Sqrt-heuristics, examples and applications
Lecture notes

Gleb Evstropov
glebshp@yandex.ru

November 12, 2016

In general, sqrt-heuristcs are methods to reduce the complexity and obtain something containing $\sqrt{n}$ (or close) multiplier, hence the name. During the lecture we will consider different examples illustrating different way to approach problems.

Sqrt-decomposition is way to build two-layer data structure that combines two different approaches to answer queries in order to optimize the complexity.

Problem 1. You are given a sequence of positive integers $a_1, a_2, \ldots, a_n$. You should answer online for queries of two types:

1. Given some index p and positive integer x, set $a_p = x$.
2. Throw a ball to position p. Then it moves to position $p + a_p$, then $p + a_p + a_{p+a_p}$ and so on, i.e. each time it jumps a_i positions forward. Print the number of jumps before the ball leaves the sequence.

Solution sketch: make $\sqrt{n}$ buckets, in each bucket compute the number of jumps to leave the bucket and the resulting position.

Problem 2. Implement Dijkstra's algorithm in $O(n\sqrt{n} + m)$ time. Solution sketch: use sqrt-decomposition to build a data structure that removes mininum in $O(\sqrt{n})$ time and relax minimum in $O(1)$.

Problem 3. Batching queries. Given a tree T of size n answer queries of two types:

1. Paint vertex v black.
2. Find the nearest black vertex for vertex v.

Solution sketch: rebuild the minimum distances each $\sqrt{n}$ paint queries.

Problem 4. Offline batching. Given a tree T of size n answer queries of three types:

1. Paint vertex v black.

2. Paint vertex v white.

3. Find the nearest black vertex for vertex v.

Solution sketch: rebuild the minimum distances each $\sqrt{n}$ paint queries, always put interesting vertices in active set.

Note: both tree problems can be efficiently solved centroid-decomposition technique but this is a topic for another lecture.

Usually, you might be not given any direct queries, but the input is processed as separate instances. Dividing instances in small and large and processing them in different ways might lead to complexity optimizations.

Problem 5. You are given a sequence of colors $a_1, a_2, \ldots, a_n$. We say that color c dominates segment $[l, r]$ if the number of elements of this color on segment is strictly greater than the number of all other elements. For each color, compute the number of segments it dominates. Solution sketch: process small and large queries in different way. If there are few elements of some color it can dominate only short segments.

Problem 6. Given an undirected graph $G = (V, E)$ compute the number of triangles, i.e. the cycles of length 3. Sketch of solution 1: split vertices in bold and normal (by value $deg(v)$). Compute the triangles that have at least one normal vertex by trying all pairs of neighbours. Compute triangles on bold vertices by trying all triples. Sketch of solution 2: for each edge uv compute the answer in $min(deg(u), deg(v))$.

To process range-queries there is a special sqrt-trick usually referred as Mo's algorithm.

Problem 7. Given a sequence $a_1, a_2, \ldots, a_n$ and a set of range queires (l_i, r_i), for each query compute $\sum_c count(l_i, r_i, c)^2$. Solution sketch: use the Mo's technique to move left and right borders of the segment by no more than 1 at each step. Break left ends in sqrt-buckets and process them separately. Go online with extra $\sqrt{n}$ in memory complexity.

Here are just some extra problems in case there will be enough time left:

Problem 8. Given a set of string $s_1, s_2, \ldots, s_n$ of total length L, proove that each path in the compressed trie containing all these strings has no more than $\sqrt{L}$ nodes.

Problem 9. Given a tree T of size n with each vertex painted some color, answer the queries (c_1, c_2 — the number of pairs (v, u) such that u lies in the subtree of v, v has color c_1 and u has color c_2.