12. 3-Sum

Lecture on Monday 10th November, 2008 by Michael Hoffmann <hoffmann@inf.ethz.ch>

The 3-Sum problem is the following: Given a set S of n integers, does there exist a three-tuple of elements from S that sum up to zero? By testing all three-tuples this can obviously be solved in $O(n^3)$ time. If the tuples to be tested are picked a bit more cleverly, we obtain an $O(n^2)$ algorithm.

Let $(s_1, \ldots, s_n)$ be the sequence of elements from S in increasing order. Then we test the tuples as follows.

```
For i = 1, ..., n − 2 {
    j = i + 1, k = n.
    While k > j {
        If s_i + s_j + s_k = 0 then exit with triple s_i, s_j, s_k.
        If s_i + s_j + s_k > 0 then k = k − 1 else j = j + 1.
    }
}
```

The runtime is clearly quadratic (initial sorting can be done in $O(n \log n)$ time). Regarding the correctness observe that the following is an invariant that holds at begin of the inner loop: There exists no suitable triple that contains s_i and s_ℓ for any $\ell < j$ or $\ell > k$.

Interestingly, this is the essentially the best algorithm known for 3-Sum. It is widely believed that the problem cannot be solved in sub-quadratic time, but so far this has been proven in some very restricted models of computation only[1].

12.1 3-Sum hardness

There is a whole class of problems that are equivalent to 3-Sum up to sub-quadratic time reductions [?]; such problems are referred to as **3-Sum-hard**.

Definition 12.1 *A problem P is 3-Sum-hard if and only if every instance of 3-Sum of size n can be solved using a constant number of instances of P—each of $O(n)$ size—and $o(n^2)$ additional time.*

As an example, consider the Problem **GeomBase**: Given n points on the three horizontal lines $y = 0$, $y = 1$, and $y = 2$, is there a non-horizontal line that contains at least three of them?

GeomBase can be reduced to 3-Sum as follows. For an instance $S = \{s_1, \ldots, s_n\}$ of 3-Sum, create an instance P of GeomBase in which for each s_i there are three points in P: $(s_i, 0)$, $(-s_i/2, 1)$, and $(s_i, 2)$. If there are any three collinear points in P, there must be one from each of the lines $y = 0$, $y = 1$, and $y = 2$. So suppose that $p = (s_i, 0)$,

[1] such as the linear decision tree model [?]

$q = (-s_j/2, 1)$, and $r = (s_k, 2)$ are collinear. The inverse slope of the line through p and q is $\frac{-s_j/2-s_i}{1-0} = -s_j/2 - s_i$ and the inverse slope of the line through q and r is $\frac{s_k+s_j/2}{2-1} = s_k + s_j/2$. The three points are collinear if and only if the two slopes are equal, that is, $-s_j/2 - s_i = s_k + s_j/2 \iff s_i + s_j + s_k = 0$.

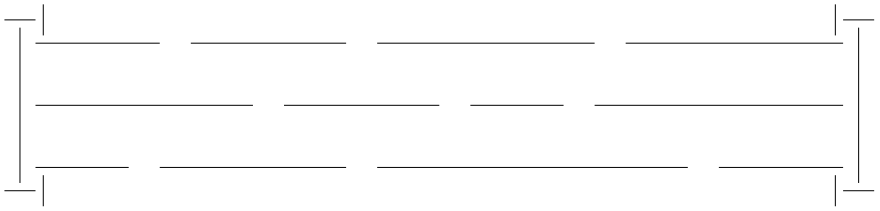
A very similar problem is **General Position**, in which one is given n arbitrary points and has to decide whether any three are collinear. For an instance S of 3-Sum, create an instance P of General Position by projecting the numbers s_i onto the curve $y = x^3$, that is, $P = \{(a, a^3) \mid a \in S\}$.

Suppose three of the points, say, (a, a^3), (b, b^3), and (c, c^3) are collinear. This is the case if and only if the slopes of the lines through each pair of them are equal. (Observe that a, b, and c are pairwise distinct.)

$$\begin{aligned}
(b^3 - a^3)/(b - a) &= (c^3 - b^3)/(c - b) \iff \\
b^2 + a^2 + ab &= c^2 + b^2 + bc \iff \\
b &= (c^2 - a^2)/(a - c) \iff \\
b &= -(a + c) \iff \\
a + b + c &= 0.
\end{aligned}$$

Minimum Area Triangle is a strict generalization of General Position and, therefore, also 3-Sum-hard.

In **Segment Splitting/Separation**, we are given a set of n line segments and have to decide whether there exists a line that does not intersect any of the segments but splits them into two non-empty subsets. To show that this problem is 3-Sum-hard, we can use essentially the same reduction as for GeomBase, where we interpret the points along the three lines $y = 0$, $y = 1$, and $y = 2$ as sufficiently small "holes". The parts of the lines that remain after punching these holes form the input segments for the Splitting problem. Horizontal splits can be prevented by putting constant size gadgets somewhere beyond the last holes, see the figure below. What "sufficiently small" means for the size



of those holes can be expressed in terms of the distance between a closest pair of points. (The closest pair of points is determined by the closest pair of numbers in the 3-Sum input. This pair can be obtained in $O(n \log n)$ time via sorting).

In **Segment Visibility**, we are given a set S of n horizontal line segments and two segments $s_1, s_2 \in S$. The question is: Are there two points, $p_1 \in s_1$ and $p_2 \in s_2$ which can see each other, that is, the open line segment $\overline{p_1 p_2}$ does not intersect any segment from S. The reduction from 3-Sum is the same as for Segment Splitting, just put s_1 above and s_2 below the segments along the three lines.

In **Motion Planning**, we are given a robot (line segment), some environment (modeled as a set of disjoint line segments), and a source and a target position. The question is: Can the robot move (by translation and rotation) from the source to the target position, without ever intersecting the "walls" of the environment?

To show that Motion Planning is 3-Sum-hard, employ the reduction for Segment Splitting from above. The three "punched" lines form the doorway between two rooms, each modeled by a constant number of segments that cannot be split, similar to the boundary gadgets above. The source position is in one room, the target position in the other, and to get from source to target the robot has to pass through a sequence of three collinear holes in the door (suppose the doorway is sufficiently small compared to the length of the robot).

12.2 Translational motion planning

In a basic instance of motion planning, a robot—modeled as a simple polygon R—moves by translation amidst a set $\mathcal{P}$ of polygonal obstacles. The placement of R is fully determined by a vector $\vec{v} = (x, y) \in \mathbb{R}^2$ specifying the translation of R with respect to the origin. Therefore $\mathbb{R}^2$ is called the *configuration space* of this robot.

For an obstacle $P \in \mathcal{P}$ the set $\mathcal{C}(P) = \{\vec{v} \in \mathbb{R}^2 \mid R + \vec{v} \cap P \neq \emptyset\}$ in configuration space corresponds to the obstacle P in the original setting. We write $R + \vec{v}$ for the *Minkowski sum* $\{\vec{r} + \vec{v} \mid \vec{r} \in R\}$. Our interest is focused on the set $\mathcal{F} = \mathbb{R}^2 \setminus \bigcup_{P \in \mathcal{P}} \mathcal{C}(P)$ of *free placements* in which the robot does not intersect any obstacle.

Proposition 12.2 $\mathcal{C}(P) = P - R$.

Proof. $\vec{v} \in P - R \iff \vec{v} = \vec{p} - \vec{r}$, for some $\vec{p} \in P$ and $\vec{r} \in R$. On the other hand, $R + \vec{v} \cap P \neq \emptyset \iff \vec{r} + \vec{v} = \vec{p}$, for some $\vec{p} \in P$ and $\vec{r} \in R$. □

It follows that $\mathcal{F}$ is bounded by at most $O(nm)$ line segments, if R has m edges and the objects in $\mathcal{P}$ have n edges in total.

12.3 Arrangements of Line Segments

Definition 12.3 *A (n, s)-Davenport-Schinzel sequence is a sequence over an alphabet A of size n in which*

- *no two consecutive characters are the same and*
- *there is no alternating subsequence of the form $\ldots a \ldots b \ldots a \ldots b \ldots$ of $s + 2$ characters, for any $a, b \in A$.*

Let $\lambda_s(n)$ be the length of a longest (n, s)-Davenport-Schinzel sequence.

For example, $abcbacb$ is a $(3,4)$-DS-sequence but not a $(3,3)$-DS-sequence because it contains the subsequence $bcbcb$.