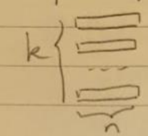
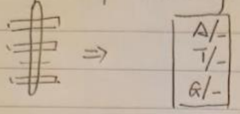
Multiple Sequence Alignment



n is the maximum length, k is how many sequences
seq[a][b] means ath seq's bth letter.

① Since I have k seqs, with or without gap, there are 2^k-1 situations in one pair of searching.



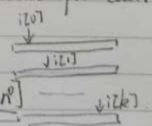
: 2^k-1 situations

implement:	for t in 1 to 2^k-1: # get rid of all gaps
$t_{(10)} \to t_{(2)}$ and store all digits in r[k]
example: t = 7 means no gap, all are letters, when k=3
$t_{(10)} = 7 \Rightarrow t_{(2)} = 111 \Rightarrow r[\,] = [1, 1, 1]$.

② In the slide, it use D[ ][ ][ ] to represent the final score for 3 sequences. But since we have k seqs, we cannot create a k-dimentional array each time. My solution is to change D[ ][ ][ ] from k dimentions to 1 dimention

Theory: If I have a number (decimal) 123
I can use a 3 dims array to save it $\Rightarrow$ D[ ][ ][ ] = [1][2][3]
While it's also as $123 = 1\times10^2 + 2\times10^1 + 3\times10^0$
$\Rightarrow D[\,] = 1\times10^2 + 2\times10^1 + 3\times10^0$.

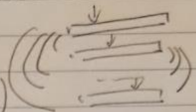
So I can also use this method. k seqs means k dims
plus: Since I cannot use i, j, k ... to represent for each dim,
I use i[0 ... k-1] for k dims
$\Rightarrow$ D[i[0]][i[1]][i[2]] ... [i[k-1]]



So D[i[0]] ... [i[k-1]] $= D[i[0]\cdot n^{k-1} + i[1]\cdot k^{k-2} + \cdots + i[k-1]\cdot n^0]$

③ Calculate score in t's score's situation

Since r[0 ~ k] represent that r[i] = 0/1 means ith seq is whether a gap or not.



```
for j = 0 ~ k-1
    for l = j+1 ~ k:
        if( r[j] == 0 and r[l] == 1)
            sp(i[0]·n^{k-1} + ... + i[k-1]·n^0) += gap
        if (r[j] == 1 and r[l] == 1)
            sp(                    ) += sub(seq[j][i[j]], seq[l][i[l]]
        if       == 0               == 0
                                      += gap
        if       == 1               == 0
                                      += gap
```

\# plus: in $sp(i[0]\cdot n^{k-1} + \dots + i[k-1]\cdot n^0)$, if this situation is with a gap, then if $r[j] == 0$, $i[j]\cdot n^{k-1-j}$ should be replaced by $0 \cdot n^{k-1-j}$,

④ When working on jth and lth sequences, we should scan all patterns

```
for i[j] = 0 to n-1
    for i[k] = 0 to n-1.
```

⇒ ~~for~~ In total

```
for t = 1 ~ 2^k - 1
    for j in 0 ~ k-1
        for l in j+1 ~ k.
            for i[j] in 0 ~ n-1
                for i[l] in 0 ~ n-1:
D(i[0]·n^{k-1} + ...) = ~~min(D[i[0]n^{k-1} ...]: all the situations~~
                        min(itself, situation under t + sp( )).
```

sp is to count as ③.