Pierwszy próg złożoności: *złożoność czasowa* O($n + m$)
Drugi próg złożoności: *złożoność czasowa O(mlogn)* oraz *pamięciowa* O(1).

Dane jest pełne drzewo binarne T zawierające n wierzchołków. Każdy węzeł drzewa zawiera klucz będący liczbą całkowitą. Węzły drzewa numerujemy kolejnymi liczbami naturalnymi w ten sposób, że korzeń ma numer 1, jego synowie mają numery 2 i 3, następny poziom, od lewej do prawej, ma numery $4, 5, 6, 7$ itd. Dany jest ciąg X zawierajacy m liczb naturalnych ze zbioru $(1, \ldots, n)$. Należy założyć, że m jest istotnie mniejsze niż n. Proszę zaimplementować funkcję:

def maxim(T, X)

która zwraca maksymalny klucz spośród węzłów drzewa T o numerach wymienionych w X.

Funkcja powinna być możliwie jak najszybsza - wychodząc z założenia, że $m \ll n$, powinna działać na stałej pamięci (poza pamięcią potrzebną na przechowywanie danych wejściowych). Oszacuj złożoność czasową algorytmu.

Reprezentacja drzewa. Drzewo jest reprezentowane przy pomocy węzłów typu *Node*.

```
class Node:
    def __init__(self):
        self.left = None   # lewe poddrzewo
        self.right = None   # prawe poddrzewo
        self.parent = None   # rodzic wezla, jesli istnieje
        self.key = None  # lewe poddrzewo
```

Przykład. Rozważmy drzewo, w którym klucze warstwami drzewa są umieszczone tak:

```
   5
  2  3
1  0  8 15
```

Niech X = $[3, 6, 4]$. W takim razie funkcja **maxim** powinna zwrócić wartość 8.