Kodowanie Huffmana

Kodowanie i kompresja danych - Wykład 2

Maciek Gębala

4 marca 2020

Notatki

Własności optymalnych kodów prefiksowych

- Symbolom występującym częściej odpowiadają krótsze słowa kodowe.
- Dwa najrzadziej występujące symbole mają w kodzie optymalnym słowa kodowe tej samej długości.

Konstruowanie kodów Huffmana

- Kody dwóch najrzadziej występujących symboli różnią się tylko na ostatniej pozycji.
- Algorytm rekurencyjny: rozważ dwa najrzadziej występujące symbole rozróżniając je na końcu przez 0 i 1. Połącz oba w jeden symbol pomocniczy i rozważ teraz rekurencyjnie mniejszy alfabet. Powtarzaj aż zostanie tylko jeden symbol.

Notatki

Przykład

- $A = \{a, b, c, d, e\}$, $P(b) = 0,4$, $P(a) = P(c) = 0,2$ i $P(d) = P(e) = 0,1$.
- Rozważamy d i e: $kod(d) = kod(x)0$, $kod(e) = kod(x)1$ i $P(x) = 0,2$.
- $A' = \{a, b, c, x\}$.
- Rozważamy c i x: $kod(c) = kod(y)0$, $kod(x) = kod(y)1$ i $P(y) = 0,4$.
- $A'' = \{a, b, y\}$.
- Rozważamy a i b: $kod(a) = kod(z)0$, $kod(b) = kod(z)1$ i $P(z) = 0,6$.
- $A''' = \{y, z\}$.
- Rozważamy y i z: $kod(y) = 0$ i $kod(z) = 1$.
- Rekonstruujemy kody: $kod(a) = 10$, $kod(b) = 11$, $kod(c) = 00$, $kod(d) = 010$, $kod(e) = 011$.

Notatki

Własności kodów Huffmana

Kod Huffmana jako drzewo

Tworzenie takiego drzewa jako algorytm tworzenia kodów Huffmana.

Niejednoznaczność tworzenia kodów

Istnieje często wiele kodów dla jednego źródła danych (ale wszystkie mają tę samą średnią długość kodu).

Notatki

Optymalność kodów Huffmana

Optymalny kod powinien spełniać następujące warunki:

- Dla każdych dwóch liter a i b takich, że $P(a) \geqslant P(b)$ zachodzi $l_b \geqslant l_a$.
- Dwie litery o najmniejszych prawdopodobieństwach mają słowa kodowe o tej samej, maksymalnej długości.
- Z każdego wierzchołka wewnętrznego drzewa odpowiadającego kodowi optymalnemu powinny wychodzić oba poddrzewa.
- Jeśli połączymy dwa liście mające wspólnego ojca i ten wierzchołek wewnętrzny potraktujemy jako liść to uzyskane drzewo jest kodem optymalnym dla nowego alfabetu jeśli pierwotne drzewo było optymalne.

Notatki

Długość kodów Huffmana

Dla źródła S spełniona jest nierówność

$$H(S) \leqslant l \leqslant H(S) + 1$$

gdzie l - średnia długość kodu Huffmana dla źródła S.

Notatki

Rozszerzone i niebinarne kody Huffmana

- Kody Huffmana możemy zastosować także do bloków symboli.
- Kody Huffmana można tworzyć także dla innego niż binarny alfabetu wyjściowego - prosta modyfikacja algorytmu: zamiast dwóch symboli łączymy m symboli.

Notatki

Dynamiczne kody Huffmana

- Aby wykonać kodowanie Huffmana musimy znać prawdopodobieństwa (częstość występowania) liter.
- Co zrobić gdy dane napływają na bieżąco i nie znamy statystyk?
- Kodować $k + 1$ symbol na podstawie statystyk k symboli.

Notatki

Przygotowania

Dla alfabetu wejściowego mamy kodowanie stałej długości (pomocnicze).

Dla kodu Huffmana tworzymy na bieżąco jego drzewo o następujących własnościach:

- Każdy liść odpowiada symbolowi i zawiera wagę - ilość dotychczasowych wystąpień.
- Wierzchołki wewnętrzne mają wagę będącą suma wag liści z poddrzew.
- Każdy wierzchołek drzewa ma unikalny numer x_i. Numery te tworzą porządek zgodny z wagami wierzchołków (większa waga to większy numer wierzchołka). Dodatkowo rodzeństwo ma zawsze dwa kolejne numery.

Na początku drzewo zawiera jeden wierzchołek o wadze 0 i etykiecie NYT oznaczającej że symbol nie był jeszcze przesyłany.

Notatki

Opis algorytmu

Pierwsze wystąpienie symbolu *a*

- Wyślij kod NYT i kod stałej długości dla nowego symbolu.
- Stary NYT podziel na dwa wierzchołki potomne - nowy NYT i liść *a*, nadaj *a* wagę 1. Nadaj im odpowiednie numery.
- Zmodyfikuj drzewo dodając 1 do wierzchołków wewnętrznych na ścieżce od *a* do korzenia i przebudowując drzewo tak aby było zgodne z warunkami z poprzedniego slajdu.

Kolejne wystąpienie symbolu *a*

- Znajdź liść *a* i wyślij odpowiadający mu kod.
- Zwiększ wagę *a* o 1.
- Zmodyfikuj drzewo dodając 1 do wierzchołków wewnętrznych na ścieżce od *a* do korzenia i przebudowując drzewo tak aby było zgodne z warunkami.

Notatki

Modyfikacja drzewa

- Zbiór wierzchołków o tej samej wadze nazywamy blokiem.
- Jeśli pierwszy wierzchołek od dołu nie ma największego numeru w swoim bloku to zamieniamy go z tym o największym numerze odpowiednio przebudowując drzewo z zachowaniem własności. Następnie aktualizujemy wagę i patrzymy dalej rekurencyjnie.
- Kończymy jak dojdziemy do korzenia.

Notatki

Przykład

Mamy cztery litery z 26. (W drzewie maksymalnie 51 wierzchołków.)

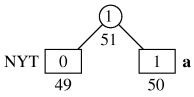
a	d	r	v
00001	00100	10010	10110

Drzewo początkowo wygląda tak:

Notatki

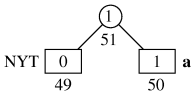
Przykład

Pojawia się litera *a*: Wysyłamy kod NYT (ε) i stały kod *a*: 00001, oraz modyfikujemy drzewo:

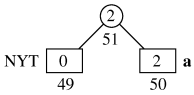


Notatki

Przykład

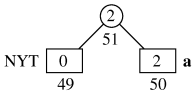


Pojawia się druga litera *a*: Wysyłamy kod *a*: 1 (z drzewa lewe krawędzie to 0 a prawe to 1), oraz modyfikujemy drzewo:

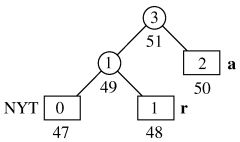


Notatki

Przykład

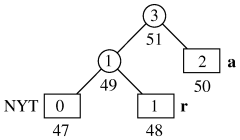


Pojawia się litera *r*: Wysyłamy kod NYT (0) i stały kod *r*: 010010, oraz modyfikujemy drzewo:

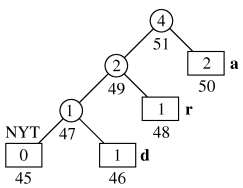


Notatki

Przykład

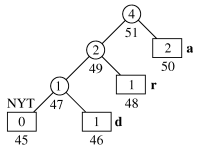


Pojawia się litera *d*: Wysyłamy kod NYT (00) i stały kod *d*: 0000100, oraz modyfikujemy drzewo:

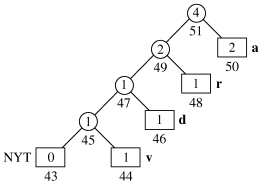


Notatki

Przykład

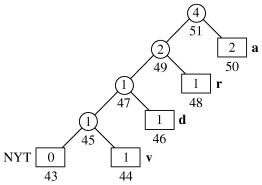


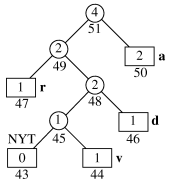
Pojawia się litera *v*: Wysyłamy kod NYT (000) i stały kod *v*: 0010110, oraz modyfikujemy drzewo:

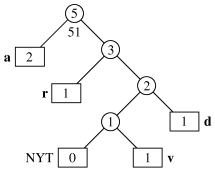


Notatki

Przykład







Notatki

Podsumowanie - własności

- Optymalność wśród kodów prefiksowych.
- Kodowanie i dekodowanie w czasie liniowym (szybkie).
- Kody rozszerzone - kompromis między optymalnością a wielkością systemu.
- Możliwość implementacji dynamicznej.

Notatki

Notatki