CSE 566 Spring 2023

Wavefront Algorithm

Instructor: Mingfu Shao

The Wavefront Algorithm

- For the edit distance problem (Myers, 1986)
  - Runs in O(nd) time, where d is the edit distance, n >= m
  - Runs in O(d) extra space
- Extension to affine gap cost
  - Runs in O(ns) time, where s is the optimal cost
  - Runs in O(s) extra space
  - https://doi.org/10.1093/bioinformatics/btaa777 (2021)
  - https://doi.org/10.1093/bioinformatics/btad074 (2023)

Diagonals of The OPT Table

1 2 n-1 n

0

	A	C	C	G	T	G	C	T
A	0	1	2	3	4	5	6	7
C	1	0	1	2	3	4	5	6
G	2	1	1	1	2	3	4	5
G	3	2	2	1	2	2	3	4
A	4	3	3	2	2	3	3	4
T	5	4	4	3	2	3	4	3

-1 -2 -m+1 -m

m+n+1 diagonals

- Table for edit distance.
- Diagonal k: k = j - i;
- If OPT(i, j) = d, then we have -d <= k <= d.
- Along any diagonal, the values are non-decreasing. (HW)

Wavefront Algorithm: An Example

$\boxed{d = 0}$

	A	C	C	G	T	G	C	T
A	0	1	2	3	4	5	6	7
C	1	0	1	2	3	4	5	6
G	2	1	1	1	2	3	4	5
G	3	2	2	1	2	2	3	4
A	4	3	3	2	2	3	3	4
T	5	4	4	3	2	3	4	3

Wavefront Algorithm: An Example

$\boxed{d = 1}$

	A	C	C	G	T	G	C	T
A	0	1	2	3	4	5	6	7
C	1	0	1	2	3	4	5	6
G	2	**1**	1	**1**	2	3	4	5
G	3	2	2	**1**	2	2	3	4
A	4	3	3	2	2	3	3	4
T	5	4	4	3	2	3	4	3

Wavefront Algorithm: An Example

d = 2

	A	C	C	G	T	G	C	T
A	0	1	2	3	4	5	6	7
C	1	0	1	2	3	4	5	6
G	2	1	1	1	2	3	4	5
G	3	**2**	2	1	**2**	**2**	3	4
A	4	3	3	2	**2**	3	3	4
T	5	4	4	3	**2**	3	4	3

Wavefront Algorithm: An Example

d = 3

	A	C	C	G	T	G	C	T
A	0	1	2	3	4	5	6	7
C	1	0	1	2	3	4	5	6
G	2	1	1	1	2	3	4	5
G	3	2	2	1	2	2	**3**	4
A	4	**3**	3	2	2	**3**	3	4
T	5	4	4	**3**	2	**3**	4	**3**

Key Definition: Wavefront

	A	C	C	G	T	G	C	T	
A	0	1	2	3	4	5	6	7	1
C	1	0	1	2	3	4	5	6	2
G	2	1	1	1	2	3	4	5	3
G	3	2	2	1	2	2	3	4	4
A	4	3	3	2	2	3	3	4	5
T	5	4	4	3	2	3	4	3	6

$W_3[1] = 5$

$W_2[2] = 4$, $W_3[-3] = 5$

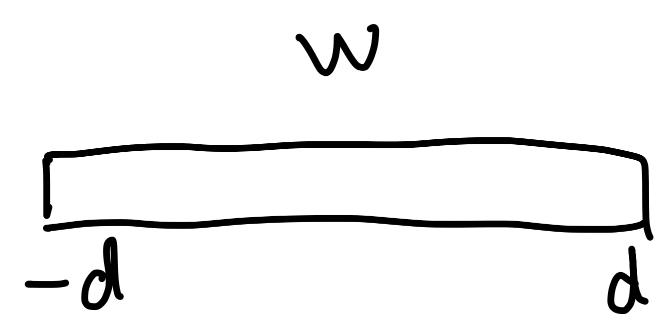
- $W_d[k]$ stores the furthest row on diagonal-k such that the OPT of that entry is d.
- $W_d[k]$ stores the largest i such that OPT(i, i + k) = d.
- Feasible range: -d <= k <= d.
- If k < -d or k > d, then we define $W_d[k] = -\infty$.

$j = i + k \Leftrightarrow k = j - i$

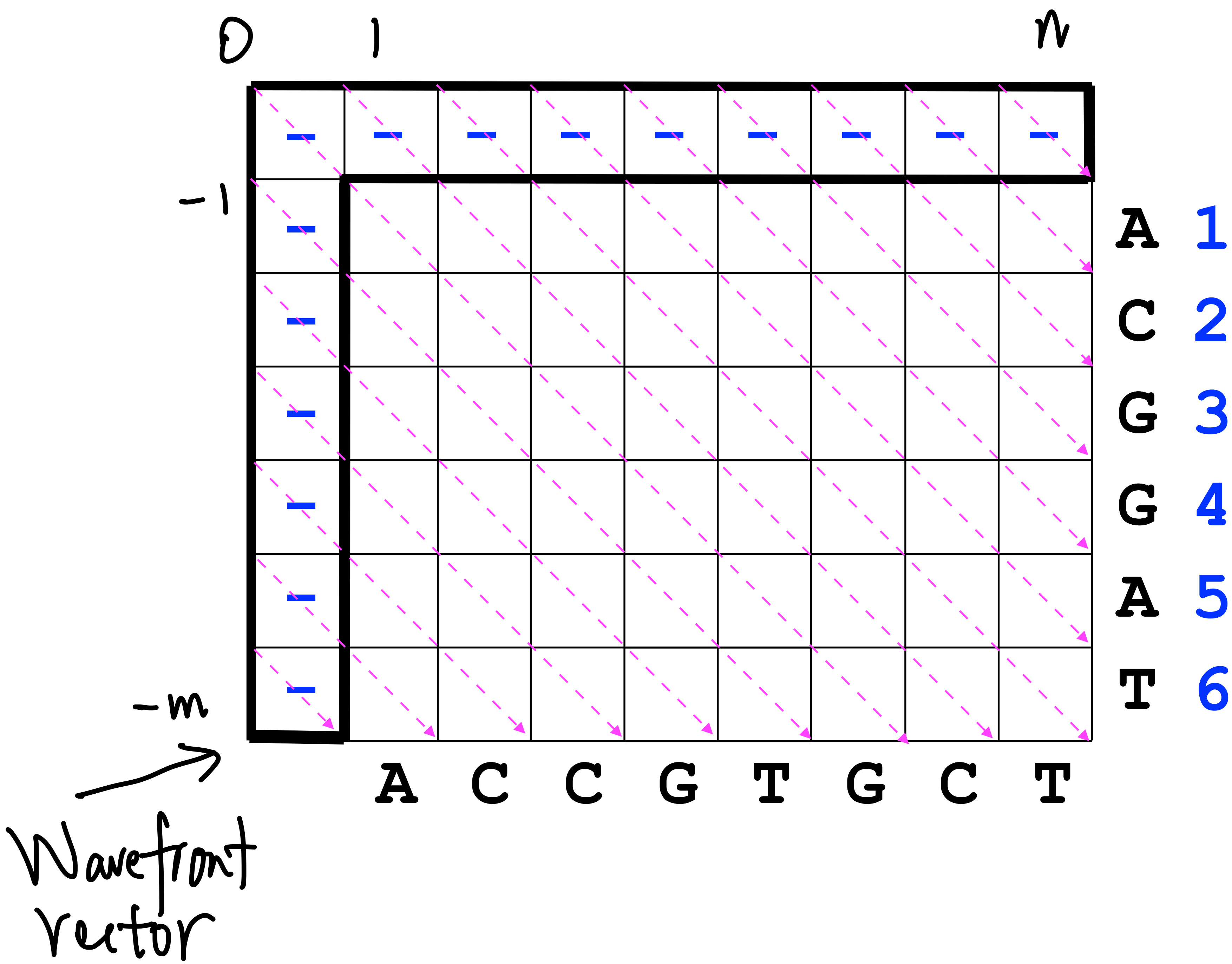
Framework of Wavefront Algorithm

- Idea: gradually increase d; for each d, calculate the wavefront vector W_d, until the (m,n) position is reached.

```
algorithm wavefront(S[1..m], T[1..n])
   init wavefront W for d = 0
   for d = 1 to max{m, n}
        for k = -d to d
             calculate W[k];
        end for
        if W[n-m] = m: return d;
    end for
end algorithm
```

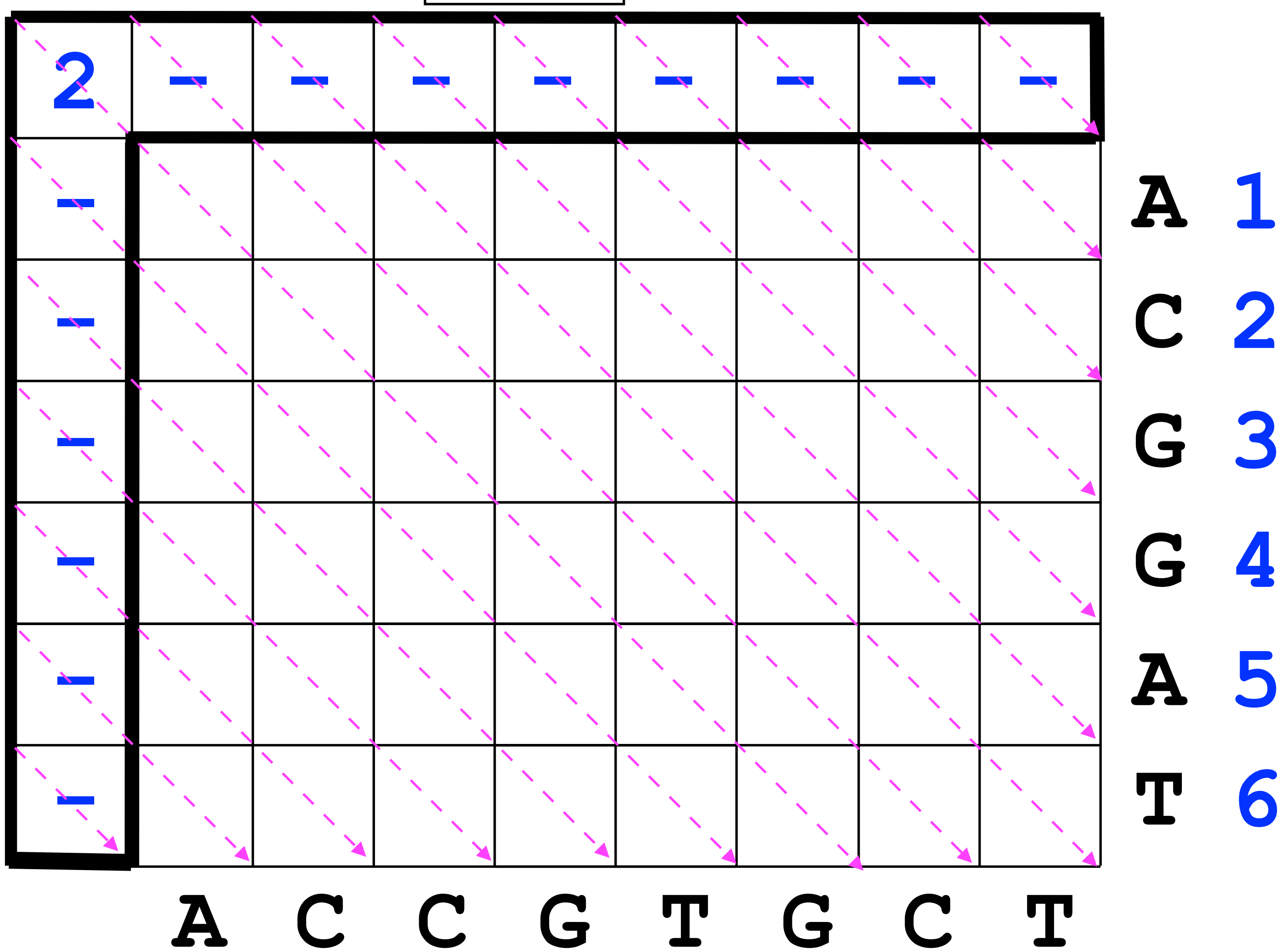


Wavefront Algorithm: An Example



Wavefront Algorithm: An Example

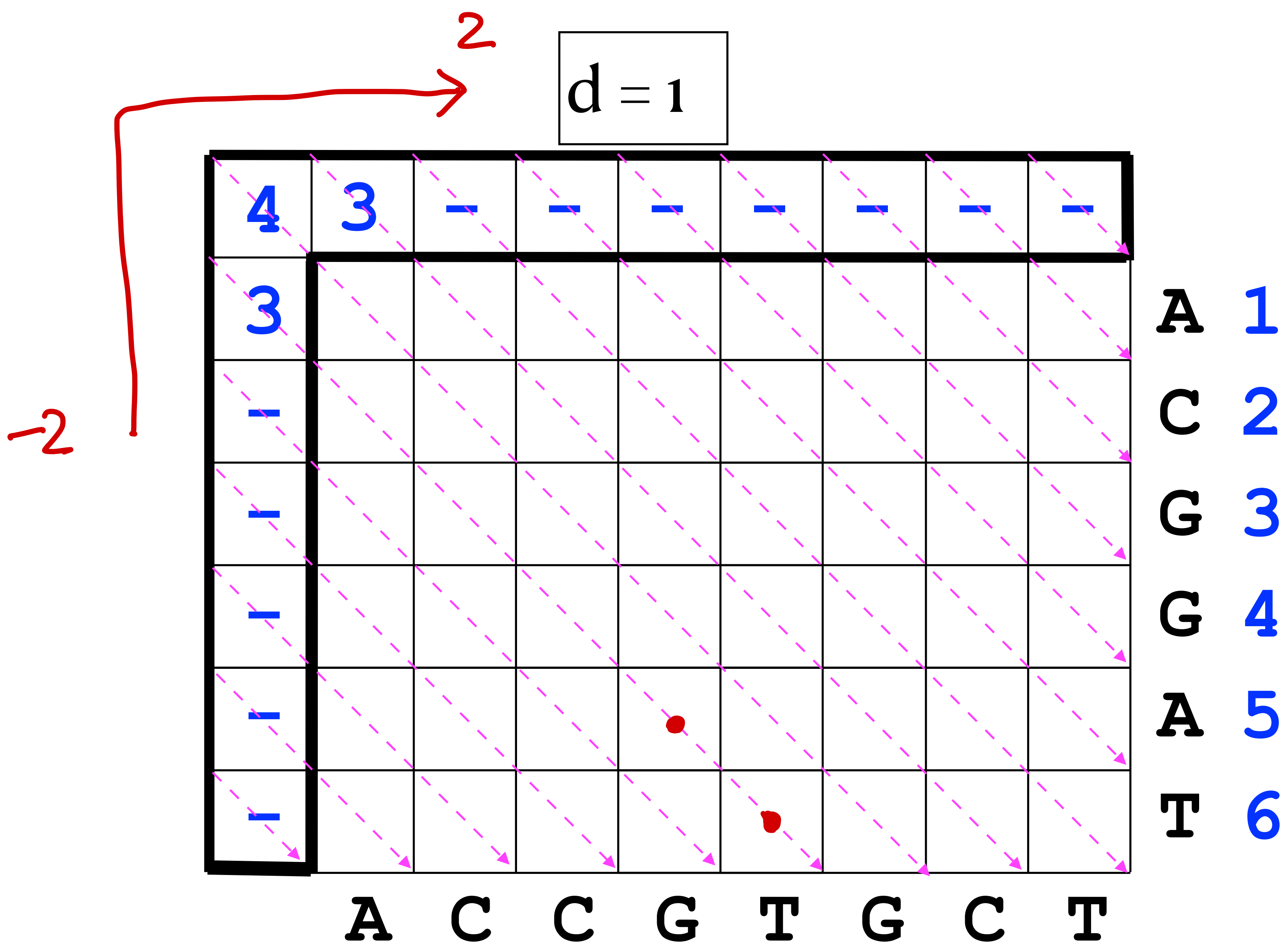
d = 0



W

$W_0[0] = 2$

Wavefront Algorithm: An Example



$d=2$, $k=-1$

step 1:

$W[-1] = \max\{3+1, 4+1\}$

$= 5$,

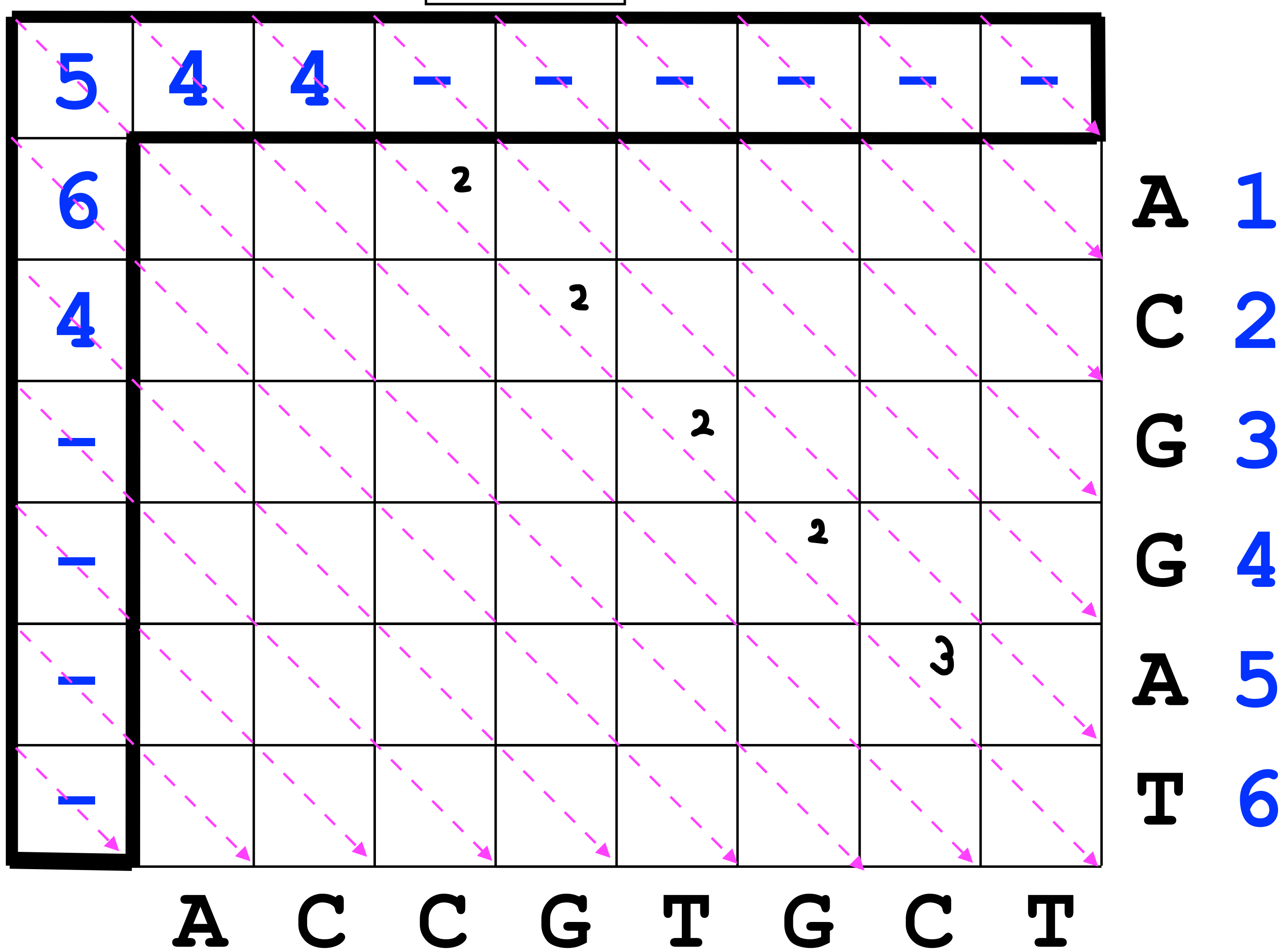
step 2:

$W[-1] = 6$

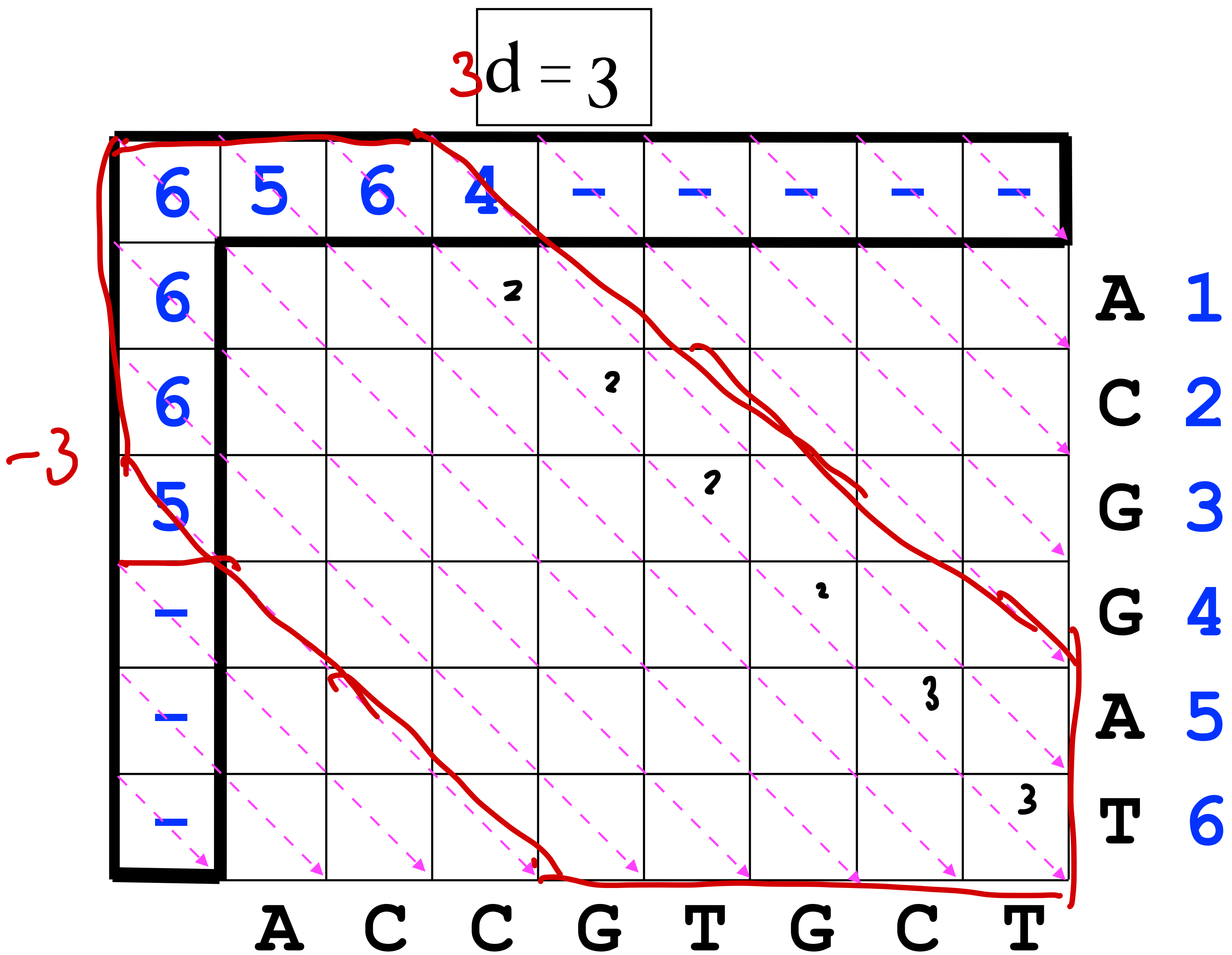
Wavefront Algorithm: An Example

$n - m = 2$

$d = 2$



Wavefront Algorithm: An Example



Initialize Wavefront

- When $d = 0$, only need to consider $k = 0$

```
i = 1;
while(S[i] == T[i]) i++;
W[0] = i - 1;
```

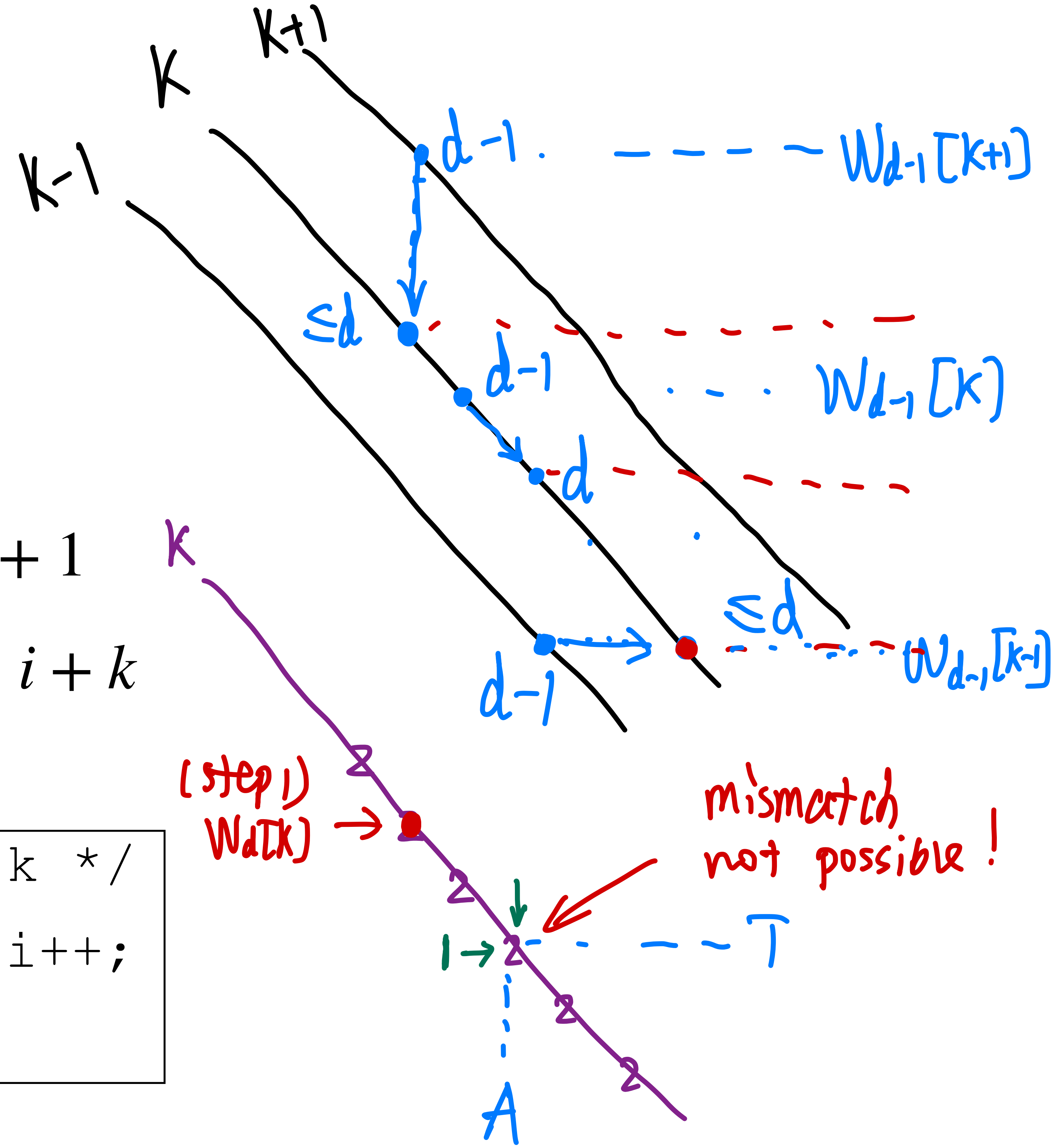
Calculate Next Wavefront

- Task: calculate W_d from W_{d-1}
- Step 1: update

$$W_d[k] = \max \begin{cases} W_{d-1}[k-1] \\ W_{d-1}[k] + 1 \\ W_{d-1}[k+1] + 1 \end{cases}$$

  - OPT(i, j) = d, $i = W_d[k], j = i + k$
- Step 2: extend

```
i = W[k]+1;/* j = i + k */
while(S[i] == T[i+k]) i++;
W[k] = i - 1;
```



Pseudo-Code

W_{d-1} W_d

$\underline{V}$, W

```
algorithm wavefront(S[1..m], T[1..n])
    i = 1; while(S[i] == T[i]) i++; V[0] = i - 1;
    for d = 1 to max{m, n}
        for k = -d to d
            W[k] = max{V[k-1],V[k]+1,V[k+1]+1};
            i = W[k]+1; /* j = i + k */
            while(S[i] == T[i+k]) i++;
            W[k] = i - 1;
        end for
        if W[n-m] = m: return d;
        V = W;
    end for
end algorithm
```

W_d

Analysis of Wavefront Algorithm

- Correctness
- Time complexity:
    - All updates: $O(d^2)$ d: edit distance
    - All extending: $O(nd)$, as there are $O(nd)$ entries in diagonals -d to d, and each entry will be compared at most once.
- Space complexity: $O(d)$ extra space.