# Bottleneck Spanning Tree

Dongfeng Gu

November 20, 2015

## 1 Introduction

In this section, we will introduce the concept of Minimum Bottleneck Spanning Tree (MBST) in a(an) directed(undirected) connected graph $G = (V, E)$ and we will use an example to show that the MBST of a graph $G$ can be more than one.

### 1.1 Minimum Bottleneck Spanning Tree

Given an undirected connected graph $G = (V, E)$ and a function $w : E \to R$, let $S$ be the set of all spanning trees in graph $G$ and $B(S_i)$ is the maximum weight edge for the spanning tree $S_i$. A Minimum Bottleneck Spanning Tree (MBST) is the spanning tree $S_j$ in $S$ which the value of $B(S_j)$ is the minimum among all the $B(S)$. To sum up, a MBST in an undirected graph is a spanning tree in which the most expensive edge is as cheap as possible [7]. The maximum edge is the Bottleneck of this spanning tree.

### 1.2 Example of MBST

Let's take an example to have a better understanding on MBST.

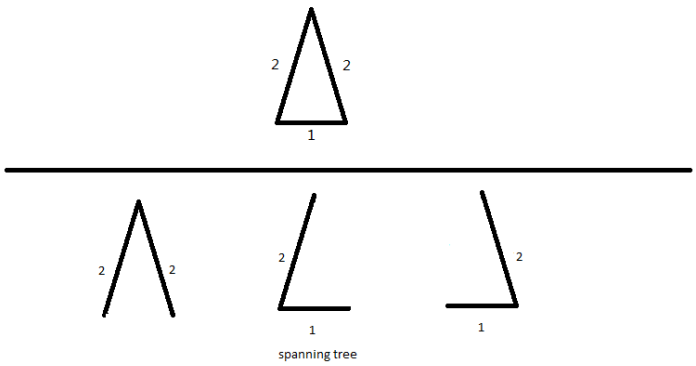


Figure 1: Spanning Tree

The bottleneck edge for all spanning trees are 2, which means there exist no other tree having its bottleneck value less than 2. Hence all three spanning trees are MBST.

### 1.3 Minimum Bottleneck Spanning Arborescence

For a directed graph, a similar problem is known as Minimum Bottleneck Spanning Arborescence (MBSA). An arborescence of $G(V, E)$ is a minimal subgraph of G which contains a directed path from a specified node "a" in G to each node of a subset $V'$ of $V - a$. Node "a" is called the root of the arborescence. An arborescence is a spanning arborescence if $V' = V - a$.

### 1.4 Example of MBSA

Here is an example of MBSA, the red lines form a spanning arborescence.

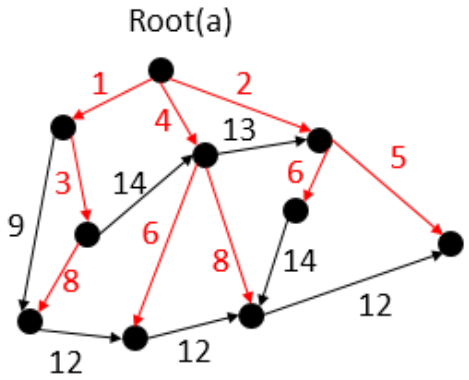


Figure 2: Minimal Bottleneck Spanning Arborescence (MBSA)

## 2 Common Problems for MBST/MBSA

In this section, we will introduce the relationship between MBST and MST and address the solution of finding MBST/MBSA in a graph $G$.

### 2.1 MST vs MBST

The well know problem for Minimum Bottleneck Spanning Tree (MBST) is related to the Minimum Spanning Tree (MST) in which the MST is necessary an MBST and opposite is not true. Therefore, any algorithm that can generate a MST can be also used for MBST.

**Proof** By contradiction. If MST is not a MBST, i.e. in a graph $G(V, E)$, $T_1$ is a MST in graph $G$ and $T_2$ is a MBST in $G$, $e_1$ is the maximum edge in $T_1$ and $e_2$ is the maximum edge in $T_2$. $W(e_1) > W(e_2)$ because $T_2$ is a MBST and it's maximum edge should be the smallest among all the possible spanning tree include the MST of graph $G$. If we add $e_1$ to $T_2$, $T_2$ will form a cycle and the cycle will have the maximum edge as $e_1$. However according to the definition of the MST,

the total weight of the MST should be the minimum among all the possible spanning tree, we can get $e_1$ cannot belong to any MST, hence the MST should be a MBNT.

### 2.2 Find MSBT in graph $G$

**Camerini's algorithm[2]** Given an undirected connected graph $G(V, E)$. Let $A$ be a subset of $E$ such that $W(e) \geq W(e')$ for all $e \in A, e' \in B = E \backslash A$. Let $F$ be a maximal Forest of $G_B$ and $\eta = N_1, N_2, ..., N_c$, where $N_i (i = 1, 2, ..., c)$ is the set of nodes of the $i - th$ component of F.

**Theorem 1** (a) If $F$ is a spanning tree of $G$, a Minimum Bottleneck Spanning Tree (MBST) of $G$ is given by any MBST of $G_B$.

(b) If $F$ is not a spanning tree of $G$, a MBST of $G$ can be obtained by adding to $F$ any MBST of $G'$, where $G'$ is the graph $G_A$ collapsed into $\eta$, i.e. $G' = (G_A)_\eta$.

**Proof** Case (a) is obvious, because all the edges in $B$ is less weight than any edges in $A$. Then the maximum weight of edges in a MBST of $G$ should be no less than the maximum weight of edges in a MBST in $G_B$. In case (b) let $S$ be a MBST of $G$, modify $S$ as many times as possible. If $e' \in B - S$ is such that the (unique) cycle in $S + e'$ contains an edge $e \in A$, then $S \leftarrow S + e' - e$. At the end, $S = F \cup S'$, where $F$ is a maximal forest of $G_B$ and $S'$ is a spanning tree of $G'$.

The following algorithm is suggested by Camerini:

**Procedure 1** MBST(G,w)

let $E$ be the set of edges of $G$;
**if** $|E| = 1$ **then**
  return $E$
**else**
  $A \leftarrow UH(E, w)$;
  $B \leftarrow E \backslash A$;
  $F \leftarrow FOREST(G_B)$;
  let $\eta = N_1, N_2, ..., N_c$, where $N_i (i = 1, 2, ..., c)$ is the set of nodes of the i-th component of $F$;
  **if** $c = 1$ **then**
    return $MBST(G_B, w)$
  **else**
    return $F \cup MBST((G_A)_\eta, w)$;
  **end if**
**end if**

In the above algorithm, the procedure $A \leftarrow UH(E, w)$ means that return a subset $A$ of $E$ such that all the edges in $A$ are less weight than the $E \backslash A$ and $|A| = \lfloor \frac{|E|}{2} \rfloor$. The procedure $Forest(G)$ returns a maximal forest of G.

**Time Complexity** According to [2], the procedure $UH(E, w)$ can be run in $O(|E|)$ steps because this problem is equivalent to the problem of finding a median of weighted set of elements[6].

The computation of $FOREST(G_B)$ can also be made in $O(|E|)$ time by performing a depth-first search of an undirected graph (Algorithm 5.2 of [1]). Therefore, the running time of $UH$, $FOREST$ can be written as $O(\frac{m}{2^i})$ at the $i$-th iteration, where m is the number of edges at the first call. To sum up, the total running time of this algorithm is:

$$O(m + \frac{m}{2} + \frac{m}{2} + \frac{m}{4} + \frac{m}{8} + ... + 1) = O(m)$$

**Example** The following example shows that how the algorithm works.

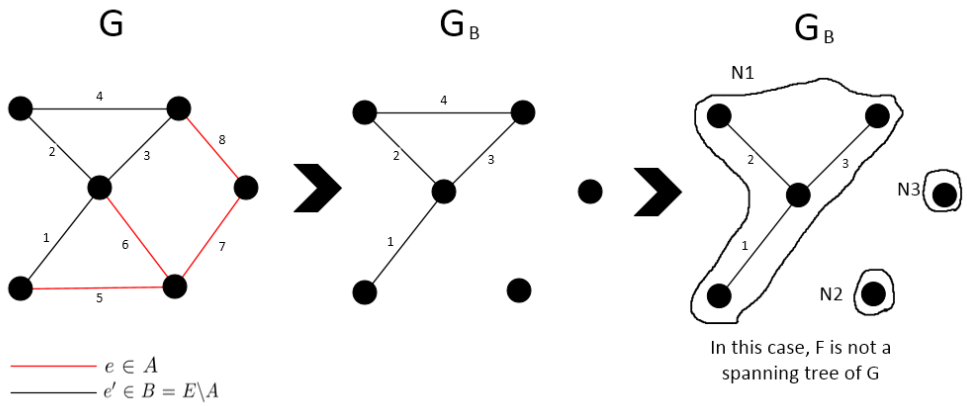


Figure 3: Procedure 1

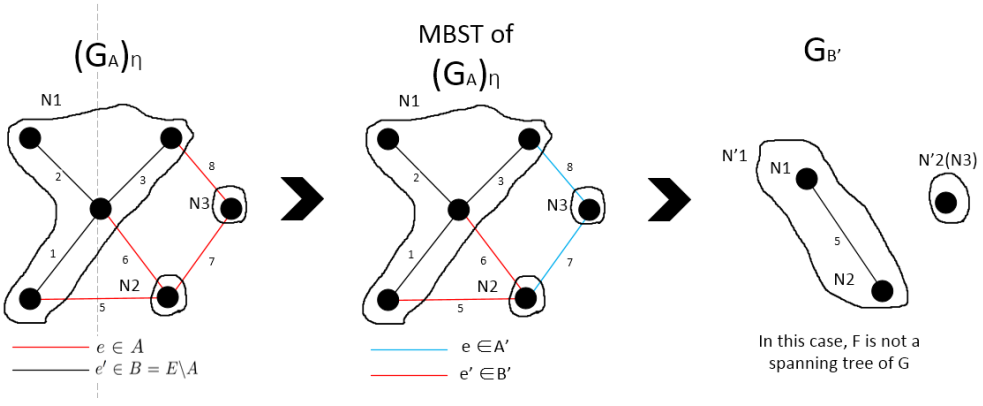


Figure 4: Procedure 2

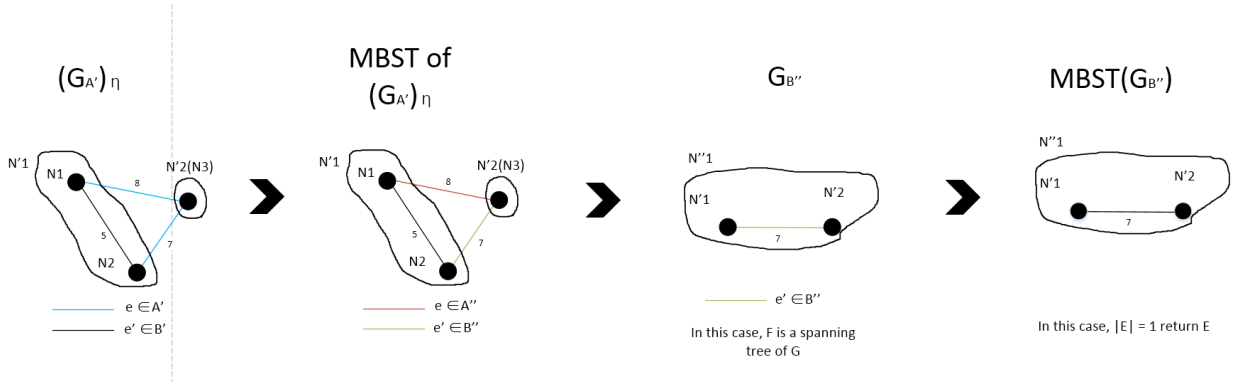


Figure 5: Procedure 3

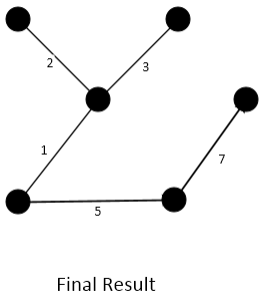


Figure 6: Procedure 4

### 2.3 Find MSBA in graph $G$

**Camerini's algorithm[2]** The Camerini's algorithm for MBSA is pretty much the same as the algorithm for MBST. It has the following concept we need to know before we go into the details of the pseudocode.

1. $T$ in the algorithm represents a subset of E for which it is know that $G_T$ does not contain any spanning arborescence rooted at node "a". Initially $T$ is empty.
2. $UH$ takes $(E-T)$ set of edges in $G$ and returns $A \subset (E-T)$ such that:

$$|A| = \left\lfloor \frac{(|E-T|)}{2} \right\rfloor$$

and

$$W_a \geq W_b, a \in Aandb \in B$$

3. $BUSH(G)$ returns a maximal arborescence of $G$ rooted at node "a".

The following algorithm is suggested by Camerini:

**Procedure 2** MBSA(G,w,T)

let $E$ be the set of edges of $G$;
**if** $|E-T|>1$ **then**
  $A \leftarrow UH(E-T)$;
  $B \leftarrow (E-T)\backslash A$;
  $F \leftarrow BUSH(G_{B\cup T})$;
  **if** $F$ is a spanning arborescence of $G$ **then**
    $S \leftarrow F$;
    $MBSA((G_{B\cup T}, w, T))$;
  **else**
    $MBSA(G, w, T\cup B)$;
  **end if**
**end if**

**Time Complexity** Accroding to [2], the procedure $UH$ requires $O(E)$, because the $UH$ function here is the same as the $UH$ function in the previous MBST function. In addition, $BUSH$ procedure requires $O(E)$ at each execution by applying a depth-first search for digraphs [1]. The number of these executions is $O(logE)$ since $|E-T|$ is being halved at each call of MBSA. To sum up, the total run time of this algorithm is $O(ElogE)$.

**Example** The following example shows that how the algorithm works.

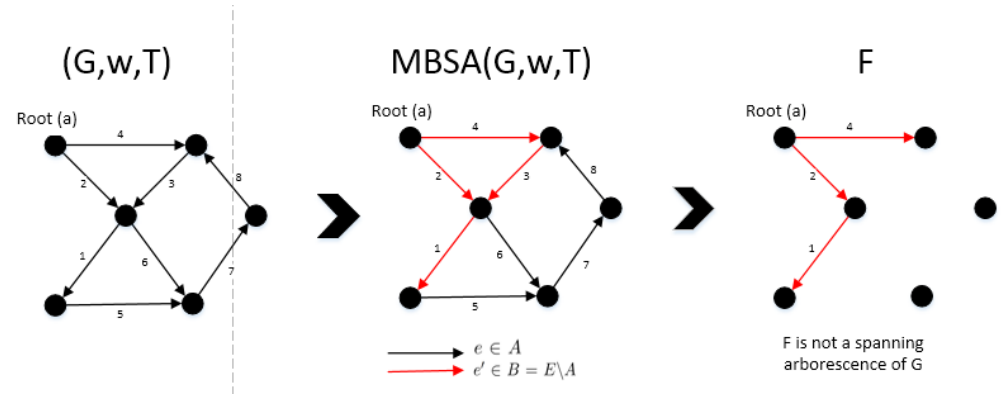


Figure 7: Procedure 1

The original graph is $(G, w, T)$, when we run the $MBSA(G, w, T)$, we can get $F$. Because $F$ is not a spanning arborescence of $G$, we run $MBSA(G, w, T\cup B)$.

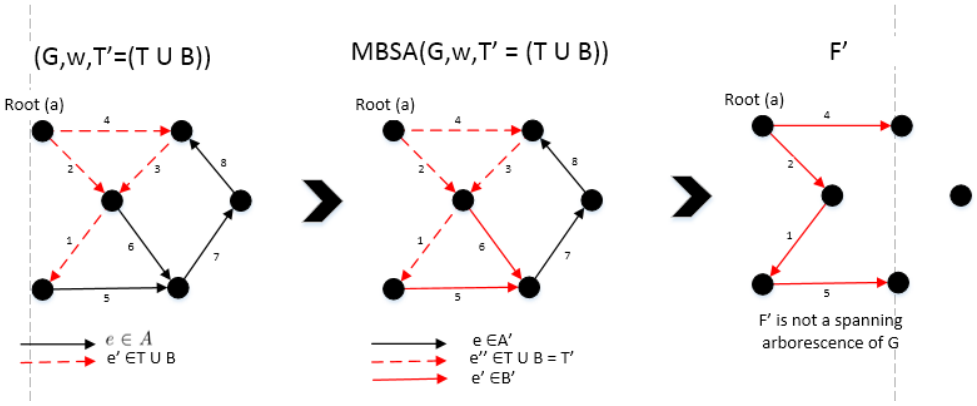


Figure 8: Procedure 2

After we run the $MBSA(G, w, T \cup B)$, we get $F'$ and $F'$ is not a spanning arborescence of $G$, so we run $MBSA(G, w, T' \cup B')$ again.

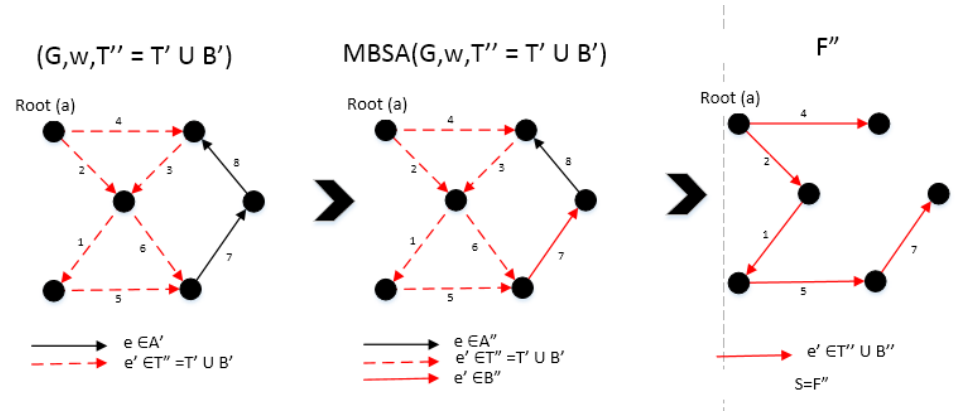


Figure 9: Procedure 3

After we run the $MBSA(G, w, T' \cup B')$, we get $F$" and $F$" is a spanning arborescence of $G$, so we run $MBSA(G_{B" \cup T"}, w, T")$.

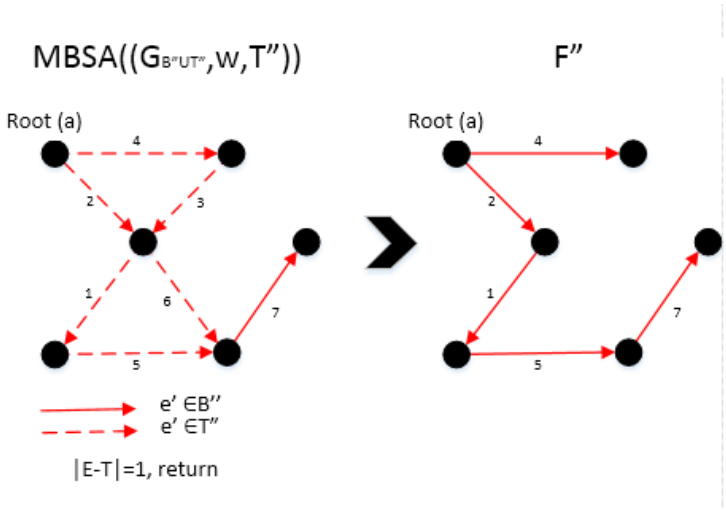


Figure 10: Procedure 4

After we run $MBSA(G_{B"\cup T"}, w, T")$, $|E - T| = 1$, so the algorithm will return and we ge the final answer which is $S = F"$.

**Gabow and Tarjan algorithm[5]** Gabow and Tarjan modified the Dijkstra's algorithm for single-source shortest path algorithm[3] to produces an MBSA. The algorithm is shown below:
In a directed graph $G = (V, E)$, starting from the root $s$ that has a path to all nodes in $G$ (This connected graph is a Tree), the algorithm will compute a parent $p(v)$ in the tree for each $v \neq s$. It maintains a collection of vertices $F$ that are candidates for inclusion in the tree. Each vertex $v \in F$ has an associated cost $c(v)$ for inclusion, which is the minimum cost of an edge from a vertex already in the tree to $v$. Initially, $F = s$ and $c(s) = -\infty$. The procedure[9] of the algorithm is shown follow:

**Procedure 3** MBSA-GT(G,w,T)

```
for |V| times do
  Select v with minimum c(v) from F;
  Delete it from the F;
  for ∀edge(v, w) do
    if w ∉ F or ∉ Tree then
      add w to F;
      c(w) = c(v, w);
      p(w) = v;
    else
      if w ∈ F and c(w) > c(v, w) then
        c(w) = c(v, w);
        p(w) = v;
      end if
    end if
  end for
end for
```

**Time Complexity** If a Fibonacci heap[4] is used to implement the frontier set $F$, this algorithm runs in $O(|V|log|V| + |E|)$ time.

**Example** The following example shows that how the algorithm works, the right graph in the figure is $G(V, E)$.

F

| Vertex X | C(v) | P(v) [parent of v] |
|---|---|---|
| 6 | -∞ | - |
| 5 | 2 | 6 |
| 4 | 4 | 6 |
| | | |
| | | |
| | | |
| | | |

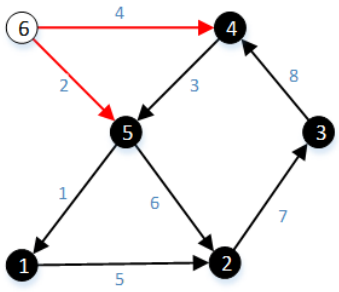


Figure 11: Procedure 1

At the first step of the algorithm we select the root $s$ from the graph $G$, in the above figure, vertex 6 is the root $s$. Then we found all the $edge(6, w) \in E$ and their cost $c(6, w)$, where $w \in V$.

F

| Vertex X | C(v) | P(v) [parent of v] |
|---|---|---|
| ~~6~~ | -∞ | - |
| 5 | 2 | 6 |
| 4 | 4 | 6 |
| 1 | 1 | 5 |
| 2 | 6 | 5 |
| | | |
| | | |

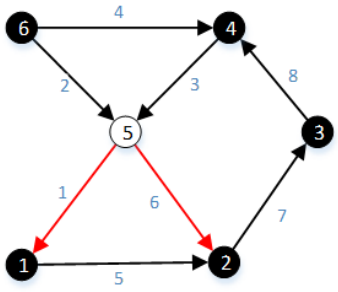


Figure 12: Procedure 2

Next we move to the vertex 5 in the graph $G$, we found all the $edge(5, w) \in E$ and their cost $c(5, w)$, where $w \in V$.

F

| Vertex X | C(v) | P(v) [parent of v] |
|---|---|---|
| ~~6~~ | -∞ | - |
| ~~5~~ | 2 | 6 |
| 4 | 4 | 6 |
| 1 | 1 | 5 |
| 2 | 6 | 5 |
| remove-> 5 | 3 | 4 |
| | | |

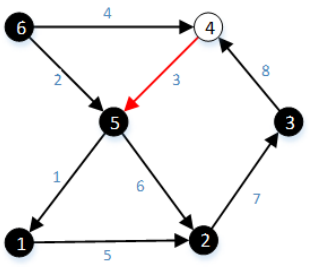


Figure 13: Procedure 3

Next we move to the vertex 4 in the graph $G$, we found all the $edge(4, w) \in E$ and their cost $c(4, w)$, where $w \in V$. We find that the $edge(4, 5) > edge(6, 5)$, so we keep $edge(6, 5)$ and remove the $edge(4, 5)$.

F

| Vertex X | C(v) | P(v) [parent of v] |
|---|---|---|
| ~~6~~ | -∞ | - |
| ~~5~~ | 2 | 6 |
| ~~4~~ | 4 | 6 |
| 1 | 1 | 5 |
| 2 | 6 | 5 |
| 2 | 5 | 1 |
| | | |

replace->

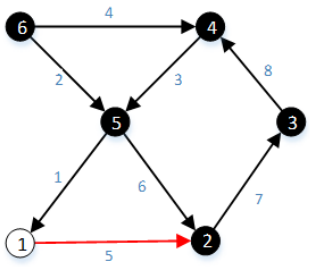


Figure 14: Procedure 4

Next we move to the vertex 1 in the graph $G$, we found all the $edge(1, w) \in E$ and their cost $c(1, w)$, where $w \in V$. We find that the $edge(5, 2) > edge(1, 2)$, so we remove $edge(5, 2)$ and keep $edge(1, 2)$.

F

| Vertex X | C(v) | P(v) [parent of v] |
|---|---|---|
| ~~6~~ | -∞ | - |
| ~~5~~ | 2 | 6 |
| ~~4~~ | 4 | 6 |
| ~~1~~ | 1 | 5 |
| 2 | 5 | 1 |
| 3 | 7 | 2 |
| | | |

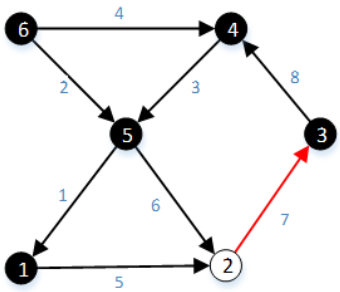


Figure 15: Procedure 5

Next we move to the vertex 2 in the graph $G$, we found all the $edge(2, w) \in E$ and their cost $c(2, w)$, where $w \in V$.

F

| Vertex X | C(v) | P(v) [parent of v] |
|---|---|---|
| ~~6~~ | -∞ | - |
| ~~5~~ | 2 | 6 |
| ~~4~~ | 4 | 6 |
| ~~1~~ | 1 | 5 |
| ~~2~~ | 5 | 1 |
| 3 | 7 | 2 |
| remove-> 4 | 8 | 3 |

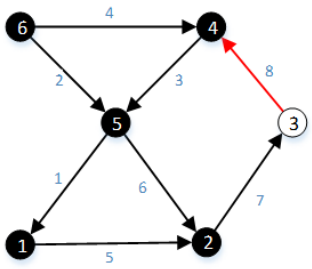


Figure 16: Procedure 6

Next we move to the vertex 3 in the graph $G$, we found all the $edge(3, w) \in E$ and their cost $c(3, w)$, where $w \in V$. We find that the $edge(3, 4) > edge(6, 4)$, so we remove $edge(3, 4)$ and keep $edge(6, 4)$.

F

| Vertex X | C(v) | P(v) [parent of v] |
|---|---|---|
| ~~6~~ | -∞ | - |
| ~~5~~ | 2 | 6 |
| ~~4~~ | 4 | 6 |
| ~~1~~ | 1 | 5 |
| ~~2~~ | 5 | 1 |
| ~~3~~ | 7 | 2 |
| End | | |

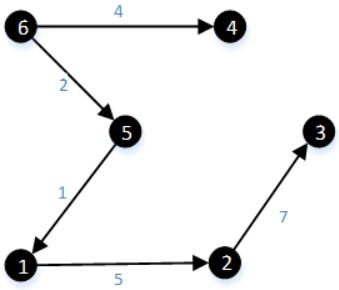


Figure 17: Procedure 7

After we loop through all the vertices in the graph $G$, the algorithm has finished.

## 3 Exercises

**1** [8] Give a Linear time algorithm to determine if a graph $G(V, E)$ contains a MBST with its $maximumedge <= b$, where $b$ is a given constant.

## References


[1] Aho, A. V., & Hopcroft, J. E. (1974). Design & Analysis of Computer Algorithms. Pearson Education India.

[2] Camerini, P. M. (1978). The min-max spanning tree problem and some extensions. Information Processing Letters, 7(1), 10-14.

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

[4] Fredman M L, Tarjan R E. Fibonacci heaps and their uses in improved network optimization algorithms[J]. Journal of the ACM (JACM), 1987, 34(3): 596-615.

[5] Gabow H N, Tarjan R E. Algorithms for two bottleneck optimization problems[J]. Journal of Algorithms, 1988, 9(3): 411-417.

[6] Schnhage, A., Paterson, M. S., & Pippenger, N. (1975). Finding the median.

[7] `https://en.wikipedia.org/wiki/Minimum_bottleneck_spanning_tree#Definitions`

[8] `http://flashing-thoughts.blogspot.ru/2010/06/everything-about-bottleneck-spanning.html`

[9] `http://people.scs.carleton.ca/~maheshwa/courses/5703COMP/14Seminars/BST-Report.pdf`