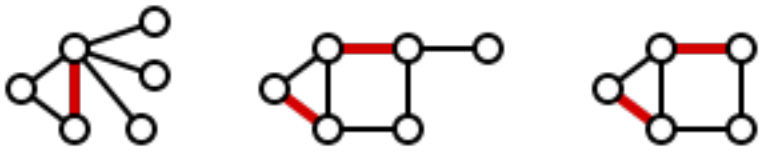
A Simple Approximation Algorithm for Weighted Matching Problem

1 Introduction:

In a graph G = (V, E), a matching M in G is a set of pairwise non-adjacent edges; that is, no two edges share a common vertex. A vertex is said to be matched if it is an endpoint of one of the edges in the matching. Otherwise, the vertex is unmatched.



(Source: Wikipedia [1])

A maximum matching is a matching which has the largest possible number of edges. There may be many maximum matchings. Also, every maximum matching is maximal, but not every maximal matching is a maximum matching.

Here, we are interested in a weighted matching problem in which we have graph G (V, E) where each edge has some weight. The weighted matching problem is to find a matching M in G that has maximum weight. For this problem, the fasted algorithm is given by Gabow [2] and has a running time of $O\left(|V||E| + |V|^2 \log\left(V\right)\right)$. As explained in the paper [3], for many practical problems which consist of very large graphs such running time can be too costly. Therefore, an approximation algorithm for a weighted matching problem which ideally has linear running time is required. Also, implementing polynomial timing exact algorithm for the weighted matching problem is a difficult task.

Greedy Matching $(\boldsymbol{G = (V, E), w : E \to \mathbb{R}_+})$

```
1  M := ∅
2  while E ≠ ∅ do begin
3      let e be the heaviest edge in E
4      add e to M
5      remove e and all edges incident to e from E
6  end
```

Figure 1: The greedy algorithm for finding maximum weight matchings.

The greedy algorithm shown in figure 1 is an approximation algorithm for weighted matching problem with performance ratio 1/2. In [4] Peris used the concept of locally heaviest edge which improves the simple greedy approach which helped him to obtain an approximation algorithm for weighted matching problem which has a performance ratio of 1/2 and running time of O(|E|). Here, the same approach is used to create a simple approximation algorithm for weighted matching problem.

2 The Path Growing Algorithm:

The Path Growing Algorithm consist of a weighted graph G (V, E) in which the algorithm starts with an arbitrary vertex and tries to extend the path in one direction where the edge has heaviest weight. When the vertex x has neighbor, the algorithm

PathGrowingAlgorithm $(G = (V, E), w : E \to \mathbb{R}_+)$

```
1   M1 := ∅, M2 := ∅, i := 1
2   while E ≠ ∅ do begin
3       choose x ∈ V of degree at least 1 arbitrarily
4       while x has a neighbor do begin
5           let {x, y} be the heaviest edge incident to x
6           add {x, y} to Mi
7           i := 3 − i
8           remove x from G
9           x := y
10      end
11  end
12  return max(w(M1), w(M2))
```

checks for the heaviest incident edge and accordingly moves to the vertex y of the heaviest edge. Before moving to the y vertex of heaviest edge, it stores the heaviest edge in M1 and increments the 'i' so that next heaviest edge is stored in M2 and deletes the remaining edges on the x vertex of heaviest edge. After this, the vertex y is assigned to x and the loop goes on till all the vertices are covered. Out of these two matching M1 and M2, one with maximum edge weight is returned. This algorithm has a performance ratio of 1/2.

3 The running time is linear

Theorem 1: The Path Growing Algorithm has running time O(|E|).

Proof:

- All the vertices are processed at most one in a while loop.
- After finding a heaviest edge on a vertex the edges are deleted.
- The path is started from the end vertex of a heaviest edge so no vertex of degree zero will be processed.
- The number of vertices of the degree at least 1 in G is O(|E|).

Hence, the total running time of this algorithm is bounded by the sum of degree of vertices in V which is O(|E|).

4 The performance ratio is 1/2

Theorem 2: The Path Growing Algorithm has performance ratio of 1/2.

Proof:

Assigning each edge of a graph to some vertex in following way.

- Whenever a vertex is removed in line 8, all the edges that are currently assigned to that vertex x are assigned to x. In this way, each edge of G is assigned to exactly one vertex of G.
- For maximum weight matching M, incident edges are not allowed. So, the heaviest edge is currently incident on the vertex is chosen and it is added to M1 and M2 alternatively. Hence, the weight of M1 $\cup$ M2 is at least as large as the weight of M.

$$\max\left(w(M1), w(M2)\right) \geq \frac{1}{2}\, w(M1 \cup \mathrm{M2}\,) \geq \frac{1}{2}\, w(M)$$

Hence, Proved.

5 Implementation in Java

A graph is a made up of Edges and Vertices. The code uses Edge class to represent the edges between the nodes in a graph.

The Edge class has following functions:

- setWeight () to set the edge weight between vertices.
- getWeight () to get the edge weight.
- Constructor () which takes in edge weight, vertex 1 and vertex 2 as parameter.
- getVertexOne () and getVertexTwo () to get the respective vertex.
- setVertexOne () and setVertexTwo () to set the respective vertex.

The Graph class has following functions:

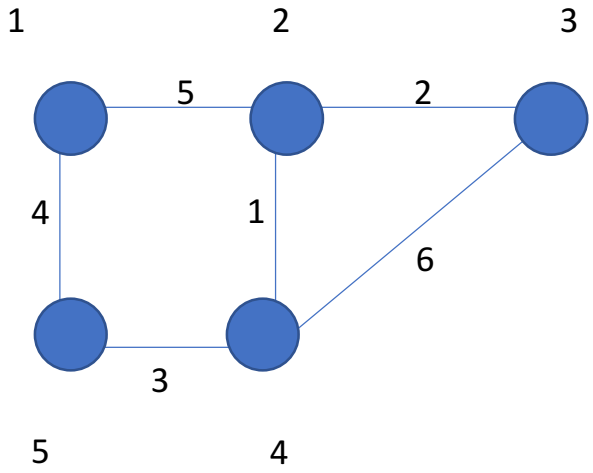
- getEdge () and setEdges () to get and set the edges respectively.
- getVertices and setVertices to get and set the vertices respectively.
- Graph () constructor which takes in parameter the number of vertices and an array list of edges.
- getHeaviestEdge () to get the edge incident to the vertex which has the highest edge weight.
- getVertexWeight () to get degree of vertex.
- weight () calculates the weight of M1 and M2.
- remove () deletes the vertex and incident edges.

The Main class:

- pathGrowingAlgorithm () to implement the algorithm.

5.1 Working:

The following graph is used for the implementation of the algorithm.



This graph is added in the form of an array list of edge in the graph class with number of vertices.

The pathGrowingAlgorithm () method selects a random vertex from the graph.

1) Suppose the random vertex is 1 then

the path followed with a heaviest edge is 1 ➔ 2 ➔ 3 ➔ 4 ➔ 5

	Edges	Weight
M1 = 5 + 6	2	11
M2 = 2 + 3	2	5

Max is 11.

```
Random vertex 1
second vertex 2
second vertex 3
second vertex 4
second vertex 5
 M1 has total of weight of 11 with number of edges 2
 M2 has total of weight of 5 with number of edges 2
The maximum weighted matching using the Path growing algorithm is 11

Process finished with exit code 0
```

2)Suppose the random vertex is 5 then

the path followed with heaviest edge is 5 ➔ 1 ➔ 2 ➔ 3 ➔ 4

	Edges	Weight
M1 = 4 +2	2	6
M2 = 5 + 6	2	11

Max is 11.

```
Random vertex 5
second vertex 1
second vertex 2
second vertex 3
second vertex 4
 M1 has total of weight of 6 with number of edges 2
 M2 has total of weight of 11 with number of edges 2
The maximum weighted matching using the Path growing algorithm is 11

Process finished with exit code 0
```

References:

[1] "Matching (graph theory)", *En.wikipedia.org*, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Matching_(graph theory). [Accessed: 12- Dec- 2017].

[2] H. Gabow, "A Data Structure for Nearest Common Ancestors with Linking", *ACM Transactions on Algorithms*, vol. 13, no. 4, pp. 1-28, 2017.

[3] D. Drake and S. Hougardy, "A simple approximation algorithm for the weighted matching problem", *Information Processing Letters*, vol. 85, no. 4, pp. 211-213, 2003.

[4] R. Preis, "Linear Time 1/2-Approximation Algorithm for Maximum Weighted Matching in General Graphs", 2017.

Saurabh Jaiswal