アルゴリズムとデータ構造 2015　第 13 回　　年　　組　　番 氏名

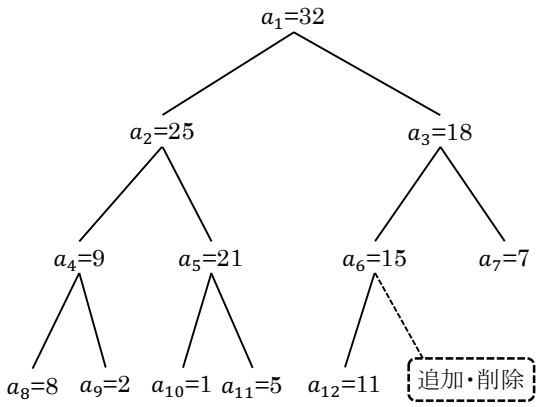
1. 2 分木において，図のように常に，親ノードの値≧子ノードの値という条件（またはその逆）が成り立つものを **2 分ヒープ**と呼ぶ。**ヒープソート**は配列の添字を使い，a_1を根としてa_kの左右の子をそれぞれa_{2k}，a_{2k+1}と定義することでほぼ平衡な 2 分木を構築する（今回は簡単のためa_0（a[0]）は使わない）。

ヒープの構築は根（a_1）だけのヒープから始めて，末尾に要素を追加していく方法が簡単である。追加された要素は，順々に親と比較・交換しながら，ヒープの条件（親ノードの値≧子ノード）を満たす位置まで上げていく。この操作を**アップヒープ**と呼ぶ。

ヒープでは最大値が根に来るので，根の値を取り出しながらヒープを再構築していくことでソートができる。根を取り出した後には，いったん末尾の要素を根に置き，順々に左右の子と比較・交換しながら，ヒープの条件を満たす位置まで下ろしていく。こちらの操作は**ダウンヒープ**と呼ぶ。

下記に示したのは，ヒープソートのプログラム例である。空欄を適当に埋め，適当なクラス等を補って実行させて，ヒープソートのしくみを理解せよ。

プログラムが理解できたら，a_0（a[0]）から配列の全要素をソートするためにはどう改良すればいいか考えてみよ。




```
// a[1]～a[last]をヒープソートで整列する
public static void heapSort(double[] a, int last) {

    // ヒープになっている領域を 1 つずつ広げていく
    for (int i = 1; i <= last; i++)
        upHeap(a, i);

    // 最大値を取り出し，ヒープを 1 つずつ縮めていく
    for (int i = last; i >= 2; i--)
        downHeap(a, i);
}

// アップヒープ操作： a[1]～a[last-1]のヒープができているとき，
// a[last]を追加してからヒープを再構成して a[1]～a[last]に 1 つ分広げる
private static void upHeap(double[] a, int last) {

    // 追加要素の添字を i として，順々に親と比較して上げていく
    int i = last;

    for (;;) {
        // a[i]の親 a[j]の添字 j を計算で求める

        int j =

        // 親がいないか親の方が大きかったら，もう上げる必要はないのでループから脱する

        if (j == 0 || a[   ] >= a[   ])
            break;

        // 親の方が小さければ、追加要素を親と交換して一段上げる
        swap(a, i, j);

        // 追加要素の新しい位置を i とする

        i =
    }
}
```

```
// ダウンヒープ操作： a[1]〜a[last]のヒープができているとき，
// 根（a[1]）にある最大値を a[last]と交換し，ヒープを再構成して a[1]〜a[last-1]に 1 つ分縮める
private static void downHeap(double a[], int last) {

    // まず a[1]（最大値）を a[last]と交換し，a[last]を切り離す
    swap(a, 1, last);

    // いったん根に上げた要素 a[1]を，順々に下げていく
    int i = 1;

    for (;;) {
        // a[i]の左の子 a[j]の添字 j を計算で求める

        int j =

        // j が範囲外ならもう子はいないので，ループから脱する
        if (j >= last) break;

        // 右にも子がいて左の子よりも大きかったら，右の子を比較対象に再設定する

        if (j + 1 < last && a[   ] < a[       ])
            j = j + 1;

        // 子の方が小さかったらもう下げる必要はないので，ループから脱する
        if (a[   ] >= a[   ]) break;

        // 子の方が大きければ，いったん上げた要素を子と交換して一段下げる
        swap(a, i, j);

        // いったん上げた要素の新しい位置を i とする

        i =
    }
}

// 配列要素の交換
private static void swap(double[] a, int i, int j) {
    double t = a[j];
    a[j] = a[i];
    a[i] = t;
}


public static void main(String[] args) {

    double[] array = new double[21];

    for (int i = 1; i < array.length; i++)
        array[i] = Math.random() * 100;

    heapSort(array, array.length - 1);

    for (int i = 1; i < array.length; i++)
        System.out.printf("%2d: %2.0f%n", i, array[i]);
    System.out.println();
}
```