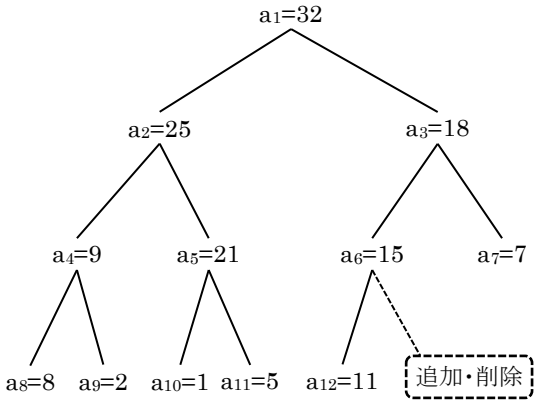
1. 2 分木において，図のように常に，親ノードの値≧子ノードの値という条件（またはその逆）が成り立つものを **2 分ヒープ**と呼ぶ。**ヒープソート**では配列の添字を使い，$a_1$ を根として $a_k$ の左右の子をそれぞれ $a_{2k}$，$a_{2k+1}$ と定義することでほぼ平衡な 2 分木を構築する（今回は簡単のため a[0]は使わない）。

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。追加された要素をその親と比較・交換しながら，ヒープの条件を満たす位置まで上げていく。

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最構築していくことでソートができる。この際，いったん末尾の要素を先頭（根）に入れ，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。

上記の操作は，それぞれ**アップヒープ**と**ダウンヒープ**と呼ばれる。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 main 関数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。




```
void up_heap(double a[], int last);
void down_heap(double a[], int last);

/* a[1]～a[last]をヒープソートで整列する */
void heap_sort(double a[], int last)
{
    int i;

    /* ヒープになっている領域を広げていく */
    for (i = 1; i <= last; i++)
        up_heap(a, i);
    /* 最大値を取り出しヒープを縮めていく */
    for (i = last; i >= 2; i--)
        down_heap(a, i);
}

/* a[1]～a[last-1]のヒープができているとき,
   a[last]を追加してヒープを 1 つ分広げる */
void up_heap(double a[], int last)
{
    int i, j;
    double t;

    /* 追加要素を親と比較して上げていく */
    i = last;
    for (;;) {
        /* a[i]の親の添字 */

        j =

        /* 親の方が大きかったらここでいい */

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

        /* 親の方が小さければ交換して上へ */
        t = a[j]; a[j] = a[i]; a[i] = t;

        i =
    }
}
```

```
/* 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 a[1]～a[last-1]に 1 つ分縮める */
void down_heap(double a[], int last)
{
    int i, j;
    double t;

    /* まず a[1]（最大値）を a[last]と交換 */
    t = a[last]; a[last] = a[1]; a[1] = t;

    /* いったん根に上げた要素を下げていく */
    i = 1;
    for (;;) {
        /* a[i]の左の子の添字 */

        j =

        /* 範囲外ならもう子はいない */
        if (j >= last) break;

        /* 左の子より右の子が大きかったら */

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

        /* 子の方が小さかったらここでいい */

        if (a[   ] >= a[   ]) break;

        /* 子の方が大きければ交換して下へ */
        t = a[j]; a[j] = a[i]; a[i] = t;

        i =
    }
}
```