DATA STRUCTURES AND ITS APPLICATIONS UE19CS202

Shylaja S S & Kusuma K V
Department of Computer Science & Engineering

DATA STRUCTURES AND ITS APPLICATIONS

Implementation of Priority Queue using min heap/max heap

Shylaja S S
Department of Computer Science & Engineering

Ascending and Descending Heap

- Ascending Heap: Root will have the lowest element. Each node's data is greater than or equal to its parent's data. It is also called min heap.

- Descending Heap: Root will have the highest element. Each node's data is lesser than or equal to its parent's data. It is also called max heap.

DATA STRUCTURES AND ITS APPLICATIONS

Priority Queue using Heap

- Priority Queue is a Data Structure in which intrinsic ordering of the elements does determine the results of its basic operations

- Ascending Priority Queue: is a collection of items into which items can be inserted arbitrarily and from which only the smallest item can be removed

- Descending Priority Queue: is a collection of items into which items can be inserted arbitrarily and from which only the largest item can be removed

Priority Queue using Heap

Consider the properties of a heap

- The entry with largest key is on the top(Descending heap) and can be removed immediately. But O(logn) time is required to readjust the heap with remaining keys

- If another entry need to be done, it requires O(logn)

- Therefore, heap is advantageous to implement a Priority Queue

Priority Queue using Heap: Implementation

- dpq: Array that implements descending heap of size k (position from 0 to k-1)
- pqinsert(dpq,k,elt): insert element into the heap dpq of size k. Size increases to k+1
- This insertion is done using siftup operation

Priority Queue using Heap: Implementation

Algorithm for siftup

```
c = k;
p = (c-1)/2;
while(c>0 && dpq[p]<elt) {
   dpq[c]=dpq[p];
   c=p;
   p=(c-1)/2;
}
dpq[c]=elt;
```

Priority Queue using Heap: Implementation

pqmaxdelete(dpq,k) //for a descending heap of size k

```
p = dpq[0];

adjustheap(0,k-1)

return p;
```

Priority Queue using Heap: Implementation

Algorithm largechild(p,m)

```
c = 2*p+1;
if(c+1 <= m && x[c] < x[c+1])
  c=c+1;
if(c > m)
  return -1;
else
  return (c);
```

Priority Queue using Heap: Implementation

Algorithm adjustheap(root,k) //recursive

```
p = root;
c = largechild(p,k-1);
if(c >= 0 && dpq[k] < dpq[c]){
  dpq[p] = dpq[c];
  adjustheap(c,k);
}
else
  dpq[p] = dpq[k];
```

Priority Queue using Heap: Implementation

Iterative version

```
p = root;
kvalue = dpq[k];
c = largechild(p,k-1);
while(c >= 0 && kvalue < dpq[c]){
    dpq[p] = dpq[c];
    p = c;
    c = largechild(p,k-1);
}
dpq[p] = kvalue;
```

THANK YOU

Shylaja S S

Department of Computer Science & Engineering

shylaja.sharath@pes.edu

+91 9449867804