第二関数

```
def heapify(arr, n, i):
    largest = i # 親ノード
    l = 2*i + 1  # 左の子ノード
    r = 2*i + 2  # 右の子ノード

    # 左の子ノードが親ノードより大きい場合
    if l < n and arr[largest] < arr[l]:
        largest = l
    # 右の子ノードが親ノードより大きい場合
    if r < n and arr[largest] < arr[r]:
        largest = r
    # 親ノードが最大値でない場合、親ノードと最大値を交換する
    if largest != i:
        arr[i], arr[largest] = arr[largest], arr[i]

        # 子ノードのヒープも修正する
        heapify(arr, n, largest)
```

入力引数の３番目＝親ノード
　左右の子ノードの添字を求める

左右の子ノード、親ノードの間で
　最大値を求める

最大値を親ノードにセットし
　子ノードに親ノードの値をセットする
　　※入れ替える

引数
arr　入力引数の２番目　最大値の添字

第一関数

```
def heap_sort(arr):
    n = len(arr)

    # 最初に、配列をヒープ構造にする
    for i in range(n // 2 - 1, -1, -1):
        heapify(arr, n, i)
    # ヒープから一つずつ最大値を取り出して、配列の末尾に配置する
    for i in range(n-1, 0, -1):
        arr[i], arr[0] = arr[0], arr[i]  # 最大値を末尾に配置
        heapify(arr, i, 0)  # 未ソート部分のヒープを再構築する
```

引数
arr　7　（2, 1, 0）

引数
arr　（6, 5, 4, 3, 2, 1）　0

メイン処理

```
# 使用例
arr = [64, 34, 25, 12, 22, 11, 90]
heap_sort(arr)
print("ソート済み配列：")
for i in range(len(arr)):
    print("%d" % arr[i])
```

i＝（0, 1, 2, 3, 4, 5, 6）