1. (a) Code inspection yields the following recursive equation:

$$\begin{aligned} T(2) &= 1 \\ T(n) &= 2 \cdot T(\tfrac{n}{2}) + 2 \text{ voor alle } n \geq 2 \end{aligned}$$

The two comparisons made in the return statements lead to constant cost 2.

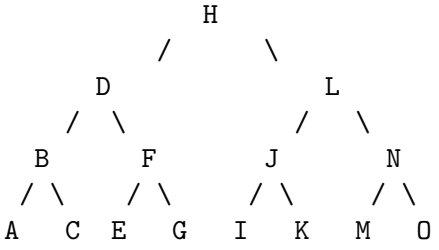
(b) We apply the Master theorem: $a = 2$, $b = 2$, $f(n) = 2$, and $E = \log a / \log b = 1$. Since $f(n) \in O(n^{E-\varepsilon})$ for e.g. $\varepsilon = \frac{1}{2}$, the first case applies, so

$$W(n) \in \Theta(n^E) \quad \text{so } W(n) \in \Theta(n)$$

2. (a) The minimum element is in the leaves of the maxheap. The leaves have indices from $\lfloor n/2 \rfloor$ to $n-1$, so $\lceil n/2 \rceil$ elements. Finding the minimum of these asks no more than $n/2$ comparisions. The maximum number is $E[0]$. So the algorithm:

```
def difheap(E,n):
k=n//2
minel=E[k]
for i in range (k+1,n):
minel=min(minel,E[i])
return E[0]-minel
```

(b) The tree (in fact a BST) is



If you traverse this tree in a pre–order way you encounter the letters in the order HDBACFEGLJIKNMO.

(a) Either you don't put object i in the backpack, so the remainder weight is $R(i-1, g)$, or you do put object i in it, and then the remainder weight $R(i-1, g-w_i)$. You have to choose the minimum of those two choices, so
$R(i, g) = min\{R(i-1, g), R(i-1, g-w_i)\}$.

(b) The algorithm to fill the matrix `R` containing the remainder weights (we assume the indices in `w` range from 1 to n) :

```
def backpack(w,G):

    n=len(w)
    R=[[0 for g in range(G+1)] for i in range(n+1)]

    for g in range(0,L+1):
        R[0,g]=g           # restgewicht gelijk aan g

    for i in range(1,n+1):
        for g in range(0,G+1):
            if (g-w[i])<0:
                R[i,g]=R[i-1,g]
            else:
                 R[i,g]=min(R[i-1,g],R[i-1,g-w[i]])

    return G-R[n,G];
```

The complexity of this algorithm is $\Theta(n^2)$.