Randomized linear-time algorithm for MST

Analysis and Design of Algorithms, CS302

Dr. Apurva Mudgal

Group members -

Monu Kumar, Bajrang Lal, Avnish Kumar

Ayush Singh, Anivesh Singh Gurjer

Objective

- Introduction to Minimum Spanning Tree
- Boruvka's algorithm.
- Heavy Edge and MST verification.
- Random Sampling for a graph.
- Randomized algorithm for MST.
- Analysis for Expected Time Complexity.

Minimum Spanning Tree (MST)

- G(V,E) - Connected Graph with real valued edge weights.
- A spanning tree in G is an acyclic subgraph of G that includes every vertex of G and is connected, every spanning tree has exactly n-1 edges. The weight of a tree is the sum of the weights of its edges.
- A Minimum spanning tree is a spanning tree of minimum weight.
- The minimum spanning tree problem (MSTP) is – Given G, find and MST of G.

Minimum Spanning Forest

- Let G(V,E) be a disconnected graph.
- A spanning forest F is a forest whose trees are spanning tree for the connected components of the graph G.
- A minimum spanning forest F = (V, Eo) is a subgraph of G(V,E) such that the sum of all the weights w(e) for all the edges e in Eo is minimal.

Algorithms for the MSTP

The following are deterministic algorithms based on a greedy strategy to output the MST of a graph G.

1. Kruskal's algorithm - O(E*log(V))
2. Prim's Algorithm - O((V+E)log(V))
3. Boruvka's Algorithm.

Boruvka’s Algorithm

- There is another greedy strategy for MST called Boruvka’s algorithm, which also runs in O(m log n) time. Later we will show that using randomization in conjunction with the algorithm leads to a linear-time algorithm. Boruvka’s algorithm is based on the following lemma.

Basic Algorithm

1) Input is a connected, weighted and un-directed graph.
2) Initialize all vertices as individual components (or sets).
3) Initialize MST as empty.
4) While there are more than one components, do following for each component.
5) a) Find the closest weight edge that connects this component to any other component.
   b) Add this closest edge to MST if not already added.
6) Return MST.

Boruvka Phase

- Mark the edges to be contracted in G and store them in set T.
- Determine the connected components formed by the marked edges.
- Replace each connected component by a single vertex.
- Finally, eliminate the self-loops and multiple edges created by these contraction.
- Return contracted graph G'.

Boruvka Phase(Contraction)

- **Lemma 1** Let v ∈ V be any vertex in G. The MST for G must contain the edge (v, w) that is the minimum weight edge incident on v.
- Proof – By contradiction.

Suppose that (v, w), the minimum weight edge incident on v is not contained in the MST T of G, then we must have another edge (v, u) in T such that vertex v can be covered in T. By adding (v, w) into T, we will form a path of cycle in T which passes v. Since (v, w) is the minimum weight edge incident on v, we can remove (v, u) and thus keep (v, w) in T0 , resulting in that the weights of T 0 is less than that of T. This reaches the contradiction with that T is the MST of G. Therefore, the MST for G must contain the edge (v, w).

Continue..

- **Lemma 2 - The set of edges marked for contraction during a Boruvka phase includes a forest in G.**

- From the steps listed above, we can compute the running time of implementing such a Boruvka phase. Step 5 takes O(m+n) time; in step 5a and step 5b, we may use a Depth-First Search or the concept of DSU to find the connected components and create a new vertex that correspond to each connected component, and associate each vertex with that new vertex, all these take O(m + n) time; finally, step 4 takes O(m) time. Thus, the whole process of a Boruvka phase needs O(m + n) time.

Continue..

- **Lemma 2.1 -** The set of edges marked for contraction during a Boruvka phase induces a forest in G.
- By Lemma 2, each of the edges marked for contraction must be one of the edges in the MST of G.
- During each Boruvka phase, the edges we marked is a subgraph of the MST of G. Therefore, these edges might not be connected, but must be acyclic.
- This makes them form a forest of G. ✷ We claim that the graph G0 obtained from the Boruvka Phase has at most n/2 vertices. This is because that each contracted edge can be the minimum incident edge on at most two vertices.
- The number of marked edges is thus at least n/2 . Since each vertex chooses exactly one edge to mark, it is easy to verify that each marked edge must eliminate a distinct vertex. The number of edges in G0 is no more than m since no new edges are created during this process. By Lemma 3.1, we know that each of the contracted edges must belong to the MST of G. In fact, the forest induced by the edges marked for contracting is a subgraph of the MST.

Continue..

- Lemma: 3 Let G’ be the graph obtained from G after a Boruvka phase. The MST of G is the union of the edges marked for contraction during this phase with the edges in the MST of G’.
- we know that during each Boruvka phase, the edges we marked is a forest of G, and these edges belong to the MST of G. Each marked edge eliminate a distinct vertex of G, thus the uncovered vertices are still remained in G’. The union of the MST of G’ with these marked edges will cover all the vertices with no cycle. Furthermore, since the MST of G’ will also be induced by Boruvka phases, the MST of G’ is also a subgraph of the MST of G. Therefore, the union of the edges marked for contraction during this phase with the edges in the MST of G’ is the MST of G.

Pseudo Code

Input G(V, E).

- Output - T, the MST.

```
1. T <-- Empty Graph.
2. for each v in V do
3.     Let e be the minimum weight edge in G that is incident on v.
4.     T <-- T + {e}
5. end for
6. G' <-- G with all edges in T contracted.
7. T' <-- recursively compute the minimum spanning tree of G'.
8. return T + T'
```

Analysis

- Boruvka’s algorithm reduces the MST problem in an n-vertex graph with m edges to the MST problem in an (n/2)-vertex graph with at most m edges. The time required for the reduction is only O(m + n). The worst case running time is O(m log n).

Analysis

Time complexity for the boruvka phase - $O(n+m)$

Proof -

Marking the edges - $O(n+m)$.

To find connected components and assign them a vertex we can use DFS - $O(n+m)$.

Handling loops and the multi-edges - $O(m)$.

Time complexity for the algorithm -

$T(n, m) <= T(n/2, m) + O(n+m)$.

$T(n, m) = O(mlogn)$

The worst case time complexity of the algorithm is $O(mlogn)$.

Concept of Heavy weight

- Heavy weight: Let F be any forest in graph G and consider any pair of vertices (u, v) ∈ V . Let wF (u, v) denote the maximum weight of any edge along the unique path in F from u to v. If there is no path exists, then wF (u, v) = ∞. Note that if an edge (u, v) exists in G, the normal weight of this edge is denoted as w(u, v).

Heavy Edges w.r.t a forest

- Let F be any forest in graph G. If an edge (u,v) is F-heavy, then it does not lie in the MST of G.
- The converse is not true.
- Proof: By the definition of F-heavy edges, we know that F must contain a path from u to v using the edges of weight smaller than (u,v).
- Suppose (u,v) belongs to the MST of G, then by replacing (u,v) with the existed path in F from u to v into the MST, we can obtain another MST of G with the smaller weights.
- Therefore, there is contradiction thus (u,v) cannot lie in the MST of G. However, F may contain isolated vertex, say u. There is no path existed in F from any other vertex to u. By the definition, $w_F(u, v_i) = \infty$, $v_i \in V_u$ and (u, v_i) is F-light. It is easy to see that for a certain edge not in the MST of G, it might be F-light. Thus, the converse is not true.

Random Sampling for MST’s

- In order to reduce the number of edges in the graph, we will use the fact that a random subgraph of G has a ”similar” minimum spanning tree. To be precise, consider a (random) graph G(p) obtained by independently including each edge of G in G(p) with probability p. The graph G(p) has n vertices and expected number of edges mp.
- Note that there is no guarantee that G(p) is connected. Let F be the minimum spanning tree for G(p). We expect very few edges in G to be F-light such that F would be a good approximation to the MST of G. This expectation is explained in details in the lemma presented below

Negative Binomial Variables

- Let X1, X2, ..., Xn be independent random variables whose common distribution is the geometric distribution with parameter p,. The random variable X = X1 + X2 + ... + Xn denotes the number of coin flips needed to obtain n HEADS.
- The random variable X has the negative binomial distribution with parameters n and p. The density function for this distribution is defined only for x = n, n + 1, n + 2...:

$$\mathbf{Pr}[X = x] = \binom{x-1}{n-1} p^n q^{x-n}$$

$$\textit{The Characteristics are: } E[X] = \frac{n}{p},\ var[X] = \frac{nq}{p^2},$$

The Linear-Time MST Algorithm

- The randomized linear time MST algorithm interleaves Boruvka phases that reduce the number of vertices with random sampling phases that reduce the number of edges.
- After a random sampling phase, the minimum spanning forest F of the sampled edges is computed using recursion, and the verification algorithm is used to eliminate all but the F-light edges.
- Then, the MST with respect to the residual F-light edges is computed using another recursive invocation of the algorithm. This is summarized in the following Algorithm MST.

The Linear-Time MST Algorithm

- Linear time algorithm :
- Input: Weighted, undirected graph G(V, E) with n vertices and m edges.
- Output: Minimum spanning forest F for G.
- 1: Using three applications of Boruvka phases interleaved with simplication of the contracted graph, compute a graph G1 with at most n/8 vertices and let C be the set of edges contracted during the three phases.
- If G is empty then exit and return F = C.
- 2: Let G2 = G1(p) be a randomly sampled subgraph of G1, where p = 1/2 .
- 3: Recursively applying Algorithm MST , compute the minimum spanning forest F2 of the graph G2.
- 4: Using a linear-time verification algorithm, identify the F2-heavy edges in G1 and deleted them to obtain a graph G3.
- 5: Recursively applying Algorithm MST , compute the minimum spanning forest F3 of the graph G3.
- 6: Return forest F = C U F3.

Running Time:

- The expected running time of Algorithm MST is O(n + m).
- Proof: Let T(n, m) denote the expected running time of Algorithm MST for a graph G(V, E) with n vertices and m edges. Consider the cost of various steps in this algorithm for such input.
- Continue:

Running Time:

- 1. At step 1, the three invocations of Boruvka algorithm, which runs in O(n + m) time, run in deterministic time O(n + m).
- After this step, a graph G1 with at most n/8 vertices and m edges is produced.
- 2. At step 2, the algorithm performs a random sampling to produce the graph G2 = G1(1/2) with n/8 vertices and an expected number of edges equal to m/2 .
- This step also takes O(m + n) time.
- 3. Finding the minimum spanning forest of G2 in step 3 has the expected running time of T(n/8 , m/2).
- 4. The linear-time algorithm verification in step 4 runs in time O(n + m) and produces a graph G3 with at most n/8 vertices and an expected number of edges at most n/4
- 5.Finding the minimum spanning forest of G3 in step 3 has the expected running time of T(n/8 , n/4).
- 6. Finally, O(n) time is needed for step 6.

Proof:

- Recurrence Relation: $$T(n,m) \leq T(\frac{n}{8},\frac{m}{2}) + T(\frac{n}{8},\frac{n}{4}) + c(n+m)$$

- Using Master theorem we can proof that the running time is O(n+m);