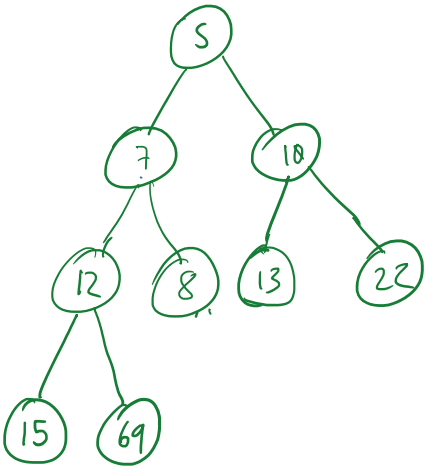
The Heap ADT and its implementations.

Friday, November 15, 2019 5:08 PM

- A heap is:
  - A binary tree
  - where all but the last level are complete
  - where for every element x, x is smaller than all its descendants.

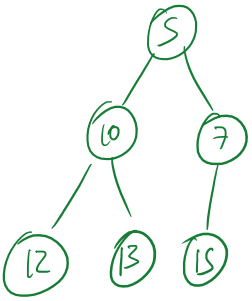
Operations:

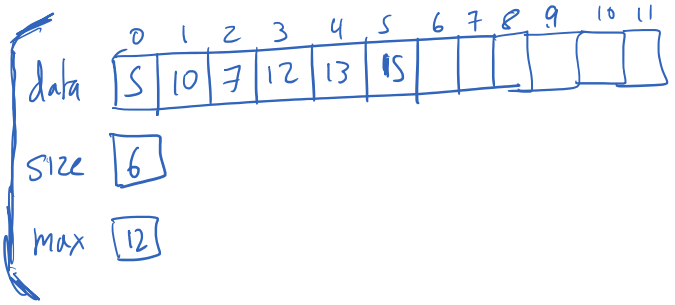
- Insert (x, t) "percolate up"
- getMin (t)
- removeMin (t) "percolate down"

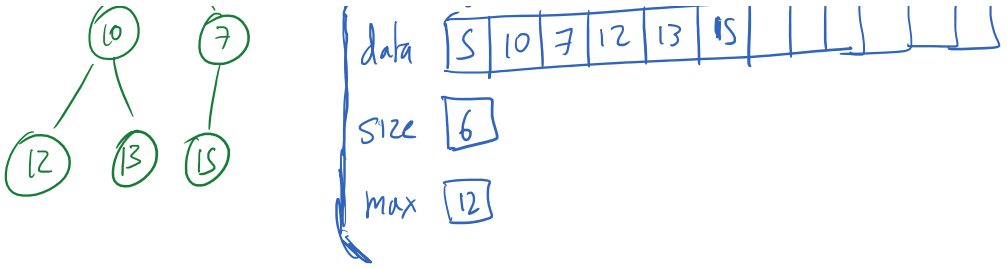


- getMin (t) is $O(1)$
- insert (x, t) is $O(\log n)$
- removeMin (t) is $O(\log n)$

- Data Structure: Array Heap.





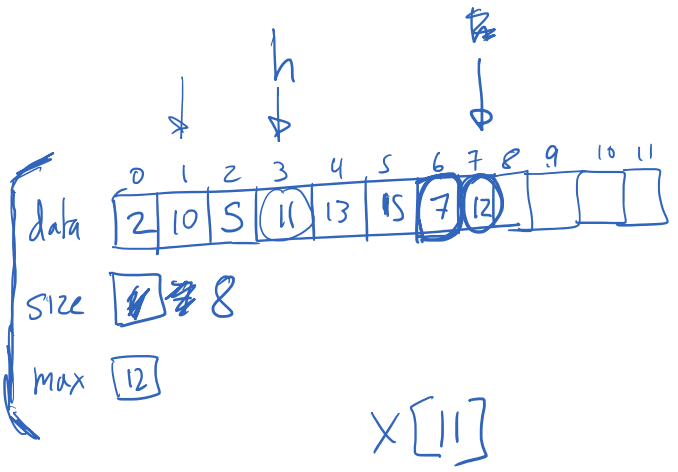


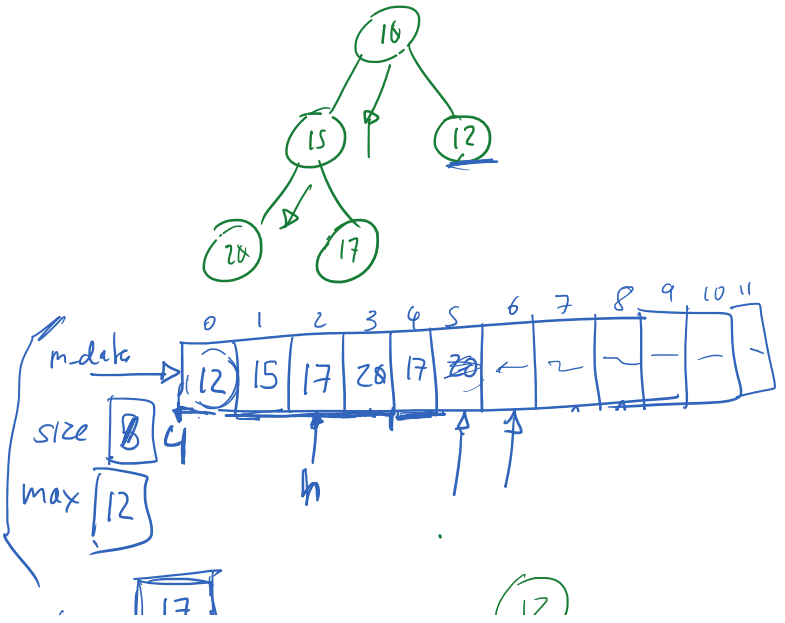

```
template <typename T>
class ArrayHeap{
public:
   T* m_data;
   int m_size;
   int m_max;
};
```

```
getMin()
{
   if (m_size != 0)
      return m_data[0];
   else
      throw tantrum();
};
```

```
insert( T x )
{
   int hole = m_size;
   m_size++;

   while (hole>0 && x<m_data[(hole-1)/2]){
      m_data[hole] = m_data[(hole-1)/2]
      hole = (hole-1)/2;
   }
   m_data[hole] = x;
};
```






```
removeMin()
{
  tmp = data[m_size-1];
  m_size--;

  int hole=0;
  bool done=false;

  while (hole*2+1 < m_size && !done)
```

```
  bool done=false;

  while (hole*2+1 < m_size && !done)
  {
     int promo = hole*2 + 1;
     if ( promo+1 < m_size &&
          m_data[promo+1] < m_data[promo] )
         promo++;

     if ( m_data[promo] < tmp ){
        m_data[hole] = m_data[promo];
        hole = promo;
     }
     else
        done = true;
  }
  m_data[hole] = tmp;
}
```