Exercise 1

Case 1 The idea is to apply mergesort on disk by considering strings as atomic objects that can be compared in constant time. After having sorted the pairs by their first component (string), it is enough to scan and accumulate the counts for every string, and keep the max. The cost is therefore $O\left(\frac{n}{B}\log_{\frac{M}{B}}\frac{n}{B} + \frac{n}{B}\right)$ I/Os

Case 2 [another can Spectral Bloom Filter]

We compute the hashing of every string, it is enough to consider an hash function over a co-domain $n^2 \longrightarrow$ 2 log n bits. This way we can bound, as in the perfect hashing, the probability of a collision, and thus have a perfect hashing.
So we scan the strings, we compute their hashing and check whether the string is already in memory (with its hash and cumulative count). If it is the first time it is met, then we store in memory, its hash and its satellite value; if it is already present we sum its satellite value to the current count. The cost is therefore the one of the scan, so $O\left(\frac{n}{B}\right)$ I/Os, given that the maximum can be computed in memory, and a final scan finds the string whose hash corresponds to the maximum sum. [we could have been more precise in the I/O-complexity by writing $O\left(\frac{|S|}{B}\right)$ where $|S|$ is the total length of the input.]

Case 3

We use a Patricia Trie where we keep all the distinct strings, and their cumulative sum.

More precisely, we scan S and for every string $P \in S$ we search P in the patricia trie currently stored in memory. This takes $O\left(\frac{|P|}{B}\right)$ I/Os. Either P occurs in the trie, and thus its cumulative sum is updated with its associated occ, or it is a new string which is then inserted in the patricia trie with sum = occ.

Eventually we compute in memory the maximum sum, and on the pointer to the string, corresponding to that sum, to recover it. The cost is $O\left(\frac{|S|}{B}\right)$ I/Os, where $|S|$ is the total string length and $|S| \geq n$.

Exercise 2

	1	2	3	4	5	6	7
$S =$	1	2	3	4	6	10	11

•) low = 1, high = 11, l = 1, r = 7

$m = \frac{1+7}{2} = 4 \longrightarrow S[4] = 4$

range = $[low + m - l, hi - r + m] = [4, 8]$

encode $4 - 4 = 0$ using $\lceil \log_2 8-4+1 \rceil = 3$ bits

•) low = 1, high = 3, l = 1, r = 3

↳ no bits emitted for this subsequence.

•) low = 5, high = 11, l = 5, r = 7

$m = \frac{5+7}{2} = 6 \longrightarrow S[6] = 10$

range = $[5 + 6 - 5, 11 - 7 + 6] = [6, 10]$

encode $10 - 6 = 4$ using $\lceil \log_2 10-6+1 \rceil = 3$ bits

•) low = 5, high = 9, $l = r = 5$

$m = 5 \rightarrow S[5] = 6$

range = $[5, 9] \rightarrow$ encode $6 - 5 = 1$ in $\lceil \log_2 5 \rceil = 3$ bits

•) low = 11, high = 11, $l = r = 7$

↳ no bits emitted

■ $\Delta S = 1, 1, 1, 2, 4, 1$

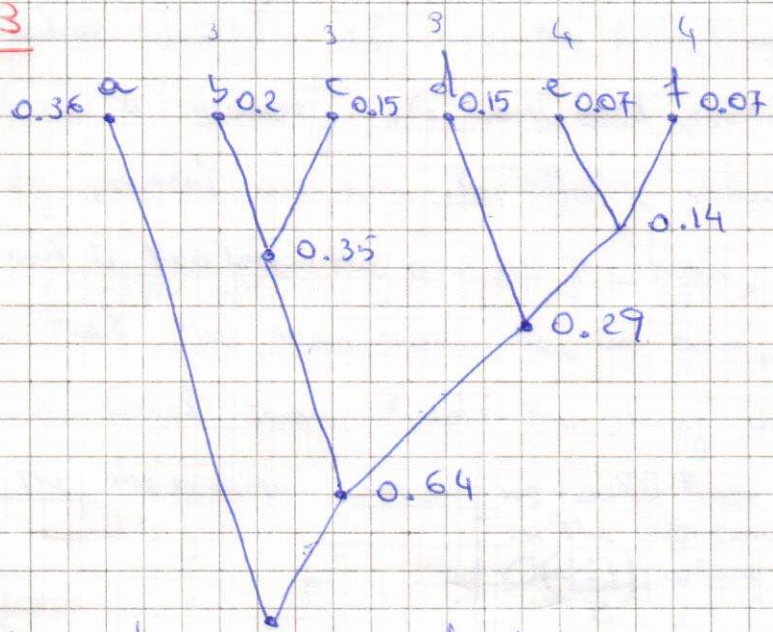
subtract base = 1

$\Delta S' = 0, 0, 0, 1, 3, 0$

Since $b = 2$, in two bits we can encode 0, 1, 2 and reserve the configuration for 3 as escape.

$\Rightarrow$ 00, 00, 00, 01, 11, 00 | 3

Exercise 3



	1	2	3	4
name	1	0	3	2
symb	a	-	b	e
			c	f
			d	

	1	2	3	4
fc	1	2	1	0
	$\frac{2+0}{2}$	$\frac{1+3}{2}$	$\frac{0+2}{2}$	

Explicit cw

1	1
2	10
3	001
4	0000

dummy value