

Bill Davis

605.421

1. 6.5-8 Give an  $\mathbf{O}(n \lg(k))$ -time algorithm to merge  $k$  sorted lists with  $n$  total elements

This can be done by creating a min-heap containing  $k$  elements, one from each list. We can then merge the lists by taking one min element from the heap, replacing it with another from the one of the  $k$  lists and repeating.

This requires  $\mathbf{O}(2n \lg(k))$  since for each element we need to first add it to the min-heap requiring  $\mathbf{O}(\lg(k))$ -time, then we need to perform HEAP-EXTRACT-MAX, again requiring  $\mathbf{O}(\lg(k))$ -time.

We also need to add another member to the heap nodes indicating which list the element originated from, in order to replace it with an element from the same list.

2. 6-2 Analysis of D-ary heaps

- (a) How to represent a d-ary heap as an array.

The root of the tree is  $A[1]$ , its children are in  $A[2]$  through  $A[d+1]$ . Given the index  $i$  of a node we can compute its PARENT, and  $k^{th}$  child by

$$\text{PARENT}(i) = \lfloor \frac{i-2}{d+1} \rfloor$$

To find the  $k$ -th child of a node  $i = di - d + k + 1$

- (b) Since each level  $n$  has  $n^d$  nodes for a d-ary tree, the height of the tree equals  $\lg_d(n)$ .
- (c) EXTRACT-MAX does not need to be altered for d-ary heaps. The only operation that needs to change is MAX-HEAPIFY. Therefore the runtime of EXTRACT-MAX = runtime of MAX-HEAPIFY. This equals  $\mathbf{O}(\lg_d(n))$  for a d-ary tree of size  $n$ .
- (d) The HEAP-INSERT does not need to be altered for d-ary heaps. Since only parent nodes are examined in HEAP-INCREASE-KEY, each  $\lg_d(n)$  parent could be compared to the new node resulting in a runtime of  $\mathbf{O}(\lg_d(n))$ .

(e) HEAP-INCREASE-KEY can be altered as follows

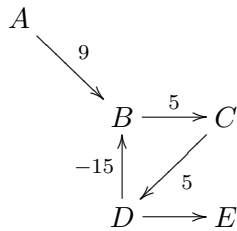
```

If key > A[i]
  then error
A[i] = key
while i > 1 and A[Parent[i]] < A[i]
  do exchange A[i] and A[Parent[i]]
  i = Parent[i]

```

We may need to traverse the whole height of the tree for each call to this function, which as we mentioned before can be done in  $\lg_d(n)$  time.

3. 24.3-2 An example where Dijkstras algorithm fails to produce correct results



The proof for Dijkstras fails because equation 24.2 does not hold. In particular

$$d[y] = \delta(s, y) \not\Rightarrow d[y] \leq \delta(s, u)$$

If edge lengths can be negative then  $\delta(s, u)$  may be less than  $\delta(s, y)$ .