一次删除单个元素

11

17

12

18

10

222222

5

7

6

2

3

8121

4

删除元素4

这个位置暂时用最后一个元素11来填补，首先向下判断

17

12

18

10

222222

5

7

6

2

3

8121

11

获取两个孩子节点2和3，得到孩子节点的最大值3，由于3<11所以向下不移位。

由于向下判断11所在的位置没有发生变化，所以可以判断移至该位置的节点比孩子节点都大。那么它会不会比父节点还大呢？如果他比父节点还大，直接将其插入该位置那么就不符合堆的特性，所以需要接着向上判断。

获取父节点10，由于10<11，所以将10和11互换位置。

17

12

18

11

222222

5

7

6

2

3

8121

10

再寻找父节点22，由于22>11，所以这便是11的最终位置。

最终删除元素4后堆的结果如图所示。

删除元素8

17

12

18

11

222222

7

6

2

3

5

10

元素5暂代其位置，向下判断，结构变化如下图所示

17

12

18

11

222222

2

3

7

10

5

6

由于孩子节点7>5，所以将元素5和元素7互换位置。由于向下判断位置已经发生变化，可以知道该元素还没有孩子节点大，根据堆的特性可知父节点肯定比该节点大，这时可以不再向上判断，所以这便是删除元素8后的最终结构。

一个删除多个元素

11

17

12

18

10

222222

5

7

6

2

3

8121

4

同时删除元素4和8。首先将其放置栈中，根据堆的特性正好可以优先处理最底端的元素

4

8

取出栈顶元素8，在堆中删除元素8，使用元素11暂时替代，通过向下判断得到如下结构

17

12

18

10

222222

5

7

6

2

3

4

11

我们发现位置11所在的位置没有发生变化，说明此时还需要继续向上判断，如果我们此时直接进行向上判断，那么下一个删除元素10的位置可能会发生变化，所以我们使用一个栈将其暂存。

11

接着再取出第一个栈中的下一个元素4，此时使用元素5暂时替代，通过向下判断得到如下结构。

17

12

18

10

222222

7

6

2

3

5

11

我们发现元素5所在的位置没有发生变化，说明此时还需要继续向上判断，此时我们使用一个栈将其暂存。最终得到的栈元素如下所示

11

5

此时再从栈中取出元素向上判断。

首先取出元素5向上判断，由于5的父节点大于5，所以结构没有发生变化

再继续取出元素11向上判断，得到的结构如下图所示

17

12

18

11

222222

7

6

2

3

5

10

按照这种方式删除两个元素后，仍然满足堆的特性。