Mahitha Valluru

Algorithms

Professor Or Beit Aharon

Homework 3

1.
    a. The algorithm would take the first element of the array and the last element of the array and compare them. Whichever value is smaller is the one that is kept.
    b. To find the maximum element of the array, we will have to first split the array in half, say into arrays B (first half) and C (second half). Then, compare the last value of B to the first value of C. Take the array with the larger value and divide that array into two more arrays, repeating the comparison step until you are left with an array of 1 value, which will be the maximum.
    c. Split the array in half, into arrays B (first half) and C (second half). Find the median of each array and split those arrays again. The median is calculated by $\lceil n/2 \rceil$, where n is the length of the array. In this case when we are trying to find the median, we will need to recurse only on the array where the values are increasing. Once we have reached a point in which the length of both arrays combined equals 5, we have reached the point where we can use insertion sort to sort these 5 values. We can now use $\lceil n/2 \rceil$ to find the median.
2.
    a. Let us consider two numbers:

        123 456

        And we will split them up into $n/3$ parts:

        1 2 3 4 5 6

        a b c x y z

        $abc = (100 * a) + (10 * b) + c$

        $xyz = (100 * x) + (10 * y) + z$

        What we want to find is abc * xyz, or:

        $[(100 * a) + (10 * b) + c] * [(100 * x) + (10 * y) + z]$

        By factoring, we find out that this equals:

$$\begin{aligned}&10000ax + 1000ay + 100az +\\ &1000bx \quad + \quad 100by + \quad 10bz +\\ &100cx \quad + \quad 10cy + \quad cz\end{aligned}$$

We have 9 multiplications; however, we are looking for 6 multiplications.
We need to simplify.

We notice $1000ay$ and $1000bx$. These are 2 multiplications on their own.
By writing $(a+b)(x+y)$, this is only one multiplication. This equals $ax+ay+bx+by$. As you can see, this not only calculates ay and bx, but also calculates ax and by. As we can see below, ax and by are already being calculated, so we do not have to include those in the answer.

$$\begin{array}{l} 10000\boldsymbol{ax} + 1000\boldsymbol{ay} + 100az + \\ 1000\boldsymbol{bx} \quad + \quad 100\boldsymbol{by} + \quad 10bz + \\ 100cx \quad + \quad 10cy \; + \quad cz \end{array}$$

The simplified equation now boils down to:

$$\begin{array}{l} 10000ax + 100az + \\ 100by \quad + \quad 10bz + \\ 100cx \quad + \quad 10cy + \; cz + \\ (a+b)(x+y) \end{array}$$

Which is 8 multiplications. Good, but it could be better.

Now let's look at $100az$ and $100cz$. Again, they are 2 multiplications on their own. We can write them as $(a+c)(x+z)$. Factoring out, this equals $ax+az+cx+cz$. This solves the problem in that we find az and cy, but we also find ax and cz. Because these two are already being solved, we can count these multiplications as negligible.

$$\begin{array}{l} 10000\boldsymbol{ax} + 100\boldsymbol{az} + \\ 100by \quad + \quad 10bz + \\ 100\boldsymbol{cx} \quad + \quad 10cy + \; \boldsymbol{cz} + \\ (a+b)(x+y) \end{array}$$

The simplified equation now boils down to:

$$\begin{array}{l} 10000ax + \\ 100by \quad + \quad 10bz + \\ 10cy \quad + \quad cz + \\ (a+b)(x+y) + \\ (a+c)(x+z) \end{array}$$

This is 7 multiplications. Almost there!

Finally, let's take a look at $10bz$ and $10cy$. Being 2 multiplications on their own, we can definitely simplify these.

We can write them as $(a+b+c)(x+y+z)$, which equals $ax+ay+az+bx+by+bz+cx+cy+cz$. As we can see, all the terms other than bz and cy are already calculated, as seen below.

$10000\boldsymbol{ax}+$
$100\boldsymbol{by} \quad + \quad 10\boldsymbol{bz}+$
$10\boldsymbol{cy} \quad + \quad \boldsymbol{cz}+$
$(\boldsymbol{a}+\boldsymbol{b})(\boldsymbol{x}+\boldsymbol{y})+$
$(\boldsymbol{a}+\boldsymbol{c})(\boldsymbol{x}+\boldsymbol{z})$

So, with the new factor, we can rewrite our equation to be:

$10000ax+$
$100by \quad +$
$cz \quad +$
$1000(a+b)(x+y)+$
$100(a+c)(x+z)+$
$10(a+b+c)(x+y+z)$

As we can see here, there are 6 multiplications on the smaller parts taking place, so we have succeeded.

A more abstract algorithm for any 2 n-digit numbers is:

$10^{\frac{4n}{3}}ax \quad +$
$10^{\frac{2n}{3}}by \quad +$
$cz \quad +$
$10^{n}(a+b)(x+y)+$
$10^{\frac{2n}{3}}(a+c)(x+z)+$
$10^{\frac{n}{3}}(a+b+c)(x+y+z)$

Which has 6 multiplications.

b. 6 Multiplications on n digits:

$$T(n)=6T\left(\frac{n}{3}\right)+\theta(n)$$

This simplifies to: $T(n)=\theta\left(n^{\log_3 6}\right)\approx\theta(n^{1.63})$

The Karatsuba algorithm has $\theta(n^{1.585})$ runtime, and since $1.585<1.63$, this means that the Karatsuba algorithm is better than that of the algorithm above.

3. Let Max_i be the maximum sum of the set of contiguous elements ending at index i.

$$Max_1=A[1]$$

$$\begin{aligned} Max_i &= A[i] & &if\ Max_{i-1} \leq 0 \\ Max_i &= A[i] + Max_{i-1} & &if\ Max_{i-1} > 0 \end{aligned}$$

$Max_i = A[i]\ if\ Max_{i-1} \leq 0$ because if the maximum sum of the set ending at $n-1$ is either a negative value or 0, then $A[i]$ would be the maximum sum, since that value does nothing to make the sum maximum.
$Max_i = A[i] + Max_{i-1}\ \ if\ Max_{i-1} > 0$ because the maximum sum ending at $n-1$ is positive, so you would add $A[i]$ to the maximum sum, since that value is what will make the sum greater.

The resulting lines are the pseudocode:

- Create an array $Store$ of size n that will store the values $Max_1, Max_2, \ldots, Max_n$
- Set the first element of $Store$ to be equal to the first element of A
- For the rest of the values of $Store$ ($Store[2], \ldots, Store[n]$), we will use the recurrence we have used above.

```
$for(i = 2; i++; i \leq n)\ \{$
    $if\ (Store[i-1] \leq 0\ )\ \{\ Store[i] = Max[i]\ \}$
    $if\ (Store[i-1] > 0)\ \{\ Store[i] = Max[i] + Store[i-1]\ \}$
$\}$
```

- Find the index of the maximum value in the array $Store$
- In $Store$, move left until the beginning of the array while adding the elements to the final set

```
$for(i = index; i--; i > 0)\ \{$
    $if\ (Store[i] > 0)\ \{\ result.add(Max[i])\ \}$
    $if\ (Store[i] \leq 0)\ \{\ nothing\ \}$
$\}$
```

- Return result

With this pseudocode, we can accurately find the set of contiguous elements with the maximum sum in linear time.

4. To do this problem, we will have to first think. We will need to create a two-dimensional matrix. If the two values are equal, we need to add the value to the $A[m-1][n-1]$ position of the matrix.
   - Create the matrix A of size $[m+1][n+1]$. This is because we start the matrix from 0 to m and from 0 to n.
   - Let $M[x, y]$ represent the length of the longest common substring of $a[0]$ to $a[i]$ and $b[0]$ to $b[j]$

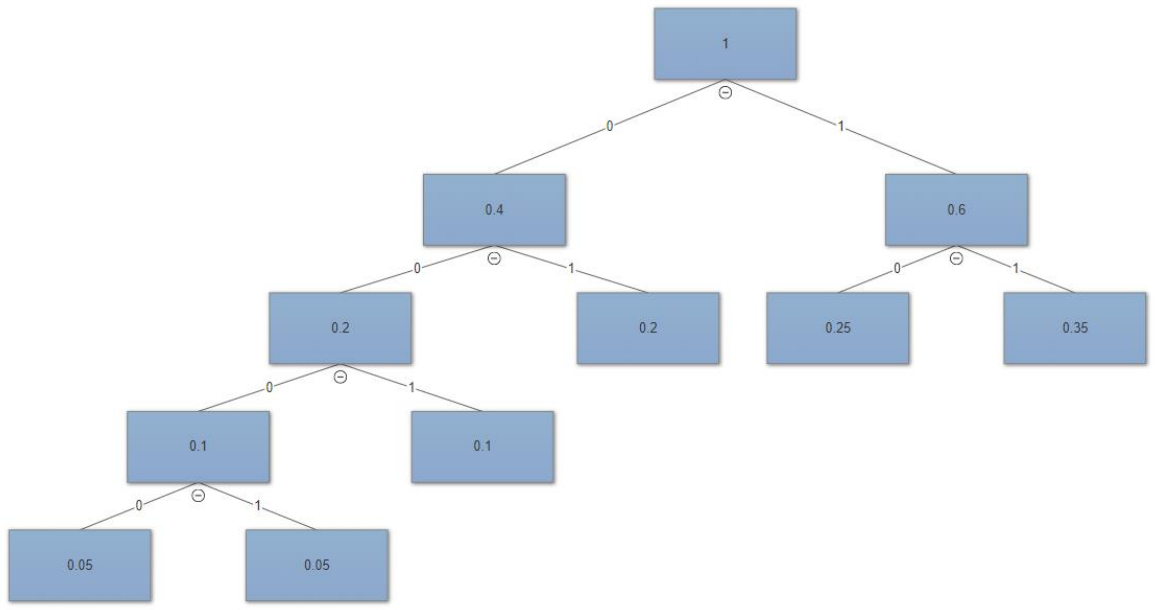
- When we fill in the matrix, what is shown below will be the format. Know that if $a[i] = b[j]$, then $M[i,j] = 1 + M[i-1, j-1]$.

```
$for\ (i = 0; i++; i \le n)\ \{$
    $for\ (j = 0; j++; j \le m)\ \{$
            $if\ (a[i] == b[j])\ \ \{M[i,j] = 1 + M[i-1, j-1]\}$
            $else\ \{M[i,j] = 0\}$

    $\}$

$\}$
```

And finally, we will return the maximum value in the array M.

5.

a. This is what the binary tree will look like:



As a result, this is what the optimal prefix codes for each probability will look like:

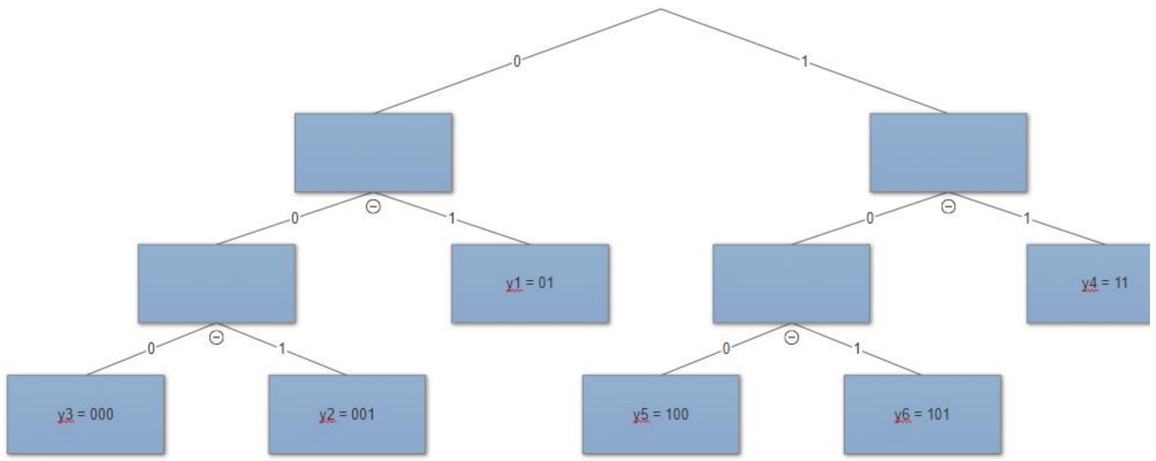
Probability	0.05	0.05	0.1	0.2	0.25	0.35
Prefix Code	0000	0001	001	01	10	11

To compute the average length:

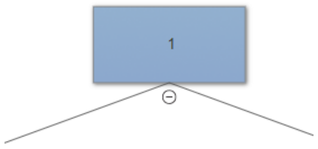
$$(0.05 * 4) + (0.05 * 4) + (0.1 * 3) + (0.2 * 2) + (0.25 * 2) + (0.35 * 2) = 2.3$$

So, the average length of a code word is 2.3.

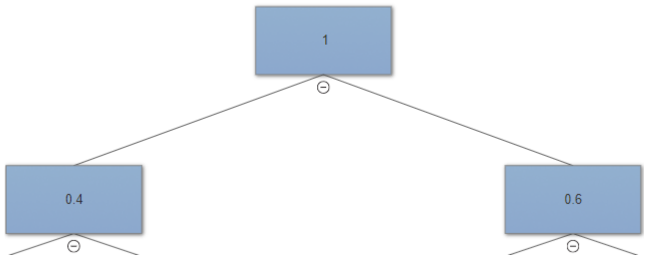
b. To do this problem, I will have to work backwards. This is what the decision tree will look like:



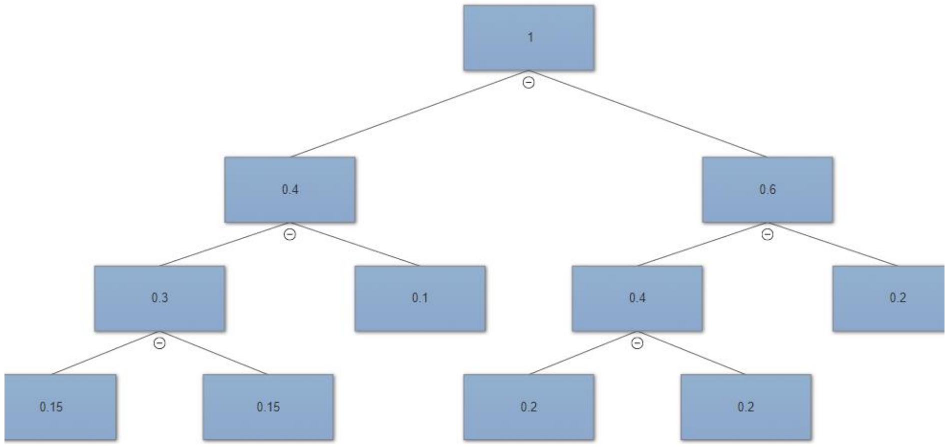
With this, we can now find the probability vector for the characters y_1 to y_6. I started off with the very top as equal to 1, as all probability vectors added up should be equal to 1.



Then, for the two sides, I can set them equal to any fraction if they add up to 1.



And I keep doing the same until I reach the end.

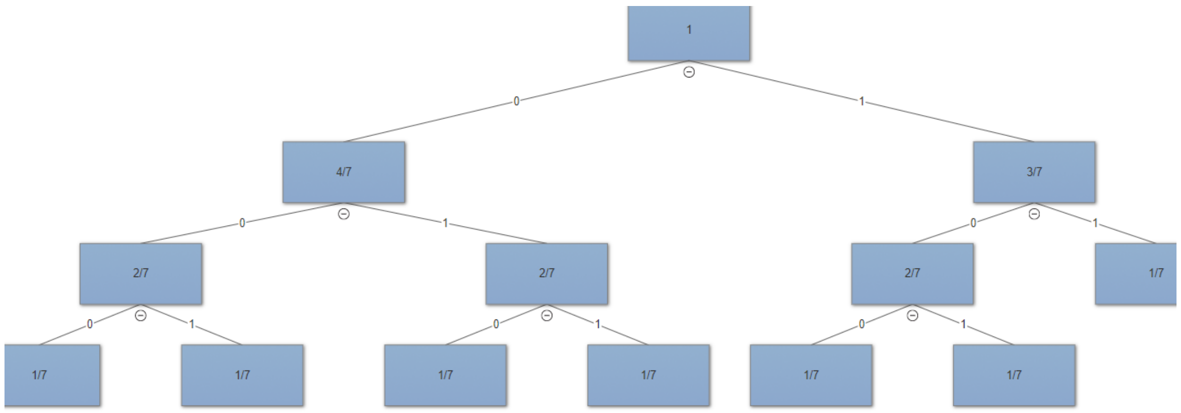


So, the probability vectors in the end will be:

y	y_1	y_2	y_3	y_4	y_5	y_6
Prefix code	01	001	000	11	100	101
p	0.1	0.15	0.15	0.2	0.2	0.2

$$p = (0.1, 0.15, 0.15, 0.2, 0.2, 0.2)$$

c. To solve this problem, we need to make sure that the probability vectors of all 7 values are at a maximum. To do this, we will need to visualize the decision tree of 7 values.



We start at the top with a sum of 1. The left node will have a sum of $\frac{4}{7}$ and the right node will have a sum of $\frac{3}{7}$. We then split the sums up so that each leaf has a p value of $\frac{1}{7}$. This is the maximum value of each p value that sum together to equal 1.

To calculate the maximum $L(p)$, we have:

$$L(p) = 3\left(\frac{1}{7}\right) + 3\left(\frac{1}{7}\right) + 3\left(\frac{1}{7}\right) + 3\left(\frac{1}{7}\right) + 3\left(\frac{1}{7}\right) + 3\left(\frac{1}{7}\right) + 2\left(\frac{1}{7}\right)$$
$$= \frac{20}{17}$$

We have the maximum possible value equals to $\frac{20}{17}$, which is equivalent to $\frac{40}{17}$, and given the problem at hand, this maximum is less than $\frac{41}{17}$.

So, because the maximum p values of p for each leaf given there must be 7 leaves is less than $\frac{41}{17}$, all probability vectors of the form $p = (p_1, p_2, \dots, p_7)$ will have $L(p) < \frac{41}{17}$.

Discussed problems alongside Ben Hackney, Marie Yatsyk, Chris Risley, Joe Annis, and Sebastian Ruiz.