Name: Boboc Bogdan
Group: 911

Priority Queue

Problem statement
ADT Priority Queue – implementation on a binary heap.
Domain
A priority queue is an abstract data type which is like a regular queue or stack data structure, but where additionally each element has a "priority" associated with it. In a priority queue, an element with high priority is served before an element with low priority. If two elements have the same priority, they are served according to their order in the queue.
H = { h / h is a heap with elements of type TElem }.
Interface

new
- creates a new empty priority queue
pre: true
post: h belongs to H, is an empty priority queue
push
- pushes an element into priority queue
pre: h belongs to H, e is a TElem
post: h' belongs to H, h = h u {e}

```
function push(TElem value)
      heap[n] <- value
      v: Integer
      v <- n;
      n <- n+1
      while( v != 0 ) do
         if( heap[ v ] < heap[ (v-1)/2 ] ) do
                  heap[ v ], heap[ (v-1)/2 ] <- heap[ (v-1)/2 ], heap[ v ]
         else
                  break;
      end_while
   end_function
```

pop
- erases the element from the top of the priority queue
pre: h belongs to H
post: h' belongs to H

```
function pop()
      heap[ 0 ], heap[ n-1 ] <- heap[ n-1 ],heap[ 0 ]
      n--
      v: Integer
      v <- 0
      while(true)
         if((2*v+1 < n and heap[ 2*v+1 ] < heap[ v ]) and ( 2*v+2 >= n or !(heap[
2*v+2 ] < heap[ 2*v+1 ]))) do
            heap[ v ], heap[ 2*v+1 ] <-  heap[ 2*v+1 ] ,heap[ v ]
            v <- 2*v+1
            continue
```

```
            end_if
            if((2*v+2 < n and heap[ 2*v+2 ] < heap[ v ] and heap[ 2*v+2 ] < heap[ 2*v+1
])) do
               heap[ v ], heap[ 2*v+2 ] <- heap[ 2*v+2 ] , heap[ v ]
                v <- 2*v+2;
                continue;
            end_if
            break;
        end_while
    end_function
```

top
- returns the top of the priority queue
pre: h belongs to H
post: e is a TElem

```
function top()
    top <- heap[ 0 ]
end_function
```

size
- returns the number of elements that are in the priority queue
pre: h belongs to H
post: s is a Integer, s <- n

```
function size()
    top <- n
end_function
```

Representation

PQueue:

```
n : Integer
h : TElem[]
```

Applications

Given a undirected graph with costs and two nodes A and B find all the edges that are on a minimum cost path between the 2 given nodes.

Why a priority queue is suitable for solving this problem:
- we only need to push, pop and get the maximum element from the queue
- using a priority queue, we can make Dijkstra's algorithm run in O(N log N) time, where N is the number of nodes.