1. **소스코드 및 주석**

**1-1.\_1번문제**

#include <string>

#include <algorithm>

#include <iostream>

using namespace std;

class Solution {

public:

int dp[502][502]{};

int minDistance(string str1, string str2) {

for (int i = 0; i < str1.size(); i++)

dp[i + 1][0] = i + 1;

for (int j = 0; j < str2.size(); j++)

dp[0][j + 1] = j + 1;

for (int i = 1; i <= str1.size(); i++)

{

for (int j = 1; j <= str2.size(); j++)

{

if (str1[i - 1] == str2[j - 1])

dp[i][j] = dp[i - 1][j - 1];

else

dp[i][j] = min(min(dp[i - 1][j], dp[i][j - 1]), dp[i - 1][j - 1]) + 1;

}

}

return dp[str1.size()][str2.size()];

}

void printDPTable(string str1, string str2) {

for (int i = 0; i <= str1.size(); i++) {

for (int j = 0; j <= str2.size(); j++) {

printf("%3d ", dp[i][j]);

}

printf("\n");

}

printf("\n");

}

};

int main(void) {

Solution solution;

string str1;

printf("str1 : ");

getline(cin, str1);

string str2;

printf("str2 : ");

getline(cin, str2);

int result = solution.minDistance(str1, str2);

solution.printDPTable(str1, str2);

printf("최소 edit distance: %d\n", result);

return 0;

}

**1-2. 2번문제**

#include <string>

#include <algorithm>

#include <iostream>

using namespace std;

string dictionaryWord[3] = { "definitely", "defiantly", "define" };

class Solution {

public:

int dp[502][502]{};

int minDistance(string str1, string str2) {

for (int i = 0; i < str1.size(); i++)

dp[i + 1][0] = i + 1;

for (int j = 0; j < str2.size(); j++)

dp[0][j + 1] = j + 1;

for (int i = 1; i <= str1.size(); i++)

{

for (int j = 1; j <= str2.size(); j++)

{

if (str1[i - 1] == str2[j - 1])

dp[i][j] = dp[i - 1][j - 1];

else

dp[i][j] = min(min(dp[i - 1][j], dp[i][j - 1]), dp[i - 1][j - 1]) + 1;

}

}

return dp[str1.size()][str2.size()];

}

void printDPTable(string str1, string str2) {

for (int i = 0; i <= str1.size(); i++) {

for (int j = 0; j <= str2.size(); j++) {

printf("%3d ", dp[i][j]);

}

printf("\n");

}

printf("\n");

}

};

int main(void) {

printf("사전 단어 : [%s, %s, %s]\n\n", dictionaryWord[0].c\_str(), dictionaryWord[1].c\_str(), dictionaryWord[2].c\_str());

Solution solution;

string str;

printf("str1 : ");

getline(cin, str);

int distance[3] = { 0, };

string str\_min = "";

int distance\_min = 0;

for(int i = 0; i < 3; i++) {

distance[i] = solution.minDistance(str, dictionaryWord[i]);

if (i == 0) {

str\_min = dictionaryWord[i];

distance\_min = distance[i];

}

else if(i!=0&& distance[i]< distance[i-1]) {

str\_min = dictionaryWord[i];

distance\_min = distance[i];

}

}

printf("추천: \"%s\" (edit distance %d)", str\_min.c\_str(), distance\_min);

return 0;

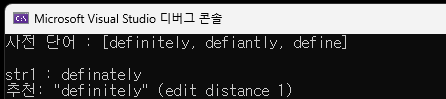
}

1. **실행화면 캡처**

**2-1.\_1번문제**



**2-2. 2번문제**



1. **고찰**

* 인간이 편집거리 문제를 볼때, 유사도를 보고 같은 순서의 문자는 그대로 둔다음 나머지 문자는 치환, 삭제, 삽입 중 선택하는 것과 달리 알고리즘에서는 모든 경우의 수를 계산함을 깨달음
* DP 테이블을 2차원 테이블로 만들기때문에 시간복잡도는 다음과 같을 것으로 예상함
  + Str1의 길이 : n
  + Str2의 길이 : m
  + DP 테이블의 길이 : (n-1) \* (m-1)
    - 3가지 경우의 수 중 최소의 값 선택 : O(1)
  + 총 시간 복잡도 : O(n\*m)
* Str1와 str2의 앞부분이 같은만큼 전부 0으로 계산한다음 기억한다면 조금더 효율적일 것이라고 생각합니다
* 아니면, 벡터를 이용해서 유사도를 찾아서 어느정도 유사한 단어들만 고른다음 그단어들과의 edit distance를 각각 계산하면 아마도 좀 더 효율적인 검색이 되지않을까 생각합니다