برای ضرب ماتریس H_k در بردار ستونی V باید درای های هر سطر ماتریس را در تمام درایه های بردار ضرب کرده و با هم جمع کنیم تا درایه های ماتریس حاصل ضرب را به دست آوریم. در این مرحله واضح است که درایه های ستون اول ماتریس تنها در درایه اول بردار ضرب خواهند شد و درایه های ستون دوم ماتریس در درایه دوم بردار ضرب خواهند شد و این روند برای تمام ستون های ماتریس تکرار خواهد شد، از این نکته برای حل سوال به شیوه تقسیم و حل استفاده خواهیم کرد.

برای بردار V داریم : $V = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}$. نیمه بالایی بردار V را به صورت $V_t = \begin{bmatrix} v_1 \\ \vdots \\ v_{n/2} \end{bmatrix}$ و نیمه پایینی این بردار را به صورت $V_b = \begin{bmatrix} v_{n/2+1} \\ \vdots \\ v_n \end{bmatrix}$ تعریف می کنیم. ماتریس H_k به صورت بازگشتی و از روی H_{k-1} تعریف شده است که ماتریس را به چهار بلوک مشابه تبدیل می کند. برای ضرب ماتریس در بردار V کافی است نیمه بالایی ماتریس را در بردار ضرب کرده و در نیمه بالایی بردار حاصلضرب قرار دهیم و این کار را برای نیمه پایینی نیز تکرار کنیم. با توجه به توضیحات داریم :

$$H_k V = \begin{bmatrix} H_{k-1}V_t + H_{k-1}V_b \\ \\ H_{k-1}V_t - H_{k-1}V_b \end{bmatrix}$$

بدین ترتیب مسئله به دو مسئله کوچکتر که $H_{k-1}V_t$ و $H_{k-1}V_b$ هستند تبدیل خواهد شد و پس از پیدا کردن جواب آنها، با جمع ماتریس ها، جواب کلی را پیدا می کنیم.

HadamardTimesV(V, k)

Require: Vector V which will be multiplied with the corresponding Hadamard matrix
Ensure: Vector $M = H_k V$
  if k $= 0$ **then**
    return V
  else
    $M_1 = HadamardTimesV(V_t, k-1)$
    $M_2 = HadamardTimesV(V_b, k-1)$
    return $\begin{bmatrix} M_1 + M_2 \\ M_1 - M_2 \end{bmatrix}$

Listing 1: Python Implementation

```
def add_vectors(v1, v2, coefficient=1):
    return [v1[i] + (coefficient) * v2[i] for i in range(len(v1))]


def hadamard_times_v(v):
    n = len(v)
    if n == 1:
        return v
    else:
        first_half = hadamard_times_v(v[: n // 2])
        second_half = hadamard_times_v(v[(n - 1) // 2 + 1 :])
        top = add_vectors(first_half, second_half)
        bottom = add_vectors(first_half, second_half, coefficient=-1)
```

return top + bottom

در روند تقسیم و حل، ماتریس H_k با اندازه 2^k به دو ماتریس کوچکتر H_{k-1} با اندازه های 2^{k-1} تقسیم می شود و برای ترکیب جواب ها $2n$ جمع خواهیم داشت، بنابر این اگر $T(n)$ تابعی باشد که زمان اجرای این الگوریتم را نشان می دهم خواهیم داشت:

$$T(n) = 2\,T(n/2) + 2n$$

طبق قضیه اصلی، زمان اجرای الگوریتم $\theta(nlog(n))$ خواهد بود.