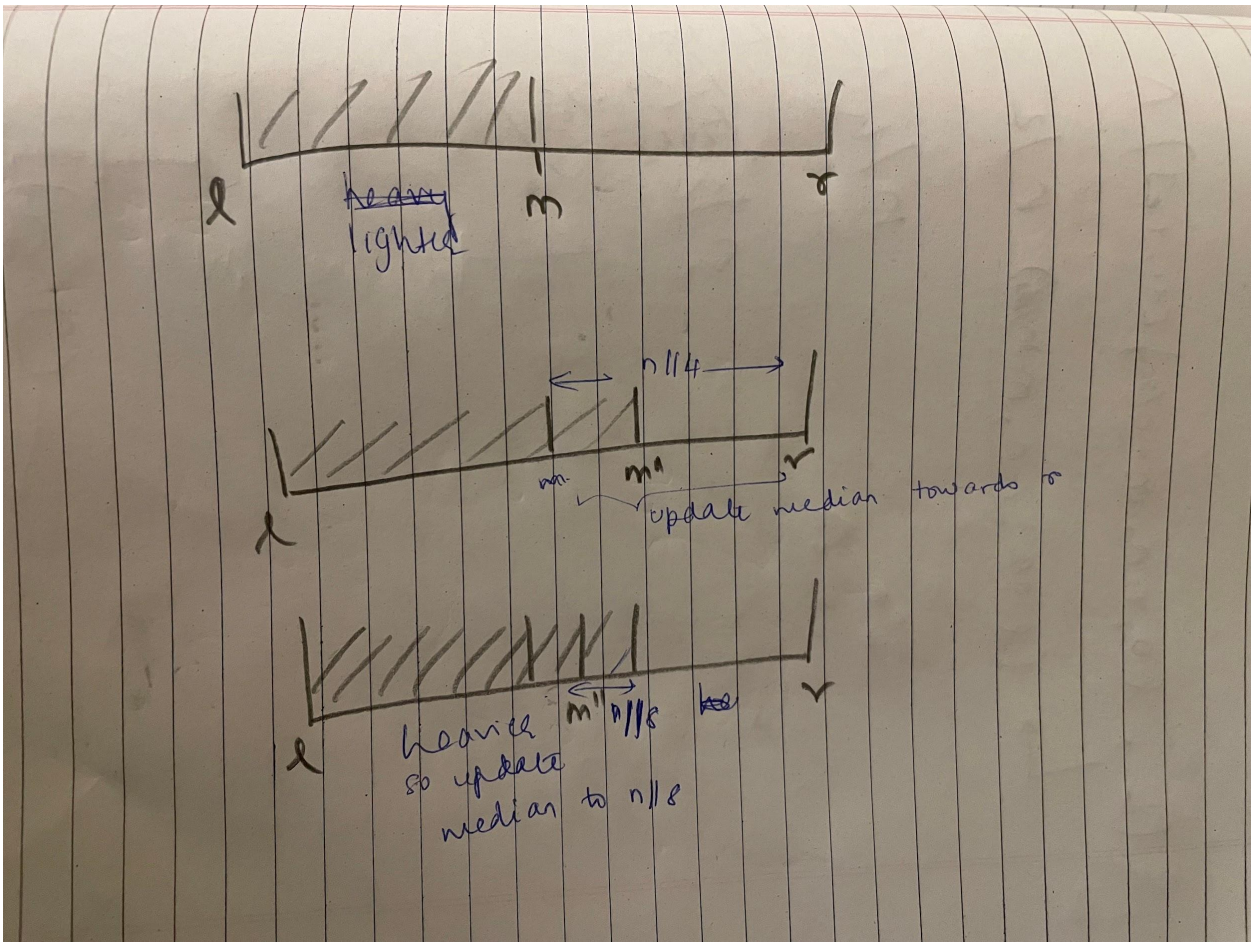
We use median value to compute the number of kids and use a recursive function to shift kids from left and right buckets to get optimum minimum variance w.r.t. the balance weight (total/2) for the left and right buckets. We consider both teachers on both sides and then compute the least difference provided combination w.r.t. Balance weight for the left side weight.

Algorithm:

We use the k select algorithm for n datapoints(kids). The expected linear time for this algorithm is O(n). If we use k select recursively then we run it for n/2, n/4... which leads to geometric progression that leads to 2 as the constant for n (1/1-alpha formula). This confirms that this

---

doesn't lead to a more complex algorithm is greater than O(n).



Problem 5

Suppose that instead of encoding with a prefix code such as determined in Problem 1, the symbols are encoded using the codewords given in the table below:

Symbol	Codeword
A	0
B	1
C	10

D 01
E 111
F 011

Given an input encoding of size n bits, design an O(n) algorithm that determines how many different decodings of those n bits are possible using the table above.

Algorithm:

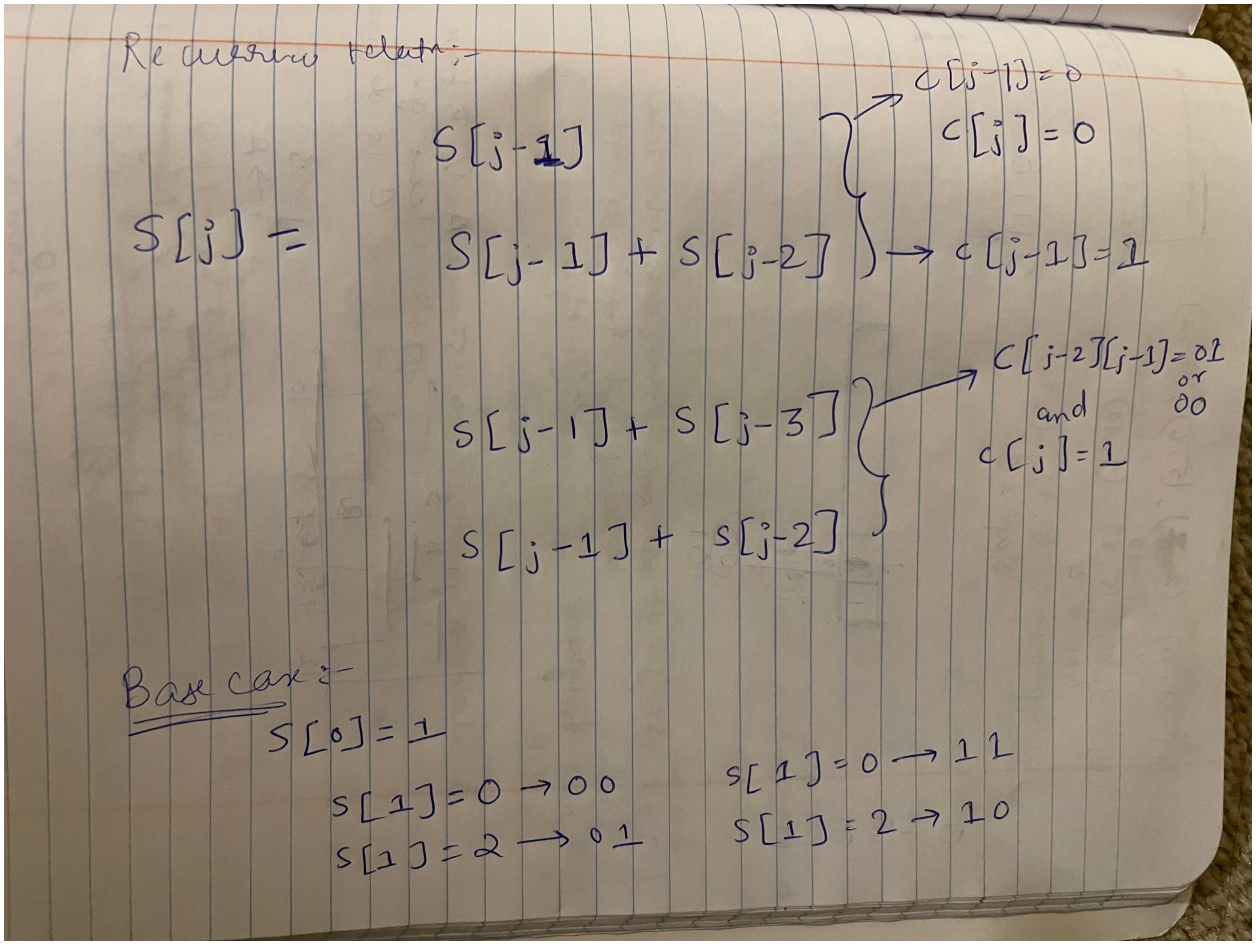
It goes through the bitstream linearly and goes back using index keeping constant time

Heart of algorithm:

Problem description:

S[j] = number of combinations of symbols that are computed (decoded) for bitstream of length from 0...j

Recurrence equation:



Final output:

S[n] = provides the combinations of codes used in bitstream of size n