① # Minimum Edit Distance 511 31-1-06

Transforming string s1 = "happy"
into s2 = "harpie"

editing operations
- delete next
- insert char
- cursor right
  - only make left-to-right progress

progress measured by i, j
current string is s2 from beginning to position j
+ s1 from position i to end.

at start $i=0, j=0 \Rightarrow$ current = "" + "happy"

delete next $\Rightarrow$ "appy" = "" + "appy" ∴ i+=1
cursor right $\Rightarrow$ "h" + "appy" ∴ i+=1, j+=1
insert 'h' $\Rightarrow$ "h" + "happy" ∴ j+=1
~~only h may be inserted~~

"h" + "happy", i=1 j=0
- delete next $\Rightarrow$ "h" + "appy" i=1 j=1
- cursor right $\Rightarrow$ "hh" + "appy" ∴ not allowed
- insert 'a' $\Rightarrow$ "ha" + "happy" i=0 j=2
- insert 'b' $\Rightarrow$ "hb" + "happy" ∴ not allowed

Can only
- delete next if there is a next : i < len(s1)
- cursor right if next is correct : s1[i]==s2[j] & i<len & j<len
- insert ~~s1~~ s2[j] : j<len

② 31-1-06

full example

```
i :  0   1   2   3   4   5
s1 : | h | a | p | p | y |        len(s1) = 5

j :  0   1   2   3   4   5   6
s2 : | h | a | r | p | i | e |    len(s2) = 6
```

---

i=0, j=0 : "" + "happy"
- poss : delete next "" + "appy"
- ~~eor~~ : cursor rt "h" + "appy" ← choose this one for no particular reason
- : ins 'h' "h" + "happy"

---

i=1, j=1 : "h" + "appy"
- poss: del nxt : "h" + "ppy"
- cursor rt : "ha" + "ppy" ← choose this again
- ins 'a' : "ha" + "appy"

---

i=2, j=2 : "ha" + "ppy"
- poss del nxt : "ha" + "py" ← choose this one
- curs rt Not allowed
- ins 'r' : "har" + "ppy"

---

i=3, j=2 "ha" + "py"
- poss del nxt : "ha" + "y"
- curs rt : Not allowed
- ins 'r' : "har" + "py" ← choose this

---

i=3, j=3 "har" + "py"

⋮ you can see what happens

---

i=4, j=6 : "harpie" + "y"
- poss del nxt : "harpie" + "" ← must do this
- curs rt : Not allowed
- ins ... : Not allowed

---

i=5, j=6 : "harpie" + "". i = len(s1), j = len(s2)

∴ finished.

③ 31-1-06

So, the recursive function is:

int MED(int i, int j) ← could make s1, s2 parameters too, but to reduce writing will consider them global

```
{ ~~int~~ int poss1=∞ , poss2=∞ , poss3=∞ ;
   if (i<s1.length()) //delete next is allowed
      poss1 = MED(i+1, j) +1;
   if (i<s1.length() && j<s2.length() && s1[i]==s2[j])
      poss2= MED(i+1, j+1);
   if (j<s2.len()) //inserting s2[j] is allowed
      poss3= MED(i, j+1)+1;
   return min(poss1, poss2, poss3); }
```

Why +1 in two cases?
We are computing the number of changes required. Moving the cursor is not a change.

Dynamic Programming Solution
Notice that computing MED(i, j) makes use of <u>next</u> values, not previous ones
∴ must complete table <u>backwards</u>

Seed Value:
MED[s1.length()][s2.length()]=0;
because when you have finished, there are no changes left to make.

④ 31-1-06

```
cout << "\nOperations:\n";
int i=0, j=0;
while (i<len1 || j<len2)
{ if (i<len1 && MED[i+1][j]==MED[i][j]-1)
  { cout << "  delete next (" << s1[i] << ")\n";
    i+=1; }
  else if (j<len2 && MED[i][j+1]==MED[i][j]-1)
  { cout << "  insert '" << s2[j] << "'\n";
    j+=1; }
  else
  { cout << "  cursor right\n";
    i+=1;
    j+=1; } } }
```

```
Enter s1: happy
Enter s2: harpie

 5  6  7  6  7  6  5
 6  5  6  5  6  5  4
 7  6  5  4  5  4  3
 6  5  4  3  4  3  2
 7  6  5  4  3  2  1
 6  5  4  3  2  1  0

Operations:
  cursor right
  cursor right
  delete next (p)
  insert 'r'
  cursor right
  delete next (y)
  insert 'i'
  insert 'e'
```

⑤ 31-1-06

```
#include <iostream>
#include <string>

int MED[10][10];

void main(void)
{ string s1, s2;
  cout << "Enter s1: ";
  cin >> s1;
  cout << "Enter s2: ";
  cin >> s2;
  int len1=s1.length(), len2=s2.length();
  MED[len1][len2]=0;
  for (int i=len1; i>=0; i-=1)
    for (int j=len2; j>=0; j-=1)
    { if (i==len1 && j==len2)
        continue;
      int best=999999;
      if (i<len1)
      { int poss=MED[i+1][j]+1;
        if (poss<best) best=poss; }
      if (i<len1 && j<len2 && s1[i]==s2[j])
      { int poss=MED[i+1][j+1];
        if (poss<best) best=poss; }
      if (j<len2)
      { int poss=MED[i][j+1]+1;
        if (poss<best) best=poss; }
      MED[i][j]=best; }

  for (int i=0; i<=len1; i+=1)
  { cout << "\n";
    for (int j=0; j<=len2; j+=1)
      cout << " " << MED[i][j]; }
  cout << "\n"; }
```

```
Enter s1: happy
Enter s2: harpie

 5 6 7 6 7 6 5
 6 5 6 5 6 5 4
 7 6 5 4 5 4 3
 6 5 4 3 4 3 2
 7 6 5 4 3 2 1
 6 5 4 3 2 1 0
```