**Computer Algorithm Homework #1**

**<Sequence Alignment>**

**Class Number : 061**

**201524582 정희석**

**1. Explain how to fill the matrix**

**Answer of fill the matrix** : 염기 서열과 관련된 길이가 7인 sequence1(ATCCTAG)와 길이가 5인 sequence2(TAGGT)를 설정하여 입력 파일에 넣은 뒤 각 sequence의 맨 뒤에 갭(-)을 추가하여 2개의 문장으로 만든 후 sequence1의 길이 X sequence2의 길이의 Matrix를 만들어서 모든 값을 0으로 초기화 하고 가장 오른쪽 하단을 기준으로 왼쪽으로 이동하면서 2(penalty)씩 더해서 초기화, 오른쪽 하단을 기준으로 위로 이동하면서 2(penalty)씩 더해서 초기화 하였다. 그 이후 matrix내부의 값을 해당 부분의 오른쪽 값 +2, 아래쪽 값 +2, 대각선 아래 값 + (해당 부분의 sequence 문자를 비교해서 같으면 +0, 다르면 +1) 이 세 개의 값 중 가장 작은 값을 넣도록 하여서 matrix를 채웠다.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **T** | **A** | **C** | **G** | **T** | **-** |  |  | **T** | **A** | **C** | **G** | **T** | **-** |
| **A** | **7** | **7** | **9** | **11** | **12** | **14** | **A** | **7** | **7** | **9** | **11** | **12** | **14** |
| **T** | **5** | **6** | **7** | **9** | **10** | **12** | **T** | **5** | **6** | **7** | **9** | **10** | **12** |
| **C** | **5** | **5** | **5** | **7** | **8** | **10** | **C** | **5** | **5** | **5** | **7** | **8** | **10** |
| **C** | **5** | **4** | **4** | **5** | **6** | **8** | **C** | **5** | **4** | **4** | **5** | **6** | **8** |
| **T** | **4** | **4** | **3** | **4** | **4** | **6** | **T** | **4** | **4** | **3** | **4** | **4** | **6** |
| **A** | **6** | **4** | **3** | **2** | **3** | **4** | **A** | **6** | **4** | **3** | **2** | **3** | **4** |
| **G** | **8** | **6** | **4** | **2** | **1** | **2** | **G** | **8** | **6** | **4** | **2** | **1** | **2** |
| **-** | **10** | **8** | **6** | **4** | **2** | **0** | **-** | **10** | **8** | **6** | **4** | **2** | **0** |

파란 글씨 : 제일 먼저 초기화 빨간 글씨 : 최단 거리

**Answer of find shortest path** : matrix의 0,0 지점이 최저 페널티이므로 (0,0) 부터 해당 위치의 값이 아래쪽 값보다 2 크면 아래로 내려가도록 했고 그렇지 않으면 대각선 아래로 가도록 했다.(오른 쪽 이동은 고려 안함 -> 이유 : 오른쪽 방향의 값들 보다 아래쪽 방향의 값들이 더 작아지므로) 이동 할 때 마다 아래 방향으로 이동하면 출력될 sequence의 문자열에 gap(-)을 추가하고 대각선 아래 방향으로 이동하면 문자열에 sequence2의 해당 하는 글자를 추가한다. 이 과정을 반복하면 최단 거리의 alignment된 sequence가 구해진다.

**2. Pseudo Code**

**#<sequence읽어오기>**

#File Open with read & write mode

#>sequence1 = file.readline() // file의 첫 번째 문장

#>sequence2 = file.readline() // file의 두 번째 문장

#>readfile.close() // read파일 닫기

#>matrix = matrix[length of sequence1][length of sequence2] // 8x6 matrix를 만듬

**#<matrix 초기설정>**

#> matrix(-1,-1) = 0 //오른쪽 맨 아래를 0으로 초기화 하고

#> matrix(l,-1) += 2 //왼쪽으로 갈수록 2씩 증가하여 추가

#> matrix(-1,m) += 2 //위쪽으로 갈수록 2씩 증가하여 추가

**#<matrix 채우기>**

#> martix의 [-2,-2](=(6,5))부터 왼쪽으로 이동하면서 추가(m--)

#> matrix(l,m) = min(matrix(l,m+1) + 2{=오른쪽 페널티+2(gap)}, matrix(l+1,m)+2 {=아래쪽 페널티 + 2(gap)} , matrix(l+1,m+1)+0 or 1{=대각선 페널티 + 문자 비교 페널티(같으면 0, 다르면 1)})

# 한 줄 다 채웠으면 윗줄 맨 오른쪽으로 이동 및 반복 (l--, m = -2 or 5)

#> matrix의 (0,0)까지 채웠으면 반복문 탈출

**#<최단 거리(최저 페널티) 찾기>**

#> matrix를 채울 때부터 가장 적은 페널티로 계산을 하였으므로 matrix의 (0,0)값이 최소 페널티가 되는 것을 알 수 있다.

#> while(l != length of sequence1 & m != length of sequence2) // 반복문을 통해 아래의 과정을 matrix의 성분이 오른쪽 맨 아래 성분이 될 때 까지 실행

#> if(matrix(l,m) == matrix(l,m-1)+2) m++, sequenceResult += ‘-‘//이를 기준으로 비교문을 통해 아래쪽 성분보다 2 크면 아래로 진행 & 출력 sequence에 gap ’-‘을 추가

#> else if(matrix(l,m) == matrix(l-1,m-1)+2) pass //오른쪽 성분보다 2 큰 경우에는 무시함

#> else sequenceResult += sequence2[m], l++, m++ // 오른쪽, 아래쪽 모두 2차이가 나지 않는다면 대각선 아래로 진행 & 출력 sequence에 sequence2의 해당 문자를 추가

# 위의 과정을 반복

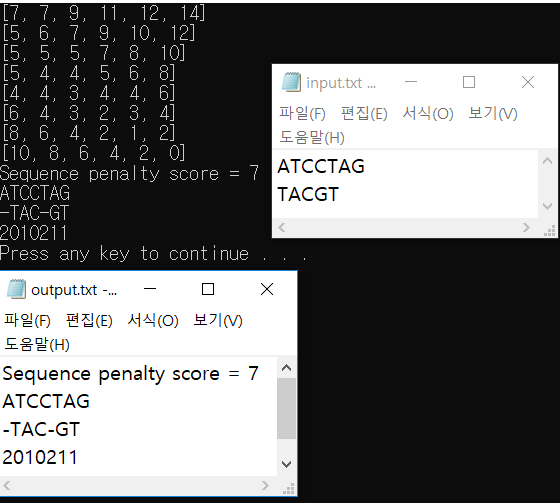
**#<file과 화면에 출력>**

#> print sequence1 and file.write(sequence1) //기준 sequence를 화면과 파일에 출력

#> print sequenceResult and file.write(sequenceResult) //패널티가 가장 적은 순서대로 정렬된 sequence를 화면과 파일에 출력

#> writefile.close() // 작성한 파일을 종료함.

3. 실행결과



4. 느낀 점

이번 과제를 막상 받았을 때 어떤 원리인지 이해가 안돼서 막막했는데 막상 작성하니까 원하는 대로 결과가 나와서 다행이라고 생각이 들었습니다. 그리고 C++과 JAVA만 사용하다가 잘 안 쓰던 Python을 사용하여 알고리즘을 짜니 처음에는 어떻게 시작해야 할 지 막막했지만(엄청 오랜만의 python사용) 역시 작성하다 보니 오히려 더 편하게 짤 수 있었다 라는 생각이 들었습니다. 무엇보다 생각하는 대로 구현이 되니 좋았습니다.