Алгоритми за компресиране

Основни понятия

- **Съобщение (s) и кодова дума (w).**
- **Множество от възможни съобщения (S).**
- **Кодиране (като множество).**
- **Кодер.**
- **Декодер.**
- **Типове алгоритми за компресиране:**
  - Без загуба на информация.
  - Със загуба на информация.

Основни понятия (продължение)

- **Компоненти на алгоритмите за компресиране:**
  - Модел.
  - Кодиращ компонент.

- **Метрики за оценка:**
  - Степен на компресия.
  - Скорост на компресиране.
  - Скорост на декомпресиране.
  - Степен на загуба на качество (алгоритми със загуба).
  - Памет, използвана по време на обработка.

Теория на информацията (Клод Шенон)

- **Количество информация.**

$$i(s) = \log_2 \frac{1}{p(s)}$$

- **Ентропия.**

$$H(S) = \sum_{s \in S} p(s) \log_2 \frac{1}{p(s)}$$

Вероятностно кодиране

- **Особености:**
  - Избор на модел.
  - Избор на множество от съобщения.
  - Кодиране на единичен елемент или редица.

- **Префиксни кодове:**
  - Дефиниция и предимства.
  - Представяне като дърво (не непременно двоично).
  - Средна дължина и оптимален префиксен код.

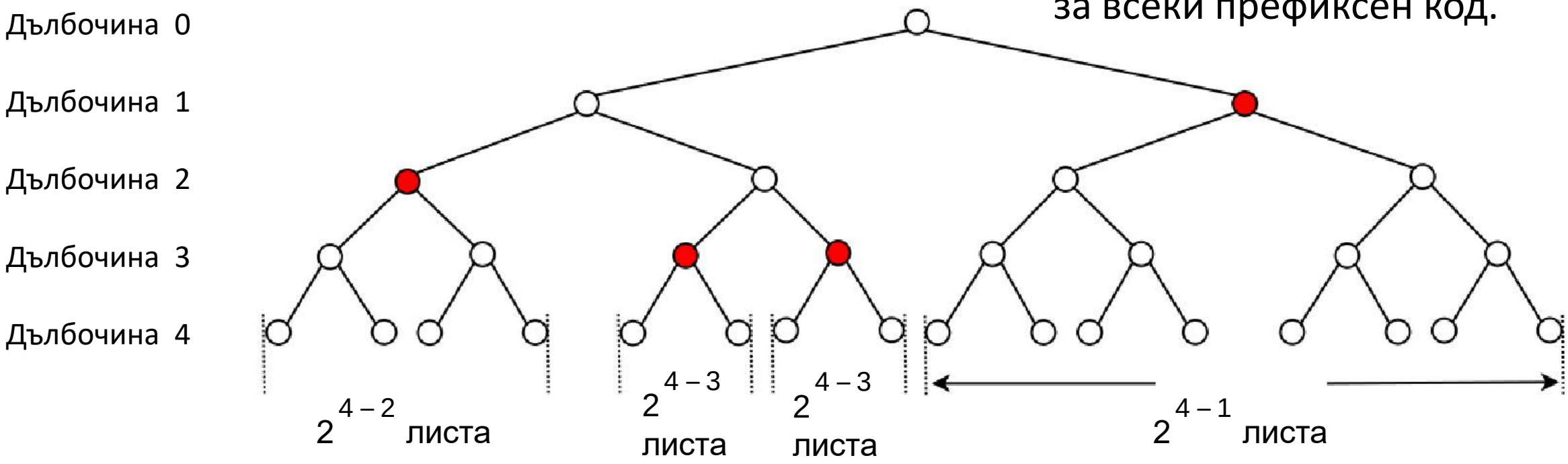
$$l_a(C) = \sum_{(s,w)\in C} p(s)l(w)$$

Вероятностно кодиране и ентропия

- **Неравенство на Крафт—Макмилан:**

$$\sum_{(s,w)\in C} 2^{-l(w)} \leq 1$$

за всеки префиксен код.



Обратно, ако някакви естествени числа удовлетворяват неравенството,
то съществува префиксен код с дължини на кодовите думи = тези числа.

Вероятностно кодиране и ентропия (продължение)

- **Връзка между ентропията и средната дължина на префиксен код:**

$$H(S) \leq l_a(C).$$

Доказателство:

$$H(S) - l_a(C) = \sum_{s \in S} p(s) \log_2 \frac{1}{p(s)} - \sum_{s \in S} p(s) l(s)$$

$$= \sum_{s \in S} p(s) \log_2 \frac{2^{-l(s)}}{p(s)} \overset{\Uparrow}{\leq} \log_2 \left(\sum_{s \in S} 2^{-l(s)} \right) \overset{\Uparrow}{\leq} 0.$$

от неравенството на Йенсен за вдлъбнатата функция $\log_2$

от неравенството на Крафт—Макмилан

Вероятностно кодиране и ентропия (продължение)

- Ако C е оптимален префиксен код, то $l_a(C) \leq H(S) + 1.$

Доказателство: На всяко съобщение $s \in S$ приписваме дължина

$$l(s) = \left\lceil \log_2 \left(\frac{1}{p(s)} \right) \right\rceil.$$

Следователно $l(s) \geq \log_2 \left(\frac{1}{p(s)} \right).$ Оттук правим извод, че

$$\sum_{s \in S} 2^{-l(s)} \leq \sum_{s \in S} p(s) = 1.$$

Вероятностно кодиране и ентропия (продължение)

От забележката след неравенството на Крафт—Макмилан следва, че съществува префиксен код C' с дължини на думите $l(w) = l(s)$.

$$l_a(C') = \sum_{(s,w)\in C'} p(s)l(w) = \sum_{(s,w)\in C'} p(s)\left\lceil \log_2\left(\frac{1}{p(s)}\right)\right\rceil$$

$$\leq \sum_{(s,w)\in C'} p(s)\left(1 + \log_2\left(\frac{1}{p(s)}\right)\right)$$

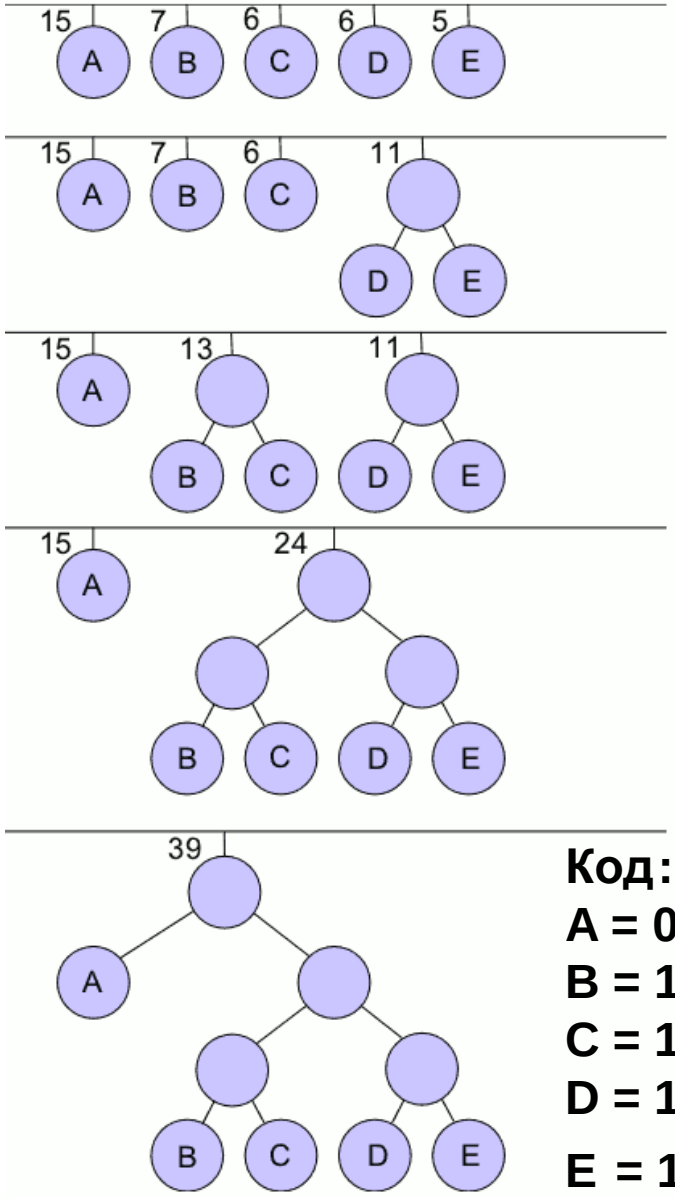
$$= \sum_{(s,w)\in C'} p(s) + \sum_{(s,w)\in C'} p(s)\log_2\left(\frac{1}{p(s)}\right) = 1 + H(S).$$

От оптималността на C следва, че $l_a(C) \leq l_a(C') \leq 1 + H(S)$.

Кодиране на Хъфман

- Алгоритъм:
  1. Започваме с гора от дървета с един връх с тегло = съответната вероятност.
  2. Докато има две или повече дървета:
     i. Избираме двете дървета с най-малки тегла.
     ii. Заместваме ги с едно дърво с тегло = сбора от теглата им.
- Оптимално префиксно кодиране.
- Избор на множество от съобщения.
- Ако при равни тегла избираме по-рано създаден връх (по-плитко дърво), се получава код с най-малка дисперсия

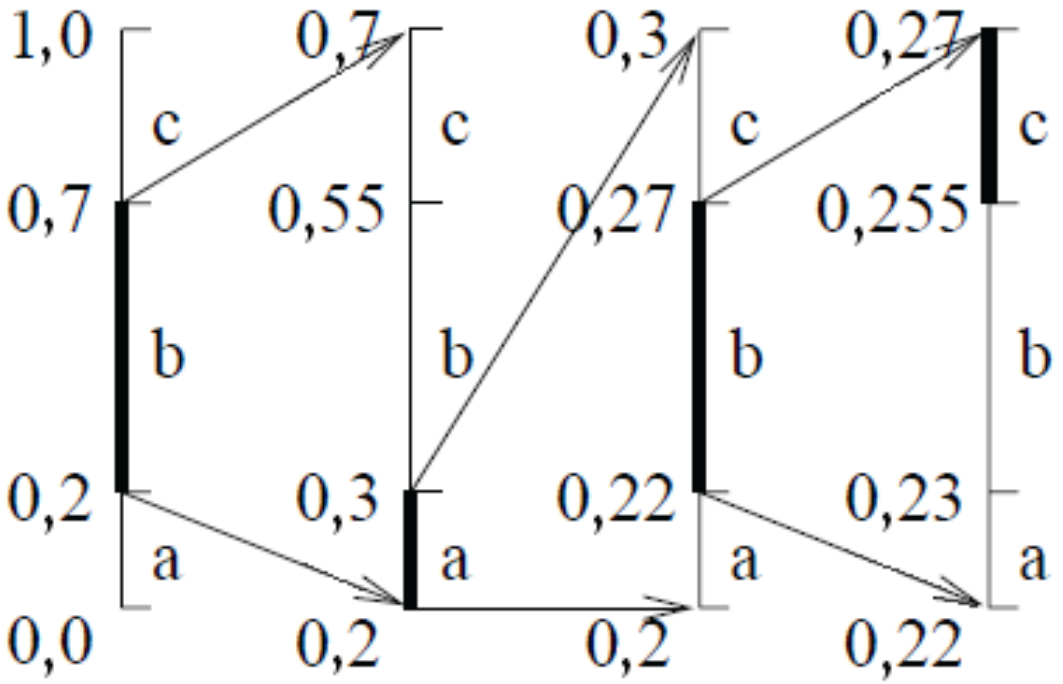
$$\sum_{c \in C} p(c)(l(c) - l_a(C))^2.$$



Код:
A = 0;
B = 100;
C = 101;
D = 110;
E = 111.

Аритметично кодиране

- Идея: цяла редица от съобщения се кодира с едно дробно число.
- Пример: Ако съобщенията **a**, **b** и **c** имат вероятности 20%, 50% и 30% съответно, то редицата **babc** задава интервала от 0,255 до 0,27, който може да се кодира например с числото 0,26.
- Проблеми: най-къс (двоичен) запис, закръгляване, представки.



Избор на модел при вероятностно кодиране

- **Статични и динамични модели.**
- **Използване на контекст:**
  - Трансформации.
  - Условни вероятности.
  - Особености и ограничения.

Примери

- **Алгоритъм RLE** (от англ. run-length encoding):
  заместване на редица от еднакви елементи с броя на повторенията.
  - AAAAAAFDDCCCCCCCAEEEEEEEEEEEEEEEE → 6A1F2D7C1A17E
- **Кодиране с остатъци:** по някакво правило (общо за кодера и декодера) се прогнозира следващата стойност и се записва само разликата между действителната и очакваната стойност. Ако разликите са малки, това води до компресиране на данните.
- **Кодиране с помощта на контекст:** контекстът задава разпределението.

Речникови алгоритми

- **Алгоритми на Лемпел—Зив:**
  - Курсор.
  - Буфер за поглед напред.
- **Алгоритъмът LZ 77**
  използва плъзгащ се прозорец.
  - Речник с фиксирана дължина преди курсора.
  - Подобрения:
    кодиране на Хъфман по канали,
    структури за ускоряване на търсенето.

1. Намираме най-дългата представка, съдържаща се в буфера, както и в речника плюс буфера.
2. Извеждаме кода (*p* ; *n* ; *c*):
   колко позиции назад да се върнем,
   колко символа да прочетем,
   кой е символ № *n* +1.
3. Придвижваме курсора напред.

1	[a]	a	c	a	a	c	a	b	c	a	b	a	a	a	c	(0; 0; a)
2	**a**	[a]	c	a	a	c	a	b	c	a	b	a	a	a	c	(1; 1; c)
3	**a**	**a**	**c**	[a]	a	c	a	b	c	a	b	a	a	a	c	(3; 4; b)
4	a	a	**c**	**a**	**a**	**c**	**a**	**b**	[c]	a	b	a	a	a	c	(3; 3; a)
5	a	a	c	a	a	c	**a**	**b**	**c**	**a**	**b**	**a**	[a]	a	c	(1; 2; c)

Алгоритъм LZW (Lempel—Ziv—Welch)

- Алгоритъмът построява речник в явен вид.
- Всяка дума се добавя в речника само ако думата, образувана от нея чрез задрасквane на последния знак, вече се съдържа в речника.
- В началото речникът съдържа всички еднобуквени думи.
- Кодът е редица от индекси в речника.

$$C' = \mathbf{AddDict}(C, x)$$

$$C' = \mathbf{GetIndex}(C, x)$$

$$W = \mathbf{GetString}(C)$$

$$\mathrm{Flag} = \mathbf{IndexInDict?}(C)$$

Кодиране и декодиране при LZW

```
function LZW_Encode(File)
    C = ReadByte(File)
    while C ≠ EOF do
        x = ReadByte(File)
        C' = GetIndex(C, x)
        while C' ≠ −1 do
            C = C'
            x = ReadByte(File)
            C' = GetIndex(C, x)
        Output(C)
        AddDict(C, x)
        C = x
```

```
function LZW_Decode(File)
    C = ReadIndex(File)
    W = GetString(C)
    Output(W)
    while C ≠ EOF do
        C' = ReadIndex(File)
        if IndexInDict?(C') then
            W = GetString(C')
            AddDict(C, W[0])
        else
            C' = AddDict(C, W[0])
            W = GetString(C')
        Output(W)
        C = C'
```

Въпроси, свързани с речника на LZW

- **Актуализиране:**
  - Създаване на речника наново, ако стане твърде голям.
  - Създаване на речника наново, ако не е ефективен.
  - Изтриване на отделни елементи, ако речникът стане голям. Обикновено се изтриват най-малко използваните елементи.
- **Реализация:**
  - Дърво на представките, зададено с помощта на указатели към родителите.
  - Индексите са указатели към елементи на дървото.
  - Указателите към децата могат да се пазят в списък, в масив или в хеш-таблица.

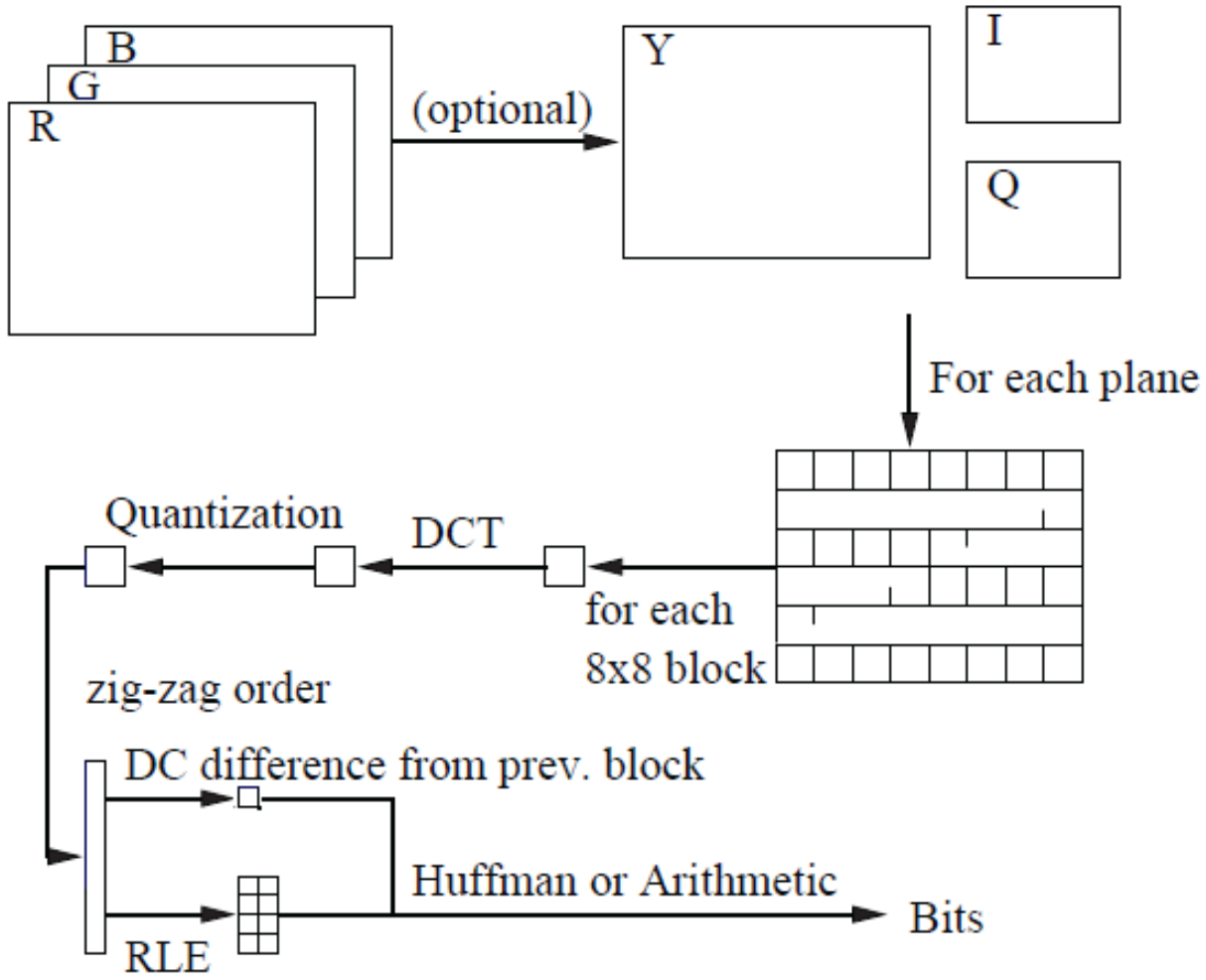
Компресиране със загуба на информация

- **Скаларно квантуване:**
  - Равномерно или неравномерно.
- **Векторно квантуване:**
  - Избор на представители.
- **Компресиране чрез трансформация:**
  - По-добра компресия срещу по-малка загуба на качество.
  - Дискретна косинусова трансформация (DCT), преобразуване на Фурие и др.

$$T_{ij} = \begin{cases} \sqrt{1/n}\cos\frac{(2j+1)i\pi}{2n}, & i = 0, \quad 0 \le j < n; \\ \sqrt{2/n}\cos\frac{(2j+1)i\pi}{2n}, & 0 < i < n, \quad 0 \le j < n. \end{cases}$$

JPEG

- **Стъпки:**
  1. Смяна на цветовото пространство.
  2. Разделяне на канали.
  3. Разделяне на плочки.
  4. Дискретна косинусова трансформация.
  5. Квантуване чрез мащабиране, специфично за всяка отделна честотна компонента.
     - Пренареждане на зигзаг.
     - Финална компресия без загуба на информация.
- **Силни страни:**
  - Контрол върху съотношението “качество—компресия”.
  - Прогресивен алгоритъм.

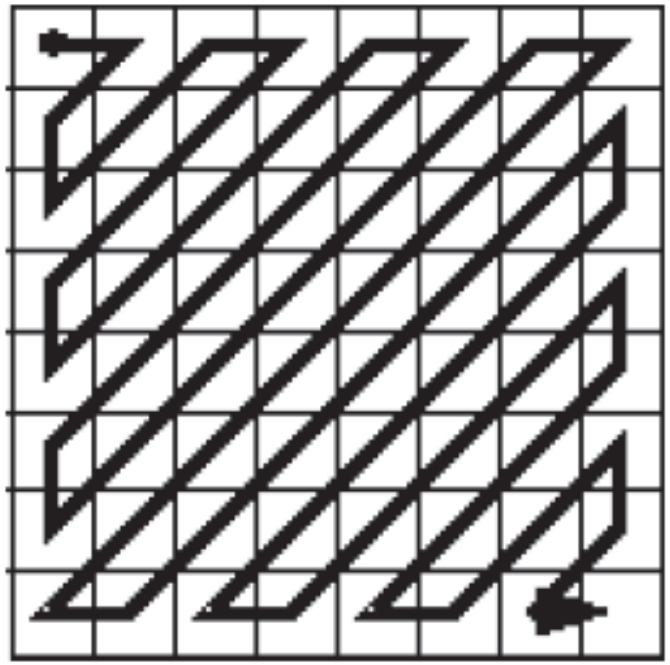


JPEG

Матрица за квантуване

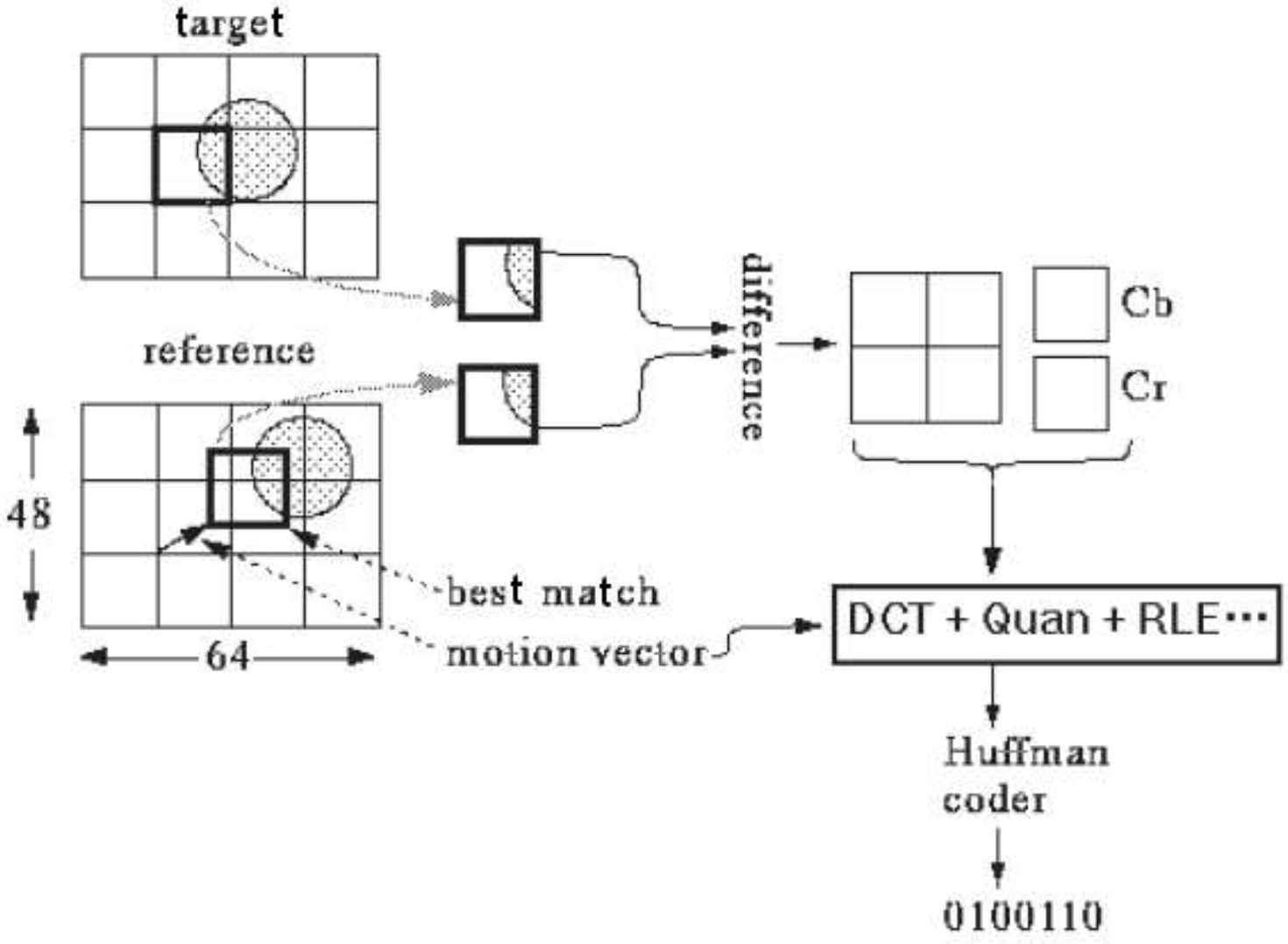
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Подреждане на зигзаг



Други алгоритми за компресиране със загуба на информация

- **MPEG за видеопотоци (редици от кадри):**
  - I-кадри;
  - P-кадри;
  - B-кадри.
- **Компресиране с други трансформации:** вълново преобразуване, фрактали.
- **Компресиране с помощта на специално обучен модел.**



За контакт:
Кристиян Митов,
kmitov94@gmail.com