الگوریتم Rabin-Karp یک الگوریتم جستجوی رشته است که برای یافتن یک الگوی مشخص در یک متن از اجزای قابل مقایسه مانند اعداد یا حروف استفاده می‌کند. الگوریتم از تابع هش برای سرعت بخشیدن به فرآیند جستجو استفاده می‌کند.

حالا بیایید این الگوریتم را به گونه‌ای بسط دهیم که بتواند یک ماتریس الگوی m در m را در ماتریس n در n از حروف بیابد و الگو در ماتریس بزرگتر چرخانده نمی‌شود. برای این کار، می‌توانیم از یک تابع هش دیگری به نام تابع هش دوبعدی استفاده کنیم .

برای این الگوریتم، از تابع هش دوبعدی استفاده خواهیم کرد که برای هر زیرماتریس m در n در n یک مقدار هش ایجاد می‌کند و با مقدار هش الگو مقایسه می‌کند. اگر مقادیر هش برابر بودند، به طور دقیق تر بررسی می‌کنیم که آیا زیرماتریس و الگو با یکدیگر برابر هستند یا نه.

در ادامه یک الگوریتم Rabin-Karp چندبعدی به عنوان نمونه ارائه شده است:

```python
def hash_matrix(matrix):
    # تابع هش دوبعدی برای ماتریس
    hash_value = 0
    for i in range(len(matrix)):
        for j in range(len(matrix[i])):
            hash_value = (hash_value * 256 + ord(matrix[i][j])) % (10**9 + 7)
    return hash_value

def rabin_karp_2d(text, pattern):
    m, n = len(pattern), len(text)
    pattern_hash = hash_matrix(pattern)

    for i in range(n - m + 1):
        for j in range(n - m + 1):
            # محاسبه هش برای هر زیرماتریس m در n
            submatrix = [row[j:j+m] for row in text[i:i+m]]
```

```
            submatrix_hash = hash_matrix(submatrix)

            # مقایسه هش زیرماتریس با هش الگو
            if submatrix_hash == pattern_hash and submatrix == pattern:
                print(f"الگو یافته شد در مختصات: ({i}, {j})")

# مثال
matrix_text = [
    "abcde",
    "fghij",
    "klmno",
    "pqrst",
    "uvwxy"
]

matrix_pattern = [
    "ijk",
    "nop",
    "qrs"
]

rabin_karp_2d(matrix_text, matrix_pattern)
```

در این مثال، تابع `hash_matrix` برای محاسبه هش یک ماتریس و `rabin_karp_2d` برای اجرای الگوریتم Rabin-Karp چندبعدی استفاده شده‌اند.