(Deng)

Hi, this is team DD, my name is Deng Kaisheng and this is my teammate Ding Jiayi, (丁哥打招呼)

it’s a great honor to introduce our miniproject to you.

Our project focuses on the paper “External-Memory Exact and Approximate All-Pairs Shortest-Paths in Undirected Graphs”. The paper mainly presented the following 3 algorithms.

1. Present a cache-oblivious exact algorithm for APSP in unweighted undirected graphs

2. Present a cache-aware approximate algorithm for APSP in undirected graphs.

3. Present a cache-aware exact algorithm for APSP in weighted undirected graphs

And for our miniproject, we only discuss the first one, that is…

Before we move a step forward into the algorithm, let’s introduce some background knowledge.

… (background knowledge)

And for today’s video, we will dive deep into the first section of this paper. A cache-oblivious exact algorithm for APSP in unweighted undirected graphs

Before we move a step forward into the new algorithm, we need look back on the MR-BFS algorithm.

(Deng)

Background Knowledge

(Ding)

MR-BFS

BLABLABLA about MR-BFS with graph example.

BLABLABLA with I/O complexity analysis.

After running the MR-BFS, we will get for an unweighted undirected graph G = (V [G],E[G]), the shortest path from a source node s to every node v where s,v ∈ V [G], and we denote this shortest distant by d(u, v).

Before we move further:

We need to clarify 2 observations:

1. d(u, v) = d(v, u) in an undirected graph.
2. For any three vertices u, v and w in G, d(u,w) − d(u, v) ≤ d(v,w) ≤ d(u,w) + d(u, v). Triangle Inequality

(Graph example)

With above Observation, If we sort the adjacency lists we just got in non-decreasing order by d(u, ·), and denote the portion of this sorted list which contains adjacency lists of vertices w with d(u,w) = j by A(j).

Then the adjacency list of any vertex w with d(v,w) = i must reside in some A(j) where i − d(u, v) ≤ j ≤ i + d(u, v).

With this observation, we have the next BFS algorithm.

BLABLABLA about I-BFS with graph example.

BLABLABLA with I/O complexity analysis.

Before we put things together to get our final algorithms, we still have a little step to finish.

Observation 4:

If ET is an Euler Tour of a spanning tree of an unweighted undirected graph G:

1. the number of edges between any two vertices x and y on ET is an upper bound on d(x, y) in G
2. ET has O(V ) edges
3. each vertex of V [G] appears at least once in ET

Ensures that BFS will be performed from each v ∈ V [G].

With this clarified, it’s time to move to the final algorithm.

Firstly, find all vertices of G, and choose a source node to perform MR-BFS. And then for rest vertices, we can use Incremental-BFS to compute the shortest path to all vertices.

BLABLABLA about AP-BFS with graph example.

BLABLABLA with I/O complexity analysis.

Thanks all we are going to share today, thanks for watching!

(Deng) 最后一段稿子

So by now we have introduced the major components of the algorithm, but there is still something you should pay attention to.

Euler Tour of spanning tree visits each edge once and exactly once, so we can ensure that each node has been visited, also, it visits each edge only once, so it guarantees we only do the calculation we need, no repetitions!

So now we can put all things together and we made it, the cache-oblivious APSP algorithm for unweighted undirected graphs. Noted that the first step only need to use DFS once.

By now we have finished the introduction of the algorithm, but maybe you are still confused, hold on and I’ll answer them for you.

Take the example below to illustrate why we use Euler tour of spanning tree

Imagine we are applying our algorithm on this graph, and in current iteration, u equals 0, v equals 3