REMEMBER Solution:

Xét các vị trí j của xâu ký tự từ vị trí 1 đến vị trí m. Với mỗi vị trí j, xét các xâu i lần lượt từ xâu 1 đến xâu n. Gọi $mask[i,j]$ là tập hợp các xâu có ký tự ở vị trí j giống xâu i.
Gọi x là trạng thái các xâu được nhớ (0- nếu đã tồn tại vị trị nhớ, 1- chưa tồn tại vị trí để nhớ); x là một số nguyên n bit.
Xét bit i của x có giá trị 1 (xâu i chưa có vị trí nhớ). Lần lượt xét các vị trí từ 1 đến m. Với mỗi vị trí có hai cách để biến xâu i thành xâu có vị trí nhớ:

- Cách 1: Đổi ký tự của i ở vị trí j thành ký tự khác, chi phí $a[i,j]$
- Cách 2: Giữ nguyên ký tự thứ j của xâu i và đổi tất cả các ký j ở trong tập $mask[i,j]$ thành ký tự khác. Để làm cho nó nhỏ nhất cần chi phí $c[i,j]$ bằng tổng các $a[i,j]$ trong tập $mask[i,j]$ trừ đi số lớn nhất (có thể chuẩn bị trước mảng $c[i,j]$ ngay từ khi đọc. Sau khi làm như vậy tất cả các xâu trong tập $mask[i,j]$ mà bit tương ứng trong x bằng 1 cũng được nhận diện.

Đặt $dp[x]$ là chi phí tối thiểu để xử lý x và i là bit 1 của x; .ta có:
dp[x]=min{dp[x^(1<<(i-1))]+a[i][j], dp[x & (x^mask[i][j])]+c[i][j]}

Đáp số là dp[(1<<n)-1]
Code minh họa:

```
#include <bits/stdc++.h>
#define oo 2000000000

using namespace std;

int n, m, a[25][25];
char s[25][25];
int nho[1500000], dp[1500000];
int mask[25][25], cp[25][25];

int calc(int x) {
   if (x==0) return 0;
   if (nho[x]) return dp[x];
   nho[x]=1; dp[x]=oo;
   int k=0; while ((x & (1<<k))==0) k++;
   for(int j=0;j<m;j++) {
      dp[x]=min(dp[x],calc(x^(1<<k))+a[k+1][j]);
      dp[x]=min(dp[x],calc(x & (x^mask[k+1][j]))+cp[k+1][j]);
   }
   return dp[x];
}


int main() {
   #ifndef ONLINE_JUDGE
   freopen("remember.inp","r",stdin);
   freopen("remember.out","w",stdout);
   #endif // ONLINE_JUDGE
   scanf("%d%d\n",&n,&m);
```

```
    for(int i=1;i<=n;i++) gets(s[i]);
    for(int i=1;i<=n;i++)
        for(int j=0;j<m;j++) scanf("%d",&a[i][j]);
    for(int i=1;i<=n;i++)
    for(int j=0;j<m;j++) {
        mask[i][j]=0; cp[i][j]=0;
        int cpmax=0;
        for(int k=1;k<=n;k++) if (s[i][j]==s[k][j]) {
            mask[i][j] |=(1<<(k-1));
            cp[i][j]+=a[k][j];
            cpmax=max(cpmax,a[k][j]);
        }
        cp[i][j]-=cpmax;
    }
    memset(nho,0,sizeof(nho));
    int ans=calc((1<<n)-1);
    printf("%d",ans);
}
```