```
public class Vertex implements Comparator<Vertex>,
                                Index<Vertex> {
    int d;
    int index;

    public int compare(Vertex u, Vertex v) {
        return u.d - v.d;
    }
    public void putIndex( . . . )
    public int getIndex( . . . )
}
```

IndexedHeap<Vertex, Vertex> h = new IndexedHeap<>(arr, non-null Vertex object, non-null Vertex object);

Computing x^n when x and n are both big integers (represented as a polynomial in base B – Project LP1):

Let $n = \{a_0, a_1, \ldots, a_d\}$ $\quad n = a_0 + a_1 B + a_2 B^2 + \cdots + a_d \cdot B^d$

Let shift(n) be the list obtained from n by discarding the first node of its list:

$s = \text{shift}(n) = \{a_1, a_2, \ldots, a_d\}$ $\quad s = a_1 + a_2 B + a_3 B^2 + \cdots + a_d \cdot B^{d-1}$

It is easy to verify that $n = s * B + a_0$

Algorithm for Power(x, n):

if $d = 0$, then $n = \{a_0\}$. return Power(x, a_0)

else $s = \text{shift}(n)$

xtothes = Power(x, s)

return Power(xtothes, B) * Power(x, a_0)

↑ Product of two polynomials

Computing $\sqrt{x}$:

Binary search: Assume $x > 1$.

```
low = 1                          high = x

while ( low + 1 < high ) do      // LI: low^2 < x
                                 //     high^2 > x
    mid ← (low + high)/2
    if  x < mid * mid then
            high ← mid
    else if  x > mid * mid then
            low ← mid
    else    return mid
return low
```