1 Problems equivalent to APSP

Definition 1 (All-Pairs Shortest Paths (APSP))**.** Given a directed n-node graph $G = (V, E)$ with integer edge weights $w : E \to \{-M, \ldots, M\}$ and with no negative cycles, find for every pair of nodes $u, v \in V$ the length $d(u, v)$ of the shortest path from u to v.

Floyd-Warshall algorithm solves APSP in $O(n^3)$ time (do you remember it from undergrad algorithms class?). The best known APSP algorithm up to date is due to Ryan Williams and runs in $O(n^3/2^{\alpha\sqrt{\log n}})$, for some constant $\alpha > 0$.

It is conjectured that APSP cannot be solved in truly subcubic time. To make the hypothesis precise, we need to account for the range of edge weights M, because for too small weights the problem can be solved faster (we will see that later in the course) and for too large weights (e.g., exponential in n) the hypothesis is trivially true.

Hypothesis 1 (APSP Hypothesis)**.** *For every $\varepsilon > 0$ there exists c such that APSP with weights in $\{-n^c, \ldots, n^c\}$ cannot be solved in $\mathcal{O}(n^{3-\varepsilon})$ time.*

In this lecture we will see that the following problems are *subcubic equivalent* to APSP, meaning that if one of these problems has a truly subcubic algorithm, then all of them have.

Definition 2 ($(\min, +)$-product)**.** Given two $n \times n$ matrices A, B with integer entries in $\{-M, \ldots, M\}$, compute their $(\min, +)$-product $A \star B$, that is the $n \times n$ matrix such that

$$(A \star B)[i][j] = \min_{k \in [n]} (A[i][k] + B[k][j]).$$

Definition 3 (Negative Triangle)**.** Given a directed n-node graph $G = (V, E)$ with integer edge weights $w : E \to \{-M, \ldots, M\}$, decide if there exists a 3-cycle $v_1 \to v_2 \to v_3 \to v_1$ such that $w(v_1, v_2) + w(v_2, v_3) + w(v_3, v_1) < 0$.

Definition 4 (All-Pairs Negative Triangle)**.** Given a tripartite graph $G = (V_1 \cup V_2 \cup V_3, E)$ with integer edge weights $w : E \to \{-M, \ldots, M\}$, determine for every pair of nodes $v_1 \in V_1$, $v_2 \in V_2$ if there exists a 3-cycle $v_1 \to v_2 \to v_3 \to v_1$ such that $w(v_1, v_2) + w(v_2, v_3) + w(v_3, v_1) < 0$.

Definition 5 (Radius)**.** Given a directed n-node graph $G = (V, E)$ with integer edge weights $w : E \to \{-M, \ldots, M\}$ and with no negative cycles, compute

$$\min_{u \in V} \max_{v \in V} d(u, v).$$

In other words, find the smallest number r such that there exists vertex u such that every vertex is within distance at most r from u.

Remark. There are more problems known to be subcubic equivalent to APSP, e.g., Shortest Cycle, Second Shortest Path, Metricity, Replacement Paths. The fact that for none of them we have found a subcubic algorithm makes the APSP Hypothesis more believable.

2 APSP and $(\min, +)$-product have equal running times

Theorem 1. *If APSP is in $T(n)$ time, then $(\min, +)$-product is in $\mathcal{O}(T(n))$ time.*

Proof. Build a tripartite graph with vertex set $U \cup V \cup W$, $U = \{u_1, \dots u_n\}$, $V = \{v_1, \dots v_n\}$, $W = \{w_1, \dots w_n\}$. For every i, j add edge (u_i, v_j) with weight $A[i][j]$ and edge v_i, w_j with weight $B[i][j]$. Observe that $(A \star B)[i][j] = d(u_i, w_j)$. □

Theorem 2. *If $(\min, +)$-product is in $T(n)$ time, then APSP is in $\mathcal{O}(T(n) \log n)$ time.*

Proof. Repeated squaring of the (weighted) adjacency matrix with added zero-weight loops. □

Actually one can prove a stronger statement, without the $\log n$ overhead. We will see that later in the course.

3 $(\min, +)$-product is equivalent to Negative Triangle

We will show a cycle of three reductions involving $(\min, +)$-product, Negative Triangle, and All-Pairs Negative Triangle, proving that the three problems are subcubic equivalent.

Theorem 3. *If $(\min, +)$-product is in $T(n)$, then Negative Triangle is in $\mathcal{O}(T(n))$.*

Proof. Let A be the weighted adjacency matrix, that is $A[i][j] = w(i, j)$ if $(i, j) \in E$ and $A[i][j] = \infty$ otherwise. Compute $A \star A$, then check if there exists a pair i, j such that $A[i][j] + (A \star A)[j][i] < 0$. □

Theorem 4. *If All-Pairs Negative Triangle is in $T(n)$ time, then $(\min, +)$-product is in $\mathcal{O}(T(n) \log M)$ time.*

Proof. We build a tripartite graph with the vertex set $I \cup J \cup K$, where each of I, J, K is an independent copy of $\{1, 2, \dots, n\}$. For every pair $(i, k) \in I \times K$ we put $w(i, k) = A[i][k]$, and for every pair $(k, j) \in K \times J$ we put $w(k, j) = B[k][j]$. In order to compute $A \star B$ we will find, for each pair $(i, j) \in I \times J$, the smallest value $-w(j, i)$ such that there exists a negative triangle (i, j, k), i.e. $w(i, k) + w(k, j) < -w(j, i)$. We will binary search for these values, simultaneously for all pairs (i, j).

First we initialize lower and upper bounds: $l(i, j) = -2M$, $h(i, j) = 2M$. Then, for $\log(4M)$ steps, we (1) set, for all i, j, $w(j, i) = -(l(i, j) + h(i, j))/2$; (2) compute All-Pairs Negative Triangle; and (3) set, for all i, j, $h(i, j) = -w(j, i)$ if i, j belongs to a negative triangle, and $l(i, j) = -w(j, i)$ otherwise. □

Theorem 5. *If Negative Triangle is in $\mathcal{O}(n^{3-\varepsilon'})$ time, then All-Pairs Negative Triangle is in $\mathcal{O}(n^{3-\varepsilon})$ time.*

Proof. First, note that if we can *detect* if there is a negative triangle (one bit on output) in $\mathcal{O}(n^c)$ time, then we can also *find* a negative triangle (three integers on output) in $\mathcal{O}(n^c)$ time. Indeed, we split the vertex set into four roughly equal-sized parts, and we run negative triangle detection on four graphs, each composed of different three out of the four parts. If there was a negative triangle in the initial graph, then there must be a

negative triangle in at least one of those four graphs, and we recurse on one such graph. The running time is $\mathcal{O}(n^c + (\frac{3}{4}n)^c + (\frac{9}{16}n)^c + \ldots) = \mathcal{O}(n^c)$.

To solve All-Pairs Negative Triangle, we split each part of the tripartite input graph into $n^{2/3}$ groups of $n^{1/3}$ nodes each. For each triple of groups, as long as there is a negative triangle (i, j, k) in the graph induced by the three groups, we remove edge (i, j), and repeat. Hence, for a triple that yields t negative triangles, we run the negative triangle finding algorithm $t + 1$ times, each time on an $n^{1/3}$-node graph. It is important that whenever we remove an edge, it remains removed also for all other triples that we will examine later and that could possibly include that edge.

The total number of triples is $(n^{2/3})^3 = n^2$, and the total number of negative triangles reported is at most n^2 (at most one for each edge of the input graph). Each call to the negative triangle finding algorithm takes time $\mathcal{O}((n^{1/3})^{3-\varepsilon'}) = \mathcal{O}(n^{1-\varepsilon'/3})$. Hence, we solve All-Pairs Negative Triangle in $\mathcal{O}(n^{3-\varepsilon'/3})$ time. □

4 Radius

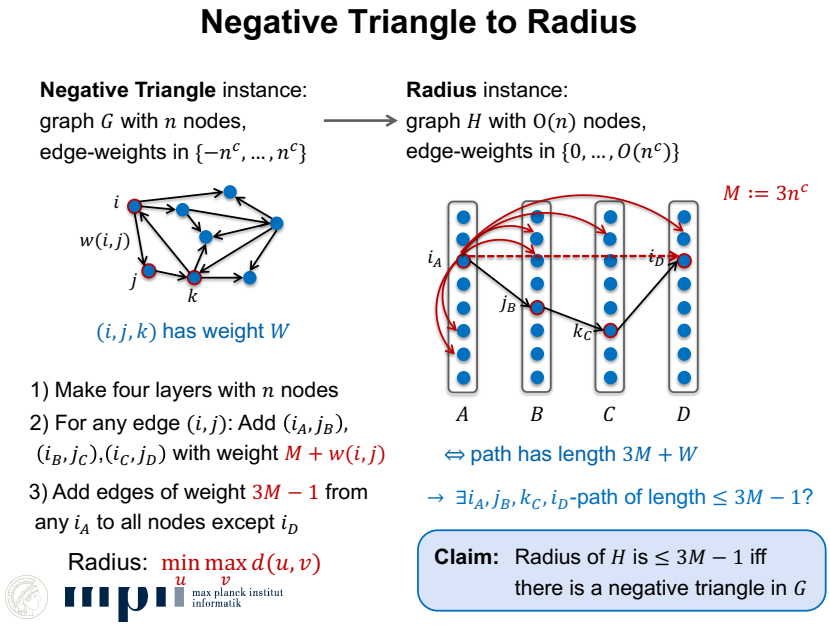
Usually, when we want to show that a problem is subcubic equivalent to APSP, it is convenient to reduce Negative Triangle (which is very simple and requires only one bit of output) to the problem, and to reduce the problem to APSP (which has a lot of expressive power).

Theorem 6. *If APSP is in $T(n)$ time, then Radius is in $\mathcal{O}(T(n))$ time.*

Proof. Trivial, just use the definition. □

Theorem 7. *If Radius is in $T(n)$ time, then Negative Triangle is in $\mathcal{O}(T(n))$ time.*

Proof. One-slide proof borrowed from Karl Bringmann's presentation available at `https://conferences.mpi-inf.mpg.de/adfocs-18/karl/2-APSP.pdf`:



□

5 Summary

We saw the following reductions: