Name: Khoeung Sokhai

ID: e20180408

Group: I5-GIC-B

Discussion_lesson07

1) Find encoder and decoder of LZ77? If we have:

Input string: “abdcaedbdcecabbdeacb” (first block = 7 and second block = 5)

I. Encoder

- Step 1: Compare 5 characters from first block with second block.

abdcaedbdcecabbdeacb

➢ “abdca” ≠ “bdcec” → move 1 character from first block

➢ “bdcae” ≠ “bdcec” → move 1 character from first block

➢ “dcaed” ≠ “bdcec” → no more character from first block

➢ Compare 4 characters → no match: “caed” ≠ “bdce”

➢ Compare 3 characters → match: “bcd” = “bdc”

a b d c a e d b d c e c a b b d e a c b

7 6 5 4 3 2 1

➢ Codeword<6, 3, C(e)> (n=3)

- Step 2: move n+3 (3+1=4) window at first block

➢ Keep taking 7 characters from first block and 5 characters from second block.

aedbdcecabbdeacb

➢ Compare 5 characters from first block with second block.

aedbdcecabbdeacb

➢ “aedbd” ≠ “cabbd” → move 1 character from first block

➢ “edbdc” ≠ “cabbd” → move 1 character from first block

➢ “dbdce” ≠ “cabbd” → no more character from first block

➢ Compare 4 characters → no match: “bdce” ≠ “cabb”

➢ Compare 3 characters → no match: “dce” ≠ “cab”

➢ Compare 2 characters → not match: “ce” ≠ “ca”

➢ Compare 1 characters → match: “c” = “c”

a e d b d c e c a b b d e a c b

7 6 5 4 3 2 1

➢ Codeword<2, 1, C(a)> (n=1)

- Step 3: move n+1 (1+1=2) window at first block
  - ➢ Keep taking 7 characters from first block, but second block rests only 4 characters, so we take only 4 from second block.

    dbdcecabbdeacb
  - ➢ Compare 2 characters from first block with second block.

    dbdcecabbdeacb
  - ➢ "dbdc" ≠ "bbde" → move 1 character from first block
  - ➢ "bdce" ≠ "bbde" → move 1 character from first block
  - ➢ "dcec" ≠ "bbde" → move 1 character from first block
  - ➢ "ceca" ≠ "bbde" → no more character from first block
  - ➢ Compare 3 characters → no match: "dce" ≠ "bbd"
  - ➢ Compare 2 characters → match: "bd" = "bd"

    d b d c e c a b b d e a c b

    7 6 5 4 3 2 1
  - ➢ Codeword<6, 1, C(d)> (n=1)
- Step 4: move n+1 (2+1=3) window at first block
  - ➢ Keep taking 7 characters from first block and 4 characters from second block.

    c e c a b b d e a c b
  - ➢ Compare 2 characters from first block with second block.

    c e c a b b d e a c b
  - ➢ "ceca" ≠ "eacb" → move 1 character from first block
  - ➢ "ecab" ≠ "eacb" → move 1 character from first block
  - ➢ "cabb" ≠ "eacb" → move 1 character from first block
  - ➢ "abbd" ≠ "eacb" → no more character from first block
  - ➢ Compare 4 characters → no match: "abbd" ≠ "eacb"
  - ➢ Compare 3 characters → not match: "bbd" ≠ "eac"
  - ➢ Compare 2 characters → not match: "bd" ≠ "ea"
  - ➢ Compare 1 characters → match: "e" = "e"

    c e c a b b d e a c b

    7 6 5 4 3 2 1
  - ➢ Codeword<6, 1, C(a)> (n=1)
- Step 5: move n+1 (1+1=2) window at first block
  - ➢ Keep taking 7 characters from first, but second block rests only 2 characters, so we take only 2 from second block.

    c a b b d e a c b

➢ “ca” ≠ “cb” → move 1 character from first block

➢ “ab” ≠ “cb” → move 1 character from first block

➢ “bb” ≠ “cb” → move 1 character from first block

➢ “bd” ≠ “cb” → no more character from first block

➢ “de” ≠ “cb” → move 1 character from first block

➢ “ea” ≠ “cb” → no more character from first block

➢ Compare 1 characters → match: “c” = “c”

c a b b d e a c b

7 6 5 4 3 2 1

➢ Codeword<7, 1, C(b)> (n=1)

- Step 6: move n+1 (1+1=2) window at first block

➢ Keep taking 7 characters from first block, but there is no character in second block.

b b d e a c b EOF

➢ So, we stop here. Actually, there are only 5 steps

➢ Encoder = {“abdcaed”, <6, 3, C(e)>, <2, 1, C(a)> , <6, 1, C(d)> , <6, 1, C(a)> , <7, 1, C(b)> }

Input String: a b d c a e d b d c e c a b b d e a c b

Encoder: <6, 3, C(e)> <2, 1, C(a)> <6, 1, C(d)> <6, 1, C(a)> <7, 1, C(b)>

II. Decoder

➢ Encoder = {“abdcaed”, <6, 3, C(e)>, <2, 1, C(a)> , <6, 1, C(d)> , <6, 1, C(a)> , <7, 1, C(b)> }

- Step 1: we have to write the first block string.

➢ So, we get: “abdcaed”

➢ Then use the first result of encoder: <6, 3, C(d)>,

➢ Give index from 1 as in the encoder.

a b d c a e d

7 6 5 4 3 2 1

a b d c a e d **b**

7 6 5 4 3 2 1

a b d c a e d **b d c e**
7 6 5 4 3 2 1

- Step 2: move n+1 (3+1=4) window

  a b d c a e d b d **c e**

  ➢ Use the second result of encoder: <2, 1, C(a)>

  a b d c a e d b d **c e**
  7 6 5 4 3 2 1
  a b d c a e d b d **c e c a**
  7 6 5 4 3 2 1

- Step 3: move n+1 (1+1=2) window

  a b d c a e d b d **c e c a**

  ➢ Use the second result of encoder: <6, 1, C(d)>

  a b d c a e d b d c e c a
  7 6 5 4 3 2 1
  a b d c a e d b d c e c a **b b d**
  7 6 5 4 3 2 1

- Step 4: move n+1 (1+1=2) window

  a b d c a e d b d c e c a **b b d**

  ➢ Use the last result of encoder: <6, 1, C(a)>

  a b d c a e d b d c e c a b b d e a c b

  a b d c a e d b d c e c a b b d
  7 6 5 4 3 2 1
  a b d c a e d b d c e c a b b d **e a**
  7 6 5 4 3 2 1

- Step 5: move n+1 (1+1=2) window

  a b d c a e d b d c e c a b b d **e a**

  ➢ Use the last result of encoder: <7, 1, C(b)>

  a b d c a e d b d c e c a b b d e a
  7 6 5 4 3 2 1

a b d c a e d b d c e c a b b d e a **c b**

7 6 5 4 3 2 1

Thus Decoder: "abdcaedbdcecabbdeacb"

2) Find encoder and decoder of LZ77?

If we have:

Input string: "daddacabeacaebccdaabbeacb"

(first block = 8 and second block = 6)

III. Encoder

- Step 1: Compare 8 characters from first block with second block.

  daddacabeacaebccdaabbeacb

  ➢ "daddac" ≠ "eacaeb" → move 1 character from first block

  ➢ "addaca" ≠ "eacaeb" → move 1 character from first block

  ➢ "ddacab" ≠ "eacaeb" → no more character from first block

  ➢ Compare 5 characters → no match: "ddacab" ≠ "eacae"

  Not match until 1 character

  ➢ Codeword<0, 0, C(e)> (n=0)

- Step 2: move n+1 (1+0=1) window at first block

  ➢ Keep taking 8 characters from first block and 6 characters from second block.

  addacabeacaebccdaabbeacb

  ➢ Compare 8 characters from first block with second block.

  addacabeacaebccdaabbeacb

  ➢ "addaca" ≠ "acaebc" → move 1 character from first block

  ➢ "ddacab" ≠ "acaebc" → move 1 character from first block

  ➢ "dacabe" ≠ "acaebc" → no more character from first block

  ➢ Compare 5 characters → no match: "ddacab" ≠ "acaeb"

  ➢ Compare 3 characters → match: "aca" = "aca"

  a d d a c a b e a c a e b c c d a a b b e a c b

  8 7 6 5 4 3 2 1

  ➢ Codeword<5, 3, C(e)> (n=3)

- Step 3: move n+1 (3+1=4) window at first block

  ➢ Keep taking 8 characters from first block and 5 characters from second block.

  acabeacaebccdaabbeacb

  ➢ Compare 8 characters from first block with second block.

acabeacaebccdaabbeacb

➢ “acabea” ≠ “ebccd” → move 1 character from first block

➢ “cabeac” ≠ “ebccd” → move 1 character from first block

➢ “abeaca” ≠ “ebccd” → no more character from first block

➢ Compare 4 characters → no match: “eaca” ≠ “ebcc”

➢ Compare 1 characters → match: “e” = “e”

a c a b e a c a e b c c d a a b b e a c b

8 7 6 5 4 3 2 1

➢ Codeword<4, 1, C(b)> (n=1)

- Step 4: move n+1 (1+1=2) window at first block

➢ Keep taking 8 characters from first block and 5 characters from second block.

a b e a c a e b c c d a a b b e a c b

➢ Compare 8 characters from first block with second block.

a b e a c a e b c c d a a b b e a c b

➢ “abeac” ≠ “ccdaab” → move 1 character from first block

➢ “beaca” ≠ “ccdaab” → move 1 character from first block

➢ “eacae” ≠ “ccdaab” → no more character from first block

➢ “acaeb” ≠ “ccdaab” → no more character from first block

➢ Compare 4 characters → no match: “caeb” ≠ “ccda”

➢ Compare 1 characters → match: “e” = “e”

a b e a c a e b c c d a a b b e a c b

8 7 6 5 4 3 2 1

➢ Codeword<4, 1, C(c)> (n=1)

- Step 5: move n+1 (1+1=2) window at first block

➢ Keep taking 8 characters from first block and 4 characters from second block.

e a c a e b c c d a a b b e a c b

➢ Compare 8 characters from first block with second block.

e a c a e b c c d a a b b e a c b

compare character not match

➢ Codeword<0, 0, C(d)> (n=0)

- Step 6: move n+1 window at first block

➢ Keep taking 8 characters from first block and 4 characters from second block.

a c a e b c c d a a b b e a c b

➢ Compare 8 characters from first block with second block.

a c a e b c c d a a b b e a c b

➢ Compare 1 characters → match: “a” = “a”

a c a e b c c d a a b b e a c b

8 7 6 5 4 3 2 1

➢ Codeword<8, 1, C(a)> (n=1)

- Step 7: move 2 window at first block

➢ Keep taking 8 characters from first block and 3 characters from second block.

a e b c c d a a b b e a c b

➢ Compare 8 characters from first block with second block.

a e b c c d a a b b e a c b

➢ Compare 1 characters → match: “b” = “b”

a e b c c d a a b b e a c b

8 7 6 5 4 3 2 1

➢ Codeword<6, 1, C(a)> (n=1)

- Step 8: move 2 window at first block

➢ Keep taking 8 characters from first block and 3 characters from second block.

b c c d a a b b e a c b

➢ Compare 8 characters from first block with second block.

b c c d a a b b e a c b

Not match for every cases

➢ Codeword<0, 0, C(e)> (n=0)

- Step 9: move 1 window at first block

➢ Keep taking 8 characters from first block and 2 characters from second block.

c c d a a b b e a c b

➢ Compare 8 characters from first block with second block.

c c d a a b b e a c b

➢ Compare 1 characters → match: “a” = “a”

c c d a a b b e a c b

8 7 6 5 4 3 2 1

➢ Codeword<5, 1, C(c)> (n=1)

- Step 10: move 2 window at first block
  - ➢ Keep taking 8 characters from first block and 1 characters from second block.

    d a a b b e a c b
  - ➢ Compare 8 characters from first block with second block.

    d a a b b e a c b
  - ➢ Compare 1 characters → match: "b" = "b"

    d a a b b e a c b

    8 7 6 5 4 3 2 1

  - ➢ Codeword<5, 1, null)> (n=1)

Therefore, Encoder: <0, 0, C(e)> <5, 3, C(e)> <4, 1, C(b)> <4, 1, C(c)> <0, 0, C(d)> <8, 1, C(a)> <6, 1, C(a)> <0, 0, C(e)> <5, 1, C(c)> <5, 1, null)>

IV. Decode

- Step 1: we have to write the first block string.
  - ➢ So, we get: "daddacab"
  - ➢ Then use the first result of encoder: <0,0, C(e)>
  - ➢ Give index from 1 as in the encoder.

    d a d d a c a b

    8 7 6 5 4 3 2 1

    d a d d a c a b **e**

    8 7 6 5 4 3 2 1

- Step 2: move 1 window

  d a d d a c a b e
  - ➢ Use the second result of encoder: <5, 3, C(e)>

    d a d d a c a b e

    8 7 6 5 4 3 2 1

    d a d d a c a b e **a c a e**

    8 7 6 5 4 3 2 1

- Step 3: move 4 window

  d a d d a c a b e a c a e

  ➢ Use the second result of encoder: <4, 1, C(b)>

  d a d d a c a b e a c a e

  8 7 6 5 4 3 2 1

  d a d d a c a b e a c a e **b**

  8 7 6 5 4 3 2 1
- Step 4: move 2 window

  d a d d a c a b e a c a e b

  ➢ Use the second result of encoder: <4, 1, C(c)>

  d a d d a c a b e a c a e b

  8 7 6 5 4 3 2 1

  d a d d a c a b e a c a e b **c**

  8 7 6 5 4 3 2 1
- Step 5: move 2 window

  d a d d a c a b e a c a e b **c**

  ➢ Use the second result of encoder: <0, 0, C(d)>

  d a d d a c a b e a c a e b **c**

  8 7 6 5 4 3 2 1

  d a d d a c a b e a c a e b **c c d**

  8 7 6 5 4 3 2 1
- Step 6: move 1 window

  d a d d a c a b e a c a e b c c d

  ➢ Use the second result of encoder: <8, 1, C(a)>

  d a d d a c a b e a c a e b c c d

  8 7 6 5 4 3 2 1

  d a d d a c a b e a c a e b c c d **a**

  8 7 6 5 4 3 2 1

- Step 7: move 2 window

  d a d d a c a b e a c a e b c c d a

  ➢ Use the second result of encoder: <6, 1, C(a)>

  d a d d a c a b e a c a e b c c d a

  8 7 6 5 4 3 2 1

  d a d d a c a b e a c a e b c c d a **a**

  8 7 6 5 4 3 2 1
- Step 8: move 2 window

d a d d a c a b e a c a e b c c d a a

➢ Use the second result of encoder: <6, 1, C(a)>

d a d d a c a b e a c a e b c c d a a

8 7 6 5 4 3 2 1

d a d d a c a b e a c a e b c c d a a **b b e a**

8 7 6 5 4 3 2 1

- Step 9: move 2 window

  d a d d a c a b e a c a e b c c d a a b b e a

  ➢ Use the second result of encoder: <0, 0, C(e)>

  d a d d a c a b e a c a e b c c d a a b b e a

  8 7 6 5 4 3 2 1

  d a d d a c a b e a c a e b c c d a a b b e a

  8 7 6 5 4 3 2 1

- Step 9: move 2 window

  daddacabeacaebccdaabbeacb

  d a d d a c a b e a c a e b c c d a a b b e a

  ➢ Use the second result of encoder: <5, 1, null)>

  d a d d a c a b e a c a e b c c d a a b b e a

  8 7 6 5 4 3 2 1

  d a d d a c a b e a c a e b c c d a a b b e a **c b**

  8 7 6 5 4 3 2 1

Therefore Decoder: “daddacabeacaebccdaabbeacb”