Heapisation. We want a heapify (A) procedure that puts array elements A [1..n] in heap order. Trivial?
for i ← 1, ..., n do swim (A [i], i, A) takes Θ (n log n) at worst. A better solution: for i ← ⌊n / 2⌋,. . . , 1 do sink (A [i], i, A, n)

heapify
puts the elements in the order of heaps in time O (n).

Demonstration. sink takes O (h) time where h is the height of the node that corresponds to the index i in the tree representation of the heap. There are ≤ ⌈n / 2h⌉ internal nodes with height h. So the computation time is limited by

Sort by heap. The priority queue allows you to sort a collection of elements: inserting n elements + call deleteMin n times. With a binary heap, we can sort it in place: it is the heapsort algorithm. After heapify, we maintain the order of heaps in the prefix A [1..i] in a loop i ← n, n - 1,. . . 2. The suffix A [i..n] is always sorted in descending order. At each iteration, after exchanging A [1] ↔ A [i], we restore the order of heaps in A [1..i - 1] for the next iteration.

Heapisation. On veut une procédure heapify(A) qui met les éléments du tableau A[1..n] dans l'ordre de tas. Triviale?
for i ← 1,...,n do swim(A[i], i, A) prend Θ(n log n) au pire. Une meilleure solution : for i ← ⌊n/2⌋, . . . , 1 do sink(A[i], i, A, n)

heapify
met les éléments dans l'ordre de tas en temps O(n).

Démonstration. sink prendO(h) temps où h est la hauteur du nœud qui correspond à l'indice i dans la représentation arborescente du tas. Il y a ≤ ⌈n/2h⌉ nœuds internes avec hauteur h. Donc le temps de calcul est borné par

Tri par tas. La file de priorité permet donc de trier une collection d'éléments : insertion de n éléments + appeler n fois deleteMin. Avec un tas binaire, on peut faire le tri en place : il s'agit de l'algorithme du tri par tas (heapsort). Après heapify, on maintient l'ordre de tas dans le préfixe A[1..i] en une boucle i ← n, n − 1, . . . 2. Le suffixe A[i..n] est toujours trié en ordre décroissant. À chaque itération, après avoir échangé A[1] ↔ A[i], on rétablit l'ordre de tas en A[1..i − 1] pour la prochaine itération.