koutecky@iuuk.mff.cuni.cz

11. tutorial

Mini-lecture on Open Addressing Hashing.

Bloom Filter. Bloom Filter is a data structure for approximate set representation. It consists of an array of m bits $B[1, \ldots, m]$ and a hash function h which assigns indices in this array to elements of the universe. `INSERT(`x`)` sets $B[h(x)] = 1$. `MEMBER(`x`)` tests whether $B[h(x)] = 1$.
Assume we have inserted a certain n-element set M into the filter. If $x \in M$, then `MEMBER(`x`)` always answers "yes" correctly. However, if we ask about some $x \notin M$, it may happen that $h(x) = h(y)$ for some $y \in M$, and thus `MEMBER(`x`)` will answer "yes" while the correct answer is "no". Compute the probability of this happening for a given m and n.
Hint: use the fact that $1 + \alpha \leq e^{\alpha}$, $\forall \alpha \in \mathbb{R}$.

Improved Bloom Filter. A "Bloom filter" is typically something a little better than the above. We still have one array B, but now we get k independent random hashing functions $h_1, \ldots, h_k$. When inserting an element x, we set $B[h_i(x)] = 1$ for every $i = 1, \ldots, k$, and when checking membership we require that all those bits are set to 1.
If the probability of a false positive of the basic filter with just one hashing function is p, then the probability of a false positive of this improved filter is p^k. Figure out how to set m and k if we want to store $n = 10^6$ elements and we want to keep the probability of a false positive at 10^{-9}. Minimize the memory consumption.