Bit tricks and partial sums

Nicola Prezza

References and Reading

[1] Chapter 3 of: Navarro, Gonzalo. Compact data structures: A practical approach. Cambridge University Press, 2016.

Exercises

1 Bit tricks The operation `popcount` takes as input a word X of w bits and outputs the number of 1s (i.e. bits set) in X. Assuming that $w = 32$, write the pseudocode of a function that computes `popcount(X)`. Tabulation (i.e. pre-computation of the answers in a table) is not allowed. The function should use only standard bitwise (`and`, `or`, `shift`) and arithmetic (addition, subtraction) operations. Try to use as few operations as possible (hint: for generic w, it is possible to use just $O(\log w)$ instructions).

2 PforDelta partial sum The `PforDelta` code (see Chapter 2) encodes a sequence of n integers $x_1, \ldots, x_n$ as follows. We divide the numbers in consecutive blocks of size k (assume that k divides n). Inside each block, we encode all integers using—for each of them—the bit-size of the largest integer in the block. The encoded $x_1, \ldots, x_n$ are then concatenated in a single packed bitvector B. Finally, we store the beginning of each block in B using n/k additional words.

Without loss of generality, assume that $x_i \geq 1$, for all $1 \leq i \leq n$, and let $N = \sum_{i=1}^{n} x_i$. Fix moreover the block size as $k = w$.

2.1 Compute the bit-size of the `PforDelta` representation.

2.2 What is the space overhead on top of the worst-case entropy of all integer sequences of length n that add up to N?

2.3 Show how to compute efficiently `access`[1], `sum`, and `search` operations on top of the above representation.

[1] `access` simply outputs x_i given i as input