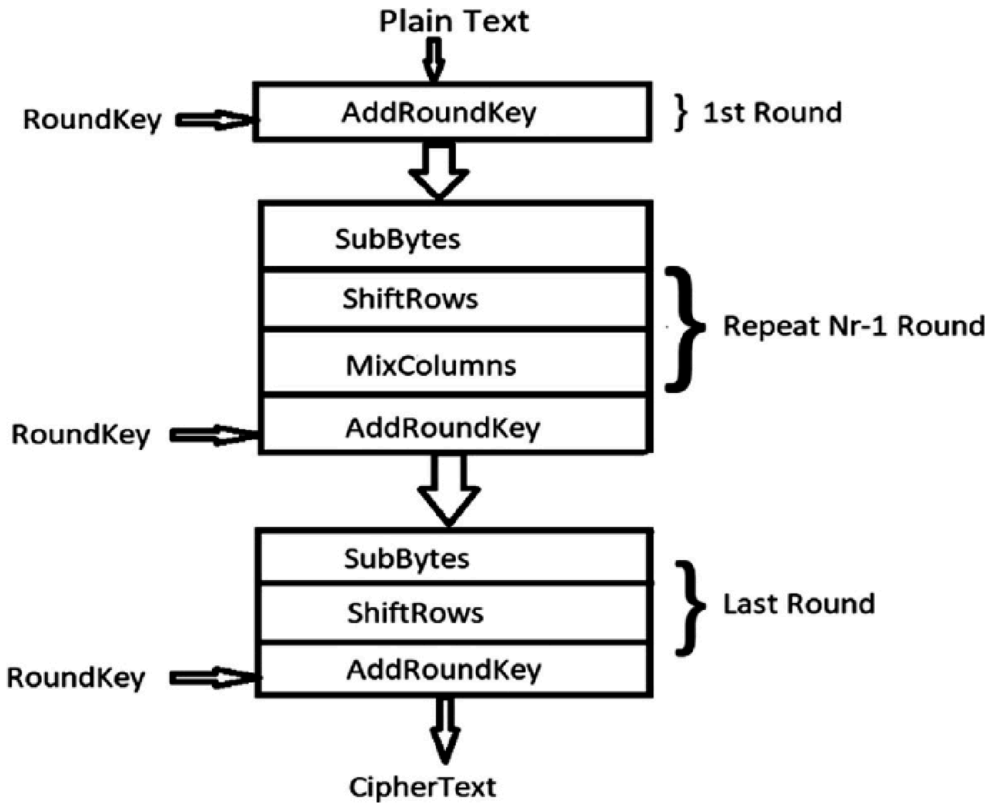
TUGAS 1
ET234203 Struktur Data dan Pemrograman Berorientasi Objek
Tahun Ajaran 2024/2025 Genap

Problem :Reverse a sublist of a linked list

Implementasi Problem Linked-List

Implementasi dalam algoritma enkripsi AES. AES adalah symmetric encryption yang mengenkripsi setiap blok per blok. Setiap 1 round block enkripsi terdapat 4 step.



Dalam implementasi ini, plaintext yang akan dienkripsi akan disimpan menggunakan linkedlist per karakter dan kemudian baru dioperasikan. Reverse sublist akan diimplementasi pada step ShiftRows untuk membalik beberapa karakter yang tersimpan.

Full code:

```
#include <iostream>
#include <cstdint>
using namespace std;

const uint8_t sbox[256] = {
   0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7,
0xab, 0x76,
   0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4,
0x72, 0xc0,
   0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8,
0x31, 0x15,
   0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb,
0x27, 0xb2, 0x75,
   0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29,
0xe3, 0x2f, 0x84,
   0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c,
0x58, 0xcf,
   0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c,
0x9f, 0xa8,
   0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff,
0xf3, 0xd2,
   0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d,
0x19, 0x73,
   0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde,
0x5e, 0x0b, 0xdb,
   0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91,
0x95, 0xe4, 0x79,
   0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a,
0xae, 0x08,
   0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b,
0xbd, 0x8b, 0x8a,
   0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86,
0xc1, 0x1d, 0x9e,
   0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce,
0x55, 0x28, 0xdf,
   0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0,
0x54, 0xbb, 0x16
};

uint8_t invsbox[256];
```

```
void generateInvSBox() {
    for (size_t i = 0; i < 256; i++) {
        invsbox[sbox[i]] = i;
    }
}

const uint8_t mix_columns_matrix[4][4] = {
    {2, 3, 1, 1},
    {1, 2, 3, 1},
    {1, 1, 2, 3},
    {3, 1, 1, 2}
};


const uint8_t inv_mix_columns_matrix[4][4] = {
    {0x0e, 0x0b, 0x0d, 0x09},
    {0x09, 0x0e, 0x0b, 0x0d},
    {0x0d, 0x09, 0x0e, 0x0b},
    {0x0b, 0x0d, 0x09, 0x0e}
};


struct Node {
    int data;
    Node* next;

    Node(int c) : data(c), next(nullptr) {}
};

class Plaintext {
private:
    void reverseSublist(int start, int end) {
        Node* prev = nullptr;
        Node* curr = head;
        for (int i = 0; curr && i < start; ++i) {
            prev = curr;
            curr = curr->next;
        }

        Node* sublistHead = prev;
        Node* sublistTail = curr;
        Node* next = nullptr;
        Node* prevSublist = nullptr;
        for (int i = start; curr && i < end; ++i) {
            next = curr->next;
            curr->next = prevSublist;
            prevSublist = curr;
            curr = next;
        }

        if (sublistHead) {
            sublistHead->next = prevSublist;
        } else {
            head = prevSublist;
        }
        sublistTail->next = curr;
    }

public:
    Node* head;

    Plaintext() : head(nullptr) {}

    void append(char c) {
        Node* newNode = new Node(static_cast<int>(c));
        if (!head) {
            head = newNode;
            return;
        }
        Node* temp = head;
        while (temp->next) {
            temp = temp->next;
        }
        temp->next = newNode;
    }

    void print() {
        Node* temp = head;
        while (temp) {
            cout << temp->data << ",";
            temp = temp->next;
        }
        cout << endl;
    }

    void subBytes(const uint8_t sbox[256]) {
        Node* temp = head;
        while (temp) {
            temp->data = sbox[static_cast<uint8_t>(temp->data)];
            temp = temp->next;
        }
    }


    void addRoundKey(const string& key) {
        Node* temp = head;
        size_t i = 0;
        while (temp && i < key.size()) {
            temp->data ^= key[i];
            temp = temp->next;
            i++;
        }
    }

uint8_t gmul(uint8_t a, uint8_t b) {
        uint8_t p = 0;
        while (b) {
            if (b & 1) {
                p ^= a;
            }
            bool hi_bit_set = (a & 0x80);
            a <<= 1;
            if (hi_bit_set) {
                a ^= 0x1b;
            }
            b >>= 1;
        }
        return p;
}

    void mixColumn(const uint8_t mix[4][4]) {
        Node* temp = head;
        if (!temp) return;

        uint8_t state[4][4];
        int index = 0;
        for (int i = 0; i < 4; ++i) {
            for (int j = 0; j < 4; ++j) {
                if (temp) {
                    state[i][j] = int(temp->data);
                    temp = temp->next;
                }
            }
        }

        uint8_t newState[4][4] = {0};
        for (int i = 0; i < 4; ++i) {
            for (int j = 0; j < 4; ++j) {
                newState[i][j] = gmul(mix[i][0], state[0][j]) ^
                                 gmul(mix[i][1], state[1][j]) ^
                                 gmul(mix[i][2], state[2][j]) ^
                                 gmul(mix[i][3], state[3][j]);
            }
        }

        temp = head;
        for (int i = 0; i < 4; ++i) {
            for (int j = 0; j < 4; ++j) {
                if (temp) {
                    temp->data = newState[i][j];
                    temp = temp->next;
                }
            }
        }
    }

    void shiftRows() {
        reverseSublist(4, 8);
        reverseSublist(12, 16);
    }

};

int main() {
    generateInvSBox();
    Plaintext text;
    string key;
    string input_text;

    do {
        cout << "Enter key (16 characters): ";
        cin >> key;
    } while (key.length() != 16);

    do {
        cout << "Enter plaintext (16 characters): ";
        cin >> input_text;
    } while (input_text.length() != 16);

    for (char c : input_text) {
        text.append(c);
    }

    cout << "SubBytes: ";
    text.subBytes(sbox);
    cout << "Before reverse sublist: ";
    text.print();
    text.shiftRows();
    cout << "Before reverse sublist: ";
    text.print();
    text.mixColumn(mix_columns_matrix);
    cout << "Mix Column: ";
    text.print();
    text.addRoundKey(key);
    cout << "Add Round Key or Ciphertext round 1: ";
    text.print();
    return 0;
}
```

Penjelasan:

Linkedlist untuk menyimpan karakter

```
struct Node {
   int data;
   Node* next;

   Node(int c) : data(c), next(nullptr) {}
};

class Plaintext {
public:
   Node* head;

   Plaintext() : head(nullptr) {}

   void append(char c) {
      Node* newNode = new Node(static_cast<int>(c));
      if (!head) {
         head = newNode;
         return;
      }
      Node* temp = head;
      while (temp->next) {
         temp = temp->next;
      }
      temp->next = newNode;
   }

  void print() {
      Node* temp = head;
      while (temp) {
         cout << temp->data << ",";
         temp = temp->next;
      }
      cout << endl;
   }

};
```

Reverse sublist linkedlist

```
Node *reverse(struct Node *head) {
   Node *prevNode = NULL;
   Node *currNode = head;
   while (currNode) {
      Node *nextNode = currNode->next;
      currNode->next = prevNode;
      prevNode = currNode;
      currNode = nextNode;
   }
   return prevNode;
}

Node *reverseBetween(Node *head, int m, int n) {

   // If m and n are the same, no reversal is needed
   if (m == n)
      return head;

   Node *revs = NULL, *revs_prev = NULL;
   Node *revend = NULL, *revend_next = NULL;

   // Traverse the list to locate the nodes
   // and pointers needed for reversal
   int i = 1;
   Node *currNode = head;
   while (currNode && i <= n) {

      // Track the node just before the start of
      // the reversal segment
      if (i < m)
         revs_prev = currNode;

      // Track the start of the reversal segment
      if (i == m)
         revs = currNode;

      // Track the end of the reversal
      // segment and the node right after it
```

```
        if (i == n) {
            revend = currNode;
            revend_next = currNode->next;
        }
        currNode = currNode->next;
        i++;
    }

    // Detach the segment to be reversed
    // from the rest of the list
    revend->next = NULL;

    // Reverse the segment from position m to n
    revend = reverse(revs);

    // Reattach the reversed segment back to the list
    // If the reversal segment was not at the head of the list
    if (revs_prev)
        revs_prev->next = revend;
    else
        head = revend;

    // Connect the end of the reversed
    // segment to the rest of the list
    revs->next = revend_next;

    return head;
}
```

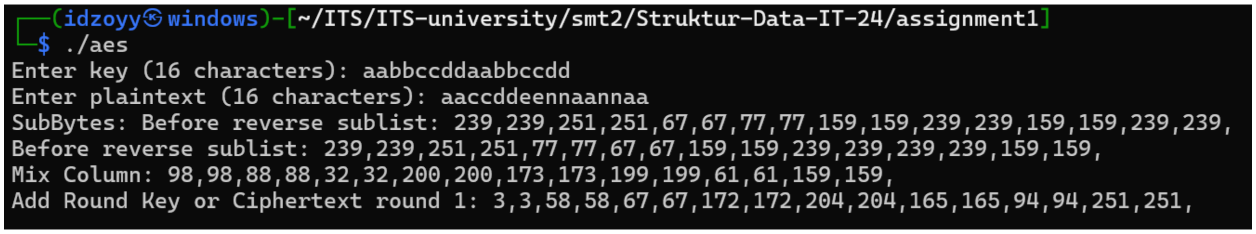
Algoritma Reverse linkedlisst:

- menyimpan nextNode agar referensi tidak hilang.
- Membalikkan arah pointer next untuk membalik urutan.
- Menggeser prevNode ke currNode untuk iterasi berikutnya.
- menggeser currNode ke nextNode untuk melanjutkan.

Algoritma Reverse sublist

- k Edge Case (m == n), return jika tidak perlu reversal.
- Temukan node m dan n, serta simpan pointer penting (revs_prev, revs, revend, revend_next).
- Putuskan koneksi sublist dari sisa linked list.
- Reverse sublist menggunakan reverse.
- Sambungkan kembali sublist ke linked list.
- Return head baru dari linked list.

Visualisasi pada 1 round AES:



```
┌──(idzoyy㉿windows)-[~/ITS/ITS-university/smt2/Struktur-Data-IT-24/assignment1]
└─$ ./aes
Enter key (16 characters): aabbccddaabbccdd
Enter plaintext (16 characters): aaccddeennaannaa
SubBytes: Before reverse sublist: 239,239,251,251,67,67,77,77,159,159,239,239,159,159,239,239,
Before reverse sublist: 239,239,251,251,77,77,67,67,159,159,239,239,239,239,159,159,
Mix Column: 98,98,88,88,32,32,200,200,173,173,199,199,61,61,159,159,
Add Round Key or Ciphertext round 1: 3,3,58,58,67,67,172,172,204,204,165,165,94,94,251,251,
```

SubBytes: Before reverse sublist:
239,239,251,251,**67,67,77,77,**159,159,239,239**,159,159,239,239,**
Before reverse sublist:
239,239,251,251,**77,77,67,67**,159,159,239,239,**239,239,159,159,**