```
int extractMin(int H[]) {
    int res = H[0];
    H[0] = H[h.size - 1];
    h.size -= 1;
    minHeapify(H, 0, h.size-1);
}
int extractMax(int H[]) {
    int res = H[0];
    H[0] = H[h.size - 1];
    h.size -= 1;
    maxHeapify(H, 0, h.size-1);
}
```

```
void maxHeapify(int H[], int i, int hd){
    int max = (H[left(i, hd)] >= H[right(i, hd)]) ? left(i, hd) : right(i, hd);
    max = (H[max] > H[i]) ? max : i ;
    if(max != i) {
        swap(H, i, max);
        maxHeapify(H, max, hd);
    }
}
```

```
void minHeapInsert(int H[], int x) {
    H.size = H.size+1;
    int hd = H.size - 1;
    int i = hd; // ultimo elemento dell'array
    H[hd] = x;
    while (i > 0 && H[parent(i,hd)] > H[i]) {
        swap(H, parent(i, hd), i);
        i = parent(i, hd);
    }
}
void maxHeapInsert(int H[], int x) {
    H.size = H.size+1;
    int hd = H.size - 1;
    int i = hd; // ultimo elemento dell'array
    H[hd] = x;
    while (i > 0 && H[parent(i,hd)] < H[i]) {
        swap(H, parent(i, hd), i);
        i = parent(i, hd);
    }
}
```

```
void buildMinHeap(int A[], int dim) {
    int m = dim / 2 + 1;
    for(int i = m; i >= 0; i--) {
        minHeapify(A, i, dim);
    }
}
// pre:  hd < dim A[]
// post: A[0..hd] e' uno heap massimo
void buildMaxHeap(int A[], int dim) {
    int m = dim/2 +1;
    for(int i = m; i >= 0; i--) {
        maxHeapify(A, i, dim);
    }
}
```

```
void minHeapify(int H[], int i, int hd){
   int min = (H[left(i, hd)] <= H[right(i, hd)]) ? left(i, hd) : right(i, hd);
   min = (H[min] < H[i]) ? min : i ;
   if(min != i) {
       swap(H, i, min);
       minHeapify(H, min, hd);
   }
}
```

```
void heapSort(int A[], int dim) {
    int hd = dim-1;
    buildMaxHeap(A, hd);
    for (int i = hd; i > 0; i--) {
        swap(A, 0, i);
        maxHeapify(A, 0, i-1);
    }
}
```