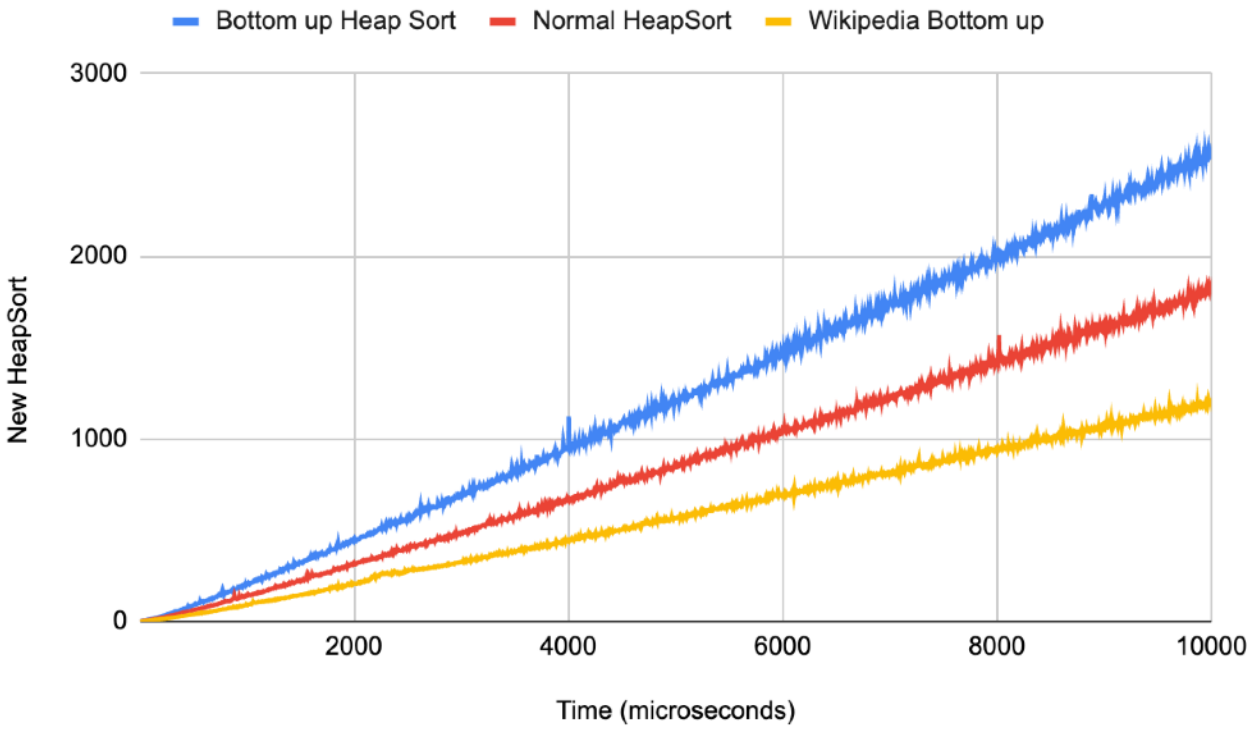
(c) Bonus (3 points) Compare the original Heap Sort and its variant from subpoint (b) for input sequences of different lengths (including larger input sequences). What can you observe?

For reference, I also included another Bottom up Heapsort Algorithm taken from Wikipedia (in German for some reason)that "correctly" implements the bottom up variant of HeapSort. The code is as follows:

```
/// @author The wonderful people at wikipedia :)
template <class T> void heapsort_bu( T * data, int n ) // zu sortierendes Feld und seine
Länge
{
        int val, parent, child;
        int root= n >> 1;               // erstes Blatt im Baum
        int count= 0;                   // Zähler für Anzahl der Vergleiche

        for ( ; ; )
        {
                if ( root ) {                   // Teil 1: Konstruktion des Heaps
                        parent= --root;
                        val= data[root];            // zu versickernder Wert
                }
                else
                if ( --n ) {                    // Teil 2: eigentliche Sortierung
                        val= data[n];               // zu versickernder Wert vom Heap-Ende
                        data[n]= data[0];           // Spitze des Heaps hinter den Heap in
                        parent= 0;                  //   den sortierten Bereich verschieben
                }
                else                            // Heap ist leer; Sortierung beendet
                        break;

                while ( (child= (parent + 1) << 1) < n )  // zweites (!) Kind;
                {                                   // Abbruch am Ende des Heaps
                        if ( ++count, data[child-1] > data[child] )  // größeres Kind wählen
                                --child;

                        data[parent]= data[child];     // größeres Kind nach oben rücken
                        parent= child;              // in der Ebene darunter weitersuchen
                }

                if ( child == n )               // ein einzelnes Kind am Heap-Ende
                {                               //   ist übersprungen worden
                        if ( ++count, data[--child] >= val ) {  // größer als der zu versick-
                                data[parent]= data[child];   //   ernde Wert, also noch nach oben
                                data[child]= val;           // versickerten Wert eintragen
                                continue;
                        }

                        child = parent;             // 1 Ebene nach oben zurück
                }
                else
                {
                        if ( ++count, data[parent] >= val ) {  // das Blatt ist größer als der
```

```
                        data[parent]= val;          //   zu versickernde Wert, der damit
                        continue;                   //   direkt eingetragen werden kann
                    }

                    child= (parent - 1) >> 1;      // 2 Ebenen nach oben zurück
                }

                while ( child != root )          // maximal zum Ausgangspunkt zurück
                {
                    parent= (child - 1) >> 1;      // den Vergleichswert haben wir bereits
                    //   nach oben verschoben
                    if ( ++count, data[parent] >= val )  // größer als der zu versickernde
                        break;                      //   Wert, also Position gefunden

                    data[child]= data[parent];     // Rückverschiebung nötig
                    child= parent;                 // 1 Ebene nach oben zurück
                }

                data[child]= val;                  // versickerten Wert eintragen
            }

            return;
}
```

Here are the results running 50 repetitions sorting arrays with random ordered inputs from 0 to 99 using array sizes from 5 to 10000:



The bottom up version of HeapSort derived from the instructions was noticeably slower than the normal implementation of Heap Sort. However, the correct implementation of Bottom Up HeapSort from wikipedia vastly outperformed the other two variants of Heap Sort.