class Solution {

public:

int minCut(string s) {

int sSize = s.size();

int\* minCut = new int[sSize];

minCut[0] = 0;

int i;

for (i = 1; i < sSize; ++i) {

//try to find palindrome in s with a[i];

int j;

for (j = 0; j < i; ++j) {

if (s[j] == s[i]) {

int r;

for (r = 0; r < (i - j)/2; ++r) {

if (s[j + r] != s[i - r]) {

break;

}

}

if ( r == (i - j)/2) {

break;

}

}

}

if (j == 0) {

minCut[i] = 0;

} else {

minCut[i] = minCut[j - 1] + 1;

}

}

int iTemp = minCut[i - 1];

delete[] minCut;

return iTemp;

}

};

class Solution {

public:

int minCut(string s) {

int sSize = s.size();

int\* minCut = new int[sSize];

minCut[0] = 0;

int\* paliMatrix = new int[sSize \* sSize];

paliMatrix[0] = 0;

int i;

for (i = 1; i < sSize; ++i) {

//try to find palindrome in s with a[i];

int j;

for (j = i; j >= 0; --j) {

if (j == i) {

paliMatrix[i \* sSize + j] = 0;

}

if ((j == i - 1) && s[j] == s[i]) {

paliMatrix[i \* sSize + j] = 0;

}

else {

paliMatrix[i \* sSize + j] = 1;

}

if ((j < i - 1) && s[j] == s[i] && paliMatrix[(i - 1) \* sSize + j + 1] == 0) {

paliMatrix[i \* sSize + j] = 0;

}

else {

paliMatrix[i \* sSize + j] = 1;

}

}

for (j = 0; j < i; ++j) {

if (paliMatrix[i \* sSize + j] == 0) {

break;

}

}

if (j == 0) {

minCut[i] = 0;

}

else {

minCut[i] = minCut[j - 1] + 1;

}

}

int iTemp = minCut[i - 1];

delete[] minCut;

delete[] paliMatrix;

return iTemp;

}

};

int minCut(string s) {

int sSize = s.size();

int\* minCut = new int[sSize];

minCut[0] = 0;

int\* paliMatrix = new int[sSize \* sSize];

paliMatrix[0] = 0;

int i;

for (i = 1; i < sSize; ++i) {

//try to find palindrome in s with a[i];

int j;

for (j = i; j >= 0; --j) {

int index = i \* sSize + j;

paliMatrix[index] = 1;

if (j == i) {

paliMatrix[index] = 0;

}

if ((j == i - 1) && s[j] == s[i]) {

paliMatrix[index] = 0;

}

if ((j < i - 1) && s[j] == s[i] && paliMatrix[(i - 1) \* sSize + j + 1] == 0) {

paliMatrix[index] = 0;

}

}

for (j = 0; j < i; ++j) {

if (paliMatrix[i \* sSize + j] == 0) {

break;

}

}

if (j == 0) {

minCut[i] = 0;

}

else {

minCut[i] = minCut[j - 1] + 1;

}

}

int iTemp = minCut[i - 1];

delete[] minCut;

delete[] paliMatrix;

return iTemp;

}

int minCut(string s) {

int sSize = s.size();

int\* minCut = new int[sSize];

minCut[0] = 0;

int\* paliMatrix = new int[sSize \* sSize];

paliMatrix[0] = 0;

int i;

for (i = 1; i < sSize; ++i) {

//try to find palindrome in s with a[i];

int j;

for (j = i; j >= 0; --j) {

int index = i \* sSize + j;

paliMatrix[index] = 1;

if (j == i) {

paliMatrix[index] = 0;

}

if ((j == i - 1) && s[j] == s[i]) {

paliMatrix[index] = 0;

}

if ((j < i - 1) && s[j] == s[i] && paliMatrix[(i - 1) \* sSize + j + 1] == 0) {

paliMatrix[index] = 0;

}

}

int iTemp = INT\_MAX;

for (j = 0; j <= i; ++j) {

if (paliMatrix[i \* sSize + j] == 0) {

if (j == 0) {

iTemp = 0;

}

else {

iTemp = iTemp < (minCut[j - 1] + 1) ? iTemp : (minCut[j - 1] + 1);

}

}

}

minCut[i] = iTemp;

}

int cutNumber = minCut[i - 1];

delete[] minCut;

delete[] paliMatrix;

return cutNumber;

}