CS 388R: Randomized Algorithms Fall 2015

Lecture 14 — October 26, 2015

Prof. Eric Price *Scribe: An T. Nguyen, Adarsh Prasad*

1 Overview

In this lecture, we study the problem of finding shortest paths. Let $\mathbb{G}$ be a graph of m edges and n vertices. First let's look at some algorithms.

Algorithm	Sources	Negative Weight	Time
Dijkstra	Single	No	$O(m + n \log n)$
Floyd-Warshall	All Pairs	Yes	$O(n^3)$
Bellman-Ford	Single	Yes	$O(mn)$

Table 1: Comparison of some shortest path algorithms.

The Floyd-Warshall algorithm is very simple:

Data: Distance matrix D
Result: Shortest path matrix D
for k *in* $[n]$ **do**
  for i *in* $[n]$ **do**
    for j *in* $[n]$ **do**
      $D_{ij} = \min(D_{ij}, D_{ik} + D_{kj})$
    end
  end
end

Algorithm 1: Floyd-Warshall algorithm

2 Faster algorithm using matrix multiplication

We can have faster algorithms for all pair shortest paths in $O(MM(n))$ time for unweighted and undirected graph, where $MM(n)$ is the time to multiply two $n \times n$ matrices. Some history of the matrix multiplication problem:

- Naive: $O(n^3)$.
- Strassen '69: $O(n^{2.8074})$.
- Coppersmith & Winograd '89: $O(n^{2.375477})$.
- Strothers '10: $O(n^{2.374})$.
- Vassilevska-Williams '11: $O(n^{2.372873})$.

- Note that the lower bound for MM is still an open problem.($O(n^2)$??)

The notation n^ω is often used in papers since ω is being continuously improved.In this lecture, MM(n) is referred to as n^ω.

First we observe the similarity between matrix multiplication and FloydWarshal. In fact, matrix multiplication is FloydWarshal where $(\min, +)$ is replaced by $(+, \times)$.

Let A be the adjacency matrix, we have that A^2_{ij} is the number of length 2 path from i to j. More generally, A^l_{ij} is the number of length l path from i to j. If we add the identity matrix to A, which is equivalent to adding all self-loop in the graph, we have that A^l_{ij} number of paths of length $\leq l$.

A naive way of using matrix multiplication to compute shortest paths is to compute: $A, A^2, ..., A^n$ then let $D_{ij} = \min_l s.t. A^l_{ij} > 0$. Two problems are $O(n^{\omega+1})$ time and big number (which can be easily solve by just storing $A^l > 0$). Suppose we want a 2-approximation to D_{ij}, which is X_{ij} such that $D_{ij} \in [\frac{X_{ij}}{2}, X_{ij}]$. We can compute $A, A^2, A^4...$ in $O(MM(n)\log n)$ by repeated squaring.

Now let D' be the distance on A^2, that is the graph with all length 2 paths added as new edges. D' can be computed recursively. Our goal is to find D from D' and A in $O(MM(n))$ time. There are two cases:

- If D_{ij} is even then $D'_{ij} = D_{ij}/2$.
- If D_{ij} is odd then $D'_{ij} = (D_{ij}+1)/2$.

So all we need to do is to find $D \mod 2$ (from D' and A). Consider again two cases:

- If D_{ij} is even then $\forall u \in N(i), D'_{uj} \in \{D'_{ij}, D'_{ij}+1\}$.
- If D_{ij} is odd then $\forall u \in N(i), D'_{uj} \in \{D'_{ij}, D'_{ij}-1\}$ and $\exists u \in N(i)$ s.t. $D'_{uj} = D'_{ij} - 1$.

That is because if the distance from i to j in A is even $(2l)$ then for a neighbor u of i the distance from u to j can only be $2l-1, 2l$ or $2l+1$. In A^2, the distance from i to j is l and from u to j is l or $l+1$ (if $D_{uj} = 2l-1$ in A then it still takes l steps from u to j in A^2). By a similar argument, we have the case for D_{ij} odd. By summing over the neighbors, we have:

- If D_{ij} is even then $\sum_{u\in N(i)} D'_{uj} \geq D'_{ij}|N(i)|$.
- If D_{ij} is odd then $\sum_{u\in N(i)} D'_{uj} < D'_{ij}|N(i)|$.

Also, these sums can be expressed as matrix multiplication:

$$\sum_{u\in N(i)} D'_{uj} = \sum_{u\in[n]} A_{iu}D'_{uj} \tag{1}$$
$$= (AD')_{ij} \tag{2}$$

So we compare AD' to $D'|N(i)|$ to get $D_{ij} \mod 2$ and set $D = 2D' - (D \mod 2)$. Each round takes n^ω time and total time is $O(n^\omega \log n)$.

3 Identifying shortest paths

Now, given D and A, we want to give an efficient algorithm for finding the shortest paths. For this lecture, we look at the case of a tripartite graph(Figure 1), with edges going from left to right. Let A,B be the adjacency matrix for $\{V_1, V_2\}$ and $\{V_2, V_3\}$ respectively, then we're interested in finding $P_{ij} = k$ such that $A_{ik} \cap B_{kj} = 1$ in $O(n^\omega)$ time.

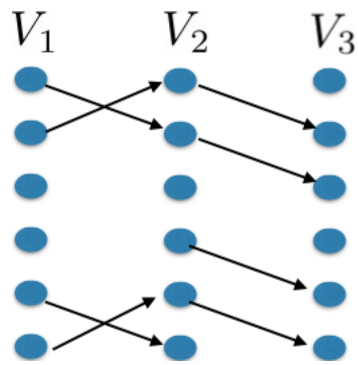

Figure 1: Tripartite Graph, $|V_i| = n \ \forall i$

Easy Case. Suppose there exists exactly one k^* such that $A_{ik^*} \cap B_{k^*j} = 1$, then:

- Define A' such that $A'_{ij} = A_{ij}.j$
- Then $(A'B)_{ij} = \sum\limits_k kA_{ik}B_{kj} = k^*$.
- So, we can identify the witness k^* for a path in $O(n^\omega)$ time.

Medium Case. Suppose there exist r witnesses $\{k_1, \ldots, k_r\}$ such that $A_{ik_d} \cap B_{k_dj} = 1$ for all $d \in [r]$, then:

- Define A' such that $A'_{ij} = A_{ij}.j.\delta_j$, where δ_j is a Bernoulli r.v. such that $P[\delta_j = 1] = 1/r$.
- Now, if exactly one of the r $\delta_{k_t} = 1$, then $(A'B)_{ij} = k_t$, so, we would have identified a witness.
- Now, $P[\sum_t \delta_{k_t} = 1] = r.\frac{1}{r}.(1 - \frac{1}{r})^{r-1} \approx 1/e > 1/4$.
- So, repeat $O(\log n)$ times and each time check result $A_{ik_t} \cap B_{k_tj} = 1$. Hence, the total runtime is $O(n^\omega \log(n))$.

Hard Case. If we don't know the number of witnesses r, then:

- **Naive Strategy.** Run medium-case strategy for all $r = 1, \ldots, n$. However, then runtime is $O(n^\omega. \log(n).n$.
- Run medium-case strategy for all $r = 1, 2, 4, \ldots, n$. For this stragy, runtime is $O(n^\omega. \log(n). \log(n)$.

- To analyse it's correctness: Suppose r :true number of witnesses, and let r' be our guess such that $r'/2 \leq r \leq r'$:

$$\begin{aligned} P[\text{exactly one of true witness } \delta_{k_t} = 1] &= r.\frac{1}{r'}.(1 - \frac{1}{r'})^{r-1} \\ &\geq 1/2e \end{aligned}$$

- Since, we have a constant probability of success, running for $O(\log(n)$ iterations at r' such that $r'/2 \leq r \leq r'$, suffices.

- Hence, an overall runtime of $O(n^{\omega}. \log(n). \log(n)$ is sufficient for find a witness.

References

[1] R. Bellman. On a routing problem. Technical report, DTIC Document, 1956.

[2] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 1–6. ACM, 1987.

[3] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[4] R. W. Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.

[5] L. R. Ford Jr. Network flow theory. Technical report, DTIC Document, 1956.

[6] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[7] A. J. Stothers. On the complexity of matrix multiplication. 2010.

[8] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4):354–356, 1969.

[9] V. V. Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 887–898. ACM, 2012.