Overview Of Minimum Spanning Tree

Monday, February 7, 2022 7:20 PM

You might wonder: what is a spanning tree? A **spanning tree** is a connected subgraph in an undirected graph where **all vertices** are connected with the **minimum number** of edges. In Figure 9, all pink edges [(A, B), (A, C), (A, D), (A, E)] form a tree, which is a spanning tree of this undirected graph. Note that [(A, E), (A, B), (B, C), (C, D)] is also a spanning tree of the undirected graph. Thus, an "undirected graph" can have multiple spanning trees.

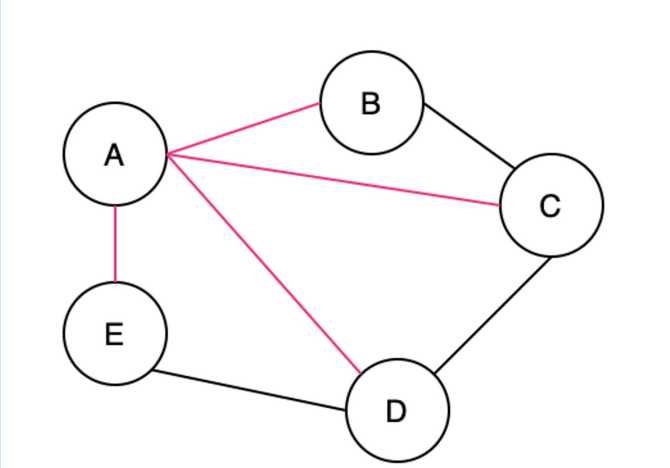

Figure 9. Spanning tree

After learning what a spanning tree is, you might have another question: what is a minimum spanning tree? A minimum spanning tree is a spanning tree with the minimum possible total edge weight in a "weighted undirected graph". In Figure 10, a spanning tree formed by green edges [(A, E), (A, B), (B, C), (C, D)] is one of the minimum spanning trees in this weighted undirected graph. Actually, [(A, E), (E, D), (A, B), (B, C)] forms another minimum spanning tree of the weighted undirected graph. Thus, a "weighted undirected graph" can have multiple minimum spanning trees.

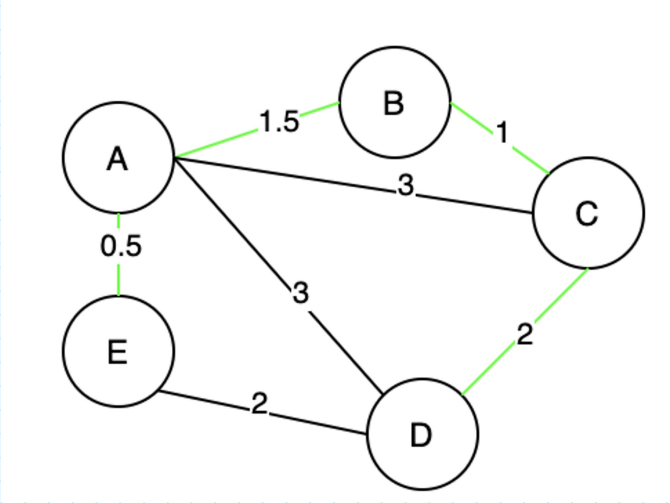

Figure 10. Minimum spanning tree

In this chapter, we will learn about the "cut property and two algorithms for constructing a "minimum spanning tree":

- *Kruskal's Algorithm*
- *Prim's algorithm*

Cut Property

Tuesday, February 8, 2022 2:13 PM

What is a "cut"? Although many theorems are named after people's names, "cut" is not one of them. To understand the "cut property", we need to understand two basic concepts.

- First, in Graph theory, a "cut" is a partition of vertices in a "graph" into two disjoint subsets. Figure 11 illustrates a "cut", where (B, A, E) forms one subset, and (C, D) forms the other subset.
- Second, a crossing edge is an edge that connects a vertex in one set with a vertex in the other set. In Figure 11, (B, C), (A, C), (A, D), (E, D) are all "crossing edges".

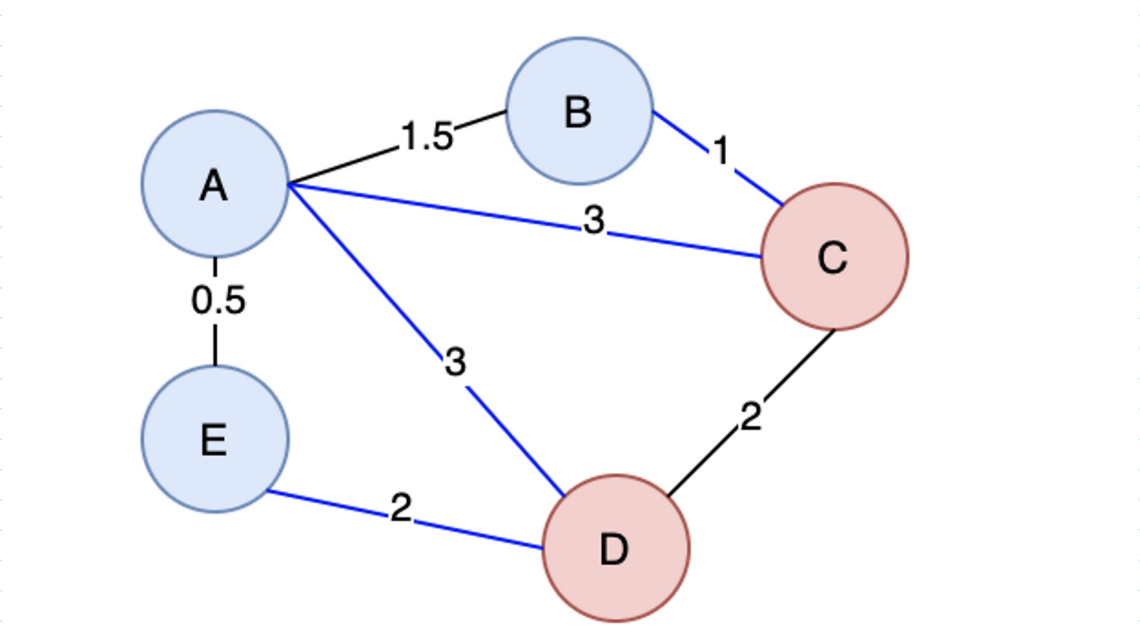

Figure 11. Graph with a cut

After knowing the basics of a graph cut, let's delve into the "cut property". The cut property provides theoretical support for Kruskal's algorithm and Prim's algorithm. So, what is the "cut property"? According to Wikipedia, the "cut property" refers to:

For any cut C of the graph, if the weight of an edge E in the cut-set of C is strictly smaller than the weights of all other edges of the cut-set of C, then this edge belongs to all MSTs of the graph.

Kruskal's Algo

Tuesday, February 8, 2022 2:16 PM

- "Kruskal's algorithm" is an algorithm to construct a "minimum spanning tree" of a "weighted undirected graph".

- This is basically forming min spanning tree by edges.
- So we need input in the form of edge
  - A - B with dist say 3.
  - {{A,B},3}
  - So, it can be like vectors of pair<pii, int> then we can sort it all by edges distance or put it in priority_queue to get min dist first.

- So, to prevent forming cycle we use union-find:
  - If A and B belong to same component we skip that edge.
  - Else add that edge to ans and increase cnt++;
  - If we reach cnt == n - 1, we done, as min edges we need is n-1 to get a connected components of n vertices.

Complexity Analysis

- Time Complexity: $O(E \log E)$. Here, E represents the number of edges.
  - At first, we need to sort all the edges of the graph in ascending order. Sorting will take $O(E \log E)$ time.
  - Next, we can start building our minimum spanning tree by selecting which edges should be included. For each edge, we will look at whether both of the vertices of the edge belong to the same connected component; which is an $O(\alpha(V))$ operation, where α refers to the Inverse Ackermann function. In the worst case, the tree will not be complete until we reach the very last edge (the edge with the largest weight), so this process will take $O(E\alpha(V))$ time.
  - Therefore, in total, the time complexity is $O(E \log E + E\alpha(V)) = O(E \log E)$.

- Space Complexity: $O(V)$. V represents the number of vertices. Keeping track of the root of every vertex in the union-find data structure requires $O(V)$ space. However, depending on the sorting algorithm used, different amounts of auxiliary space will be required to sort the list of edges in place. For instance, Timsort (used by default in python) requires $O(E)$ space in the worst-case scenario, while Java uses a variant of quicksort whose space complexity is $O(\log E)$.

Prim's Algo

Tuesday, February 8, 2022 2:42 PM

- After learning about Kruskal's Algorithm, let's look at another algorithm, "Prim's algorithm", that can be used to construct a "minimum spanning tree" of a "weighted undirected graph".

- This is basically forming min spanning tree by vertices.
- We choose a random vertex at first, add it to visited.
- We see all the edges connected by this vertex, if it is connected to a vertex that has not been visited yet, we add vertex with its distance to the heap {dist, vertex};
- We keep on following this until we get n vertices (cnt < n);
- To improve the complexity by not adding unwanted pairs in heap we can maintain a key vector. (See code).

The difference between the "Kruskal's algorithm" and the "Prim's algorithm"

- "Kruskal's algorithm" expands the "minimum spanning tree" by adding edges. Whereas "Prim's algorithm" expands the "minimum spanning tree" by adding vertices.

Complexity Analysis

V represents the number of vertices, and E represents the number of edges.

- Time Complexity: **O(ElogV)** for Binary heap, and **O(E+VlogV)** for Fibonacci heap.
    - For a Binary heap:
        - We need **O(V + E)** time to traverse all the vertices of the graph, and we store in the heap all the vertices that are not yet included in our minimum spanning tree.
        - Extracting minimum element and key decreasing operations cost $O(\log V) O(\log V)$ time.

        - Therefore, the overall time complexity is $O(V + E)\ O(\log V) = O(E \log V)$.
    - For a Fibonacci heap:
        - Extracting minimum element will take $O(\log V)$ time while key decreasing operation will take amortized $O(1)$ time, therefore, the total time complexity would be $O(E+V \log V)$.
- Space Complexity: $O(V)$. We need to store V vertices in our data structure.