Tabriz Journal of Electrical Engineering (TJEE), vol. 54, no. 4, December 2024 Serial no. 110
DOI: 10.22034/tjee.2024.58987.4752

Generating all compact codes with constraint on the smallest codelength

Hamed Narimani, Mohammadali Khosravifard*

Department of electrical and computer engineering, Isfahan University of Technology, Isfahan, Iran
E-mails: h.narimani@iut.ac.ir; khosravi@iut.ac.ir

Short Abstract
Although by employing the Huffman algorithm one can construct the compact code (code with Kraft sum equal to 1) with minimum redundancy for an information source, in some problems it is required to first construct all possible compact codes and then select an appropriate one on the basis of a desired criterion. In particular, if the length of all codewords of an n-tuple compact code is λ or more, then the difference between the largest and the smallest codeword lengths is limited to $n-2^{\lambda}$, and as a result, by considering larger values for λ, the decoding procedure can be accomplished more efficiently. The main goal of this paper is construction of all such codes and an algorithm is introduced which generates *only* these codes (i.e., n-tuple compact codes with all codewords of length λ or more). Noting the correspondence between the multiplicity vectors of the compact codes and some sequences of numbers, we find the necessary and sufficient condition that a sequence of numbers is correspondent with a compact code with the shortest codeword at least λ bits long. This way, by generating all suitable sequences, all the desired compact codes can be constructed without generating any other compact code. Using the proposed algorithm, less computational resources are required. For example, for $\lambda=3$, the required computational resources for generating only the desired compact codes are 5 percent of those when all compact codes are generated.

Keywords
Compact code, Kraft sum, Huffman code, multiplicity vector, smallest codeword length, redundancy.

1- Short Introduction

Compact codes (resp. compact binary trees) have significant role in source coding (resp. computer science). The minimum redundancy prefix-free code, which can be obtained by the Huffman algorithm, is always a compact code. However, sometimes all compact codes must be constructed and examined one by one. For instance, when the source is not completely known and/or more complicated criteria (rather than the redundancy) is considered. Construction of all compact codes has been extensively studied from different points of views. The problem of this paper is construction of those compact codes whose codeword lengths are not less than a given value λ. Such codes are important when limited variation in delay of decoding symbols is desired in addition to low redundancy.

2- Proposed Work and Methodology

If the shortest codeword of an n-tuple compact code is at least λ bits, then the difference between the lengths of the longest and the shortest codewords is at most $n-2^{\lambda}$ bits. Thus, by choosing an appropriate value between 1 and $\lfloor \log n \rfloor$ for λ, the variation in delay of decoding symbols can be limited. The main goal of the paper is construction of all n-tuple compact codes whose shortest codeword is at least λ bits. This particular problem has not been considered in the literature. For simplicity, we deal with multiplicity vector of each compact code. Using two functions $\phi(\cdot)$ and $\psi(\cdot)$, and noting that the multiplicity vector of each compact code can be obtained by applying a sequence of ϕ and ψ functions, a correspondence between compact codes and some sequences of non-negative integers can be defined. In this paper, sequences of non-negative integers that correspond with compact codes with the smallest codeword length not less than λ, are characterized and an algorithm is proposed to generate all such sequences.

3- Conclusion

A trivial solution for constructing all n-tuple compact codes with the lower bound λ on the length of the shortest codeword length is using one of the known algorithms to generate all n-tuple compact codes and filter out those which do not meet the constraint. But the problem is that the number of all n-tuple compact codes grows exponentially with n and considerable resources (time/memory) may be wasted in the mentioned procedure. For instance, for $n=33$ there exist 33818794 compact cods but only 15298 of them satisfy the constraint with $\lambda=4$. The significance of the proposed algorithm is that it generates only the required compact codes.

4- References

[1] M. Khosravifard, M. Esmaeili, H. Saidi, T. Aron Gulliver, "A tree based algorithm for generating all possible binary compact codes with N codewords," IEICE Transactions Fundamentals, vol. E86-A, no. 10, pp. 2510-2516, 2003.

[2] H. Narimani, and M. Khosravifard, "The supertree of the compact codes", Proceedings of International Symposium on Telecommunications (IST), pp. 649-655, 2008.

[3] C. Elsholtz, C. Heuberger, H. Prodinger, "The number of Huffman codes, compact trees, and sums of unit fractions", IEEE Transactions on Information Theory, vol. 59, no. 2, pp. 1065-1075, 2013.

[4] K. Hashimoto, K. Iwata, H. Yamamoto, "Enumeration and Coding of Compact Code Trees for Binary AIFV Codes", Proceedings IEEE International Symposium on Information Theory (ISIT), pp. 1527-1531, Jul. 2019.

تولید همه کدهای فشرده با محدودیت روی کوچک‌ترین طول‌کد

حامد نریمانی
استادیار، دانشکده مهندسی برق و کامپیوتر، دانشگاه صنعتی اصفهان، اصفهان، ایران

سید محمدعلی خسروی فرد
دانشیار، دانشکده مهندسی برق و کامپیوتر، دانشگاه صنعتی اصفهان، اصفهان، ایران

چکیده

اگرچه با استفاده از الگوریتم هافمن می‌توان کد فشرده (کد با مجموع کرافت مساوی یک) با حداقل افزونگی را برای یک منبع اطلاعات بدون حافظه ساخت، در برخی مسائل لازم می‌شود که ابتدا همه کدهای فشرده ممکن ساخته شوند و بعد از بین آنها کد مناسب با معیار مورد نظر انتخاب شود. به طور خاص اگر طول همه کلمه‌کدهای یک کد فشرده n تایی λ یا بیشتر باشد، آنگاه اختلاف بزرگ‌ترین و کوچک‌ترین طول کلمه‌کد آن به $n-2^{\lambda}$ محدود می‌شود و درنتیجه با افزایش مقدار λ می‌توان کدبرداری را با کارایی بالاتری انجام داد. ساخت چنین کدهایی هدف اصلی این مقاله است و برای این کار الگوریتمی ارائه می‌شود که فقط همین کدها (کدهای فشرده n تایی که طول همه کلمه‌کدهای آنها λ یا بیشتر باشد) را تولید می‌کند. با توجه به تناظری که بین بردارهای چندگانگی کدهای فشرده با برخی دنباله‌های اعداد وجود دارد، شرط لازم و کافی برای این که یک دنباله از اعداد متناظر یک کد فشرده که کوتاه‌ترین کلمه کدش حداقل λ بیت باشد را پیدا می‌کنیم. بدین ترتیب با تولید همه دنباله‌های مناسب، همه کدهای فشرده مطلوب ساخته می‌شوند بدون اینکه هیچ کد فشرده دیگری تولید شود. با استفاده از الگوریتم پیشنهادی منابع محاسباتی کمتری برای تولید کدهای مطلوب لازم می‌شود. به‌عنوان مثال برای $\lambda=3$ ، منابع محاسباتی لازم برای تولید *(فقط)* کدهای مطلوب، ۵ درصد حالتی است که همه کدهای فشرده تولید شوند.

کلمات کلیدی

کد فشرده، مجموع کرافت، کد هافمن، بردار چندگانگی، کوچک‌ترین طول کلمه‌کد، افزونگی.

نام نویسنده مسئول: دکتر سیدمحمدعلی خسروی‌فرد
ایمیل نویسنده مسئول: khosravi@iut.ac.ir

تاریخ ارسال مقاله: ۱۴۰۲/۰۸/۰۳
تاریخ(های) اصلاح مقاله: ۱۴۰۲/۱۰/۲۹
تاریخ پذیرش مقاله: ۱۴۰۲/۱۱/۳۰

۱- مقدمه

در عصر ارتباطات، فشرده‌سازی داده‌ها همواره به عنوان یک راهکار غلبه بر محدودیت‌های عملی برای ذخیره و ارسال اطلاعات مورد توجه بوده است. با وجود این که امروزه حافظه‌ها بسیار بیشتر و ارزان‌تر از گذشته در دسترس عموم هستند، ولی حجم داده‌ها با چنان سرعتی رو به فزونی است که بهره‌برداری از الگوریتم‌های فشرده‌سازی و تلاش در جهت بهبود و تکامل آنها گریزناپذیر است. روش‌های فشرده‌سازی را به‌صورت کلی می‌توان به دو دسته *بااتلاف* و *بدون‌اتلاف* تقسیم‌بندی نمود. در شرایطی که قسمتی از داده حاوی اطلاعات مفید یا حساسی نباشد می‌توان با حذف آن به‌وسیله روش‌های فشرده‌سازی بااتلاف حجم داده را کاهش داد. امروزه این الگوریتم‌ها از ساده‌ترین امور روزمره زندگی مانند ارسال سیگنال صحبت در یک مکالمه تلفنی و گرفتن یک عکس یا ضبط یک ویدیو به‌وسیله تلفن‌همراه، تا پیچیده‌ترین چالش‌های علمی نظیر ذخیره اطلاعات ژنتیکی و سیگنال‌های بیولوژیکی را تحت تاثیر قرار داده‌اند [۱-۳]. الگوریتم‌های فشرده‌سازی بدون‌اتلاف بر اساس حذف افزونگی‌های آماری حجم داده را به‌گونه‌ای کاهش می‌دهند که می‌توان داده اولیه را بدون هیچ کم و کاستی بازسازی کرد. در این مقاله فشرده‌سازی بدون اتلاف داده‌ها مورد توجه قرار می‌گیرد.

برای فشرده‌سازی دنباله سمبل‌های خروجی یک منبع اطلاعات از کدگذاری منبع استفاده می‌شود. هر کد منبع در واقع یک نگاشت از دنباله سمبل‌های منبع به مجموعه کلمه‌کدها[1] است. هنگام انتخاب یک کد منبع، اولین معیاری که باید در نظر گرفته شود قابلیت کدبرداری یکتای آن کد است، بدین معنی که برای هر دنباله از کلمه‌کدها، بتوان به صورت یکتا و بدون ابهام دنباله سمبل‌های متناظر آن را یافت. برای این که یک کد منبع n تایی با بردار طول کلمه‌کدهای $L=(\ell_1,\ell_2,\dots,\ell_n)$، قابلیت کدبرداری یکتا داشته باشد باید طول کلمه‌کدهای آن در نامساوی کرافت[2] یعنی

$$\sum_{i=1}^{n} 2^{-\ell_i} \le 1 \qquad (1)$$

صدق کند [۴]. در این مقاله اغلب به جای عبارت «طول کلمه‌کد[3]» از واژه «طول‌کد[4]» استفاده می‌شود.

اگر بتوان هر کلمه‌کد از دنباله کلمه‌کدها را بدون نیاز به کلمه‌کدهای بعد

[1] Codewords
[2] Kraft
[3] Codeword length
[4] Codelength

از آن به صورت یکتا کدبرداری کرد (یعنی سمبل متناظرش را یافت) می‌گوییم کد قابلیت کدبرداری آنی دارد یا کد مورد نظر یک کد آنی[۵] است. از آنجا که این قابلیت باعث کاهش تاخیر و سادگی عملیات کدبرداری می‌شود، در اکثر کاربردهای فشرده سازی از آن بهره برده می‌شود. برای اینکه یک کد قابلیت کدبرداری آنی داشته باشد باید بدون پیشوند[۶] باشد یعنی هیچ کلمه‌کدی از آن، پیشوند دیگر کلمه‌کدها نباشد. نکته جالب این که بهره‌برداری از مزیت کدهای بدون پیشوند، یعنی تاخیر کمتر در کدبرداری، باعث کاهش نرخ فشرده سازی نمی‌شود زیرا طول‌کدهای یک کد بدون پیشوند هم (مانند کدهای با قابلیت کدبرداری یکتا) باید در نامساوی کرافت صدق کنند و برعکس اگر طول‌های $(\ell_1, \ell_2, \dots, \ell_n)$در نامساوی کرافت صدق کنند، آنگاه می‌توان با یک الگوریتم سر راست، یک کد بدون پیشوند با این طول‌ها ساخت [۴]. به عبارت دیگر اگر یک کد با قابلیت کدبرداری یکتا در اختیار باشد می‌توان فقط با تغییر کلمه‌کدها بدون تغییر طول آنها به یک کد بدون پیشوند دست یافت. با توجه به آنچه بیان شد برای مشخص کردن یک کد بدون پیشوند کافی است بردار طول‌کدهای آن معلوم باشد.

منبع بدون حافظه[۷] منبعی است که خروجی آن در هر لحظه مستقل از خروجی در لحظات دیگر تولید شود. بنابراین هر منبع بدون حافظه n سمبلی با بردار احتمال سمبل‌های آن یعنی $P = (p_1, p_2, \dots, p_n)$ مشخص می‌شود که p_i احتمال تولید سمبل i ام در خروجی منبع است. بعد از قابلیت کدبرداری یکتا و آنی بودن کد که برقراری نامساوی کرافت برای طول‌کدها را لازم می‌دارند، مهم‌ترین معیار در انتخاب کد برای یک منبع داده شده $(p_1, p_2, \dots, p_n)$، متوسط طول‌کد یعنی $\sum_{i=1}^{n} p_i \cdot \ell_i$ است که متوسط تعداد بیت لازم برای ارسال یا ذخیره سمبل‌های خروجی منبع را نشان می‌دهد و باید تا حد امکان کم باشد. در نظریه اطلاعات ثابت می‌شود که آنتروپی یک منبع اطلاعات که به صورت $-\sum_{i=1}^{n} p_i \cdot \log p_i$ تعریف می‌شود، یک کران پایین برای متوسط طول‌کد است. از این‌رو اغلب از «افزونگی[۸]» که به صورت اختلاف متوسط طول‌کد و آنتروپی تعریف می‌شود، یعنی

$$\mathcal{R}(L, P) \triangleq \sum_{i=1}^{n} p_i \cdot \ell_i + \sum_{i=1}^{n} p_i \cdot \log p_i, \qquad (۲)$$

برای ارزیابی عملکرد یک کد برای یک منبع استفاده می‌کنند. با استفاده از الگوریتم هافمن می‌توان کد بدون پیشوند بهینه (با کمترین افزونگی یا به‌طور معادل کمترین متوسط طول‌کد) را برای یک منبع بدون حافظه با بردار احتمال داده شده به دست آورد. به بیان دیگر، برای منبعی با بردار احتمال P، الگوریتم هافمن جواب مساله بهینه سازی

$$arg \min_{\substack{L=(\ell_1,\ell_2,\dots,\ell_n) \\ \ell_i \in \mathbb{N} \\ \sum_{i=1}^{n} 2^{-\ell_i} \le 1}} \mathcal{R}(L, P) \qquad (۳)$$

را پیدا می‌کند که «کد هافمن» آن منبع نامیده می‌شود. نشان داده می‌شود که برای منابع دی‌ادیک[۹] (منابعی که احتمال وقوع همه سمبل‌های آن‌ها به‌صورت توانِ صحیح و منفی از عدد ۲ هستند) مقدار افزونگی کد هافمن دقیقا مساوی صفر است. برای منابع غیر دی‌ادیک، افزونگی کدهافمن صفر نیست ولی ثابت می‌شود که حتما کمتر از یک بیت است. باید توجه داشت که افزونگی کد هافمن کمترین مقدار افزونگی است که می‌توان با استفاده از یک کتاب‌کد[۱۰] ساده (که هر سمبل منبع را به یک دنباله بیت می‌نگارد) به آن دست یافت. ولی اگر کدگذاری براساس *دنباله‌ای* از سمبل‌های منبع تعریف شود (به‌جای فقط یک سمبل)، به‌عنوان مثال با استفاده از کدگذاری حسابی[۱۱]، می‌توان به افزونگی‌های کمتر از افزونگی کدهافمن هم دست یافت و در حالت غایی (دنباله به‌اندازه کافی طولانی از سمبل‌های منبع) افزونگی را تا حد دلخواه به صفر نزدیک کرد. در نظریه اطلاعات این مطلب با عنوان *قضیه اول شانون*[۱۲] شناخته می‌شود [۴].

ثابت می‌شود که برای هر کد هافمن تساوی کرافت، یعنی $\sum_{i=1}^{n} 2^{-\ell_i} = 1$، برقرار است. به کدهایی که طول‌کدهای آنها در تساوی کرافت صدق کنند «کد فشرده[۱۳]» می‌گویند. بنابراین هر کد هافمن یک کد فشرده است. از طرف دیگر، هر کد فشرده در واقع کد هافمن برای یک منبع دی‌ادیک است. پس به طور کلی مجموعه همه کدهای هافمن n تایی و مجموعه همه کدهای فشرده n تایی با هم مساوی هستند. در ضمن هر کد فشرده متناظر است با یک تجزیه عدد یک به صورت مجموع توانهای صحیح منفی عدد ۲. مسئله تجزیه عدد یک به توانهای منفی عدد ۲ را می‌توان به عنوان حالت خاصی از مساله کسرهای مصری[۱۴] در نظر گرفت و بررسی نمود؛ به عنوان نمونه و سرنخ می‌توان به [۵، ۶] و مراجع مرتبط به آن اشاره کرد. کدهای فشرده علاوه بر کدگذاری منبع، در زمینه ذخیره و بازیابی داده‌ها در ساختارهای درختی هم نقش مهمی دارند و توسط محققان علوم کامپیوتر بررسی شده‌اند (به عنوان نقطه شروع، به [۷] و ارجاعات آن نگاه کنید). هر کد فشرده n تایی متناظر یک درخت فشرده[۱۵]، یعنی درختی با n برگ که هر گره آن یا هیچ فرزندی ندارد یا دقیقا دو فرزند دارد، است. در این کاربرد متوسط زمان لازم برای جستجو در یک درخت فشرده معادل متوسط طول‌کد در کاربرد فشرده سازی است و باید تا حد امکان کم باشد.

تا این‌جا هرآن‌چه در مورد انتخاب کد مناسب و بهینگی کد هافمن مطرح شد مربوط به حالتی است که مقدار احتمال وقوع سمبل‌های منبع دقیقا مشخص باشند. ولی در شرایطی که بردار احتمال سمبل‌ها به طور کامل معلوم نباشد، استفاده از الگوریتم هافمن امکان پذیر نیست. به عنوان مثال حالتی را در نظر بگیرید که یک منبع اطلاعات n سمبلی در بازه‌های زمانی نامعلوم خروجی خود را بر اساس یکی از چند بردار احتمال معلوم تولید می‌کند و هدف ما انتخاب یک کد بدون پیشوند مناسب برای چنین منبعی است. در چنین شرایطی باید از معیارهایی مانند «کمترین بیشینه افزونگی یا مینیمکس[۱۶]» [۸،۹] یا «کمترین متوسط افزونگی[۱۷]» [۱۰،۱۱] استفاده کرد که الگوریتم هافمن برای آن‌ها قابل استفاده نیست. یک راه حل این است که ابتدا همه بردار طول‌کدهای فشرده nتایی ممکن تولید شوند و سپس با جستجو و بر اساس معیار مورد نظر، کد مناسب از بین آن‌ها انتخاب شود. به این ترتیب مساله تولید همه بردار طول‌کدهای فشرده nتایی حایز اهمیت است. به مثال ساده زیر توجه کنید.

مثال۱: فرض کنید یک منبع اطلاعات ۴ سمبلی در اختیار داریم که برای تولید هر سمبل خروجی ابتدا یکی از سه توزیع

$$\begin{aligned} P_1 &= (0.301, 0.275, 0.259, 0.165) \\ P_2 &= (0.383, 0.320, 0.192, 0.105) \\ P_3 &= (0.473, 0.261, 0.142, 0.124) \end{aligned} \qquad (۴)$$

را به عنوان بردار احتمال سمبل‌ها، با احتمال‌های $(\pi_1, \pi_2, \pi_3) = (0.619, 0.317, 0.064)$ انتخاب می‌کند و سپس بر اساس توزیع

[5] Instantaneous code
[6] Prefix-free
[7] Memoryless
[8] Redundancy
[9] D-adic
[10] Codebook
[11] Arithmetic coding
[12] Shannon
[13] Compact code
[14] Egyptian fractions
[15] Compact tree
[16] Minimax
[17] Minave

انتخاب شده، دنباله‌ی سمبل‌های خروجی را تولید می‌کند. با توجه به این که فقط دو بردار طول‌کد ۴تایی (مرتب شده) یعنی $L_1 = (1,2,3,3)$ و $L_2 = (2,2,2,2)$ وجود دارد، کد بهینه با معیار کمترین بیشینه افزونگی از رابطه

$$L^{minimax} = arg \min_{L\in\{L_1,L_2\}} \max_{P\in\{P_1,P_2,P_3\}} \mathcal{R}(L,P) \quad (۵)$$

به دست می‌آید؛ و محاسبه نشان می‌دهد که برای این مثال کد بهینه $L^{minimax} = L_1 = (1,2,3,3)$ است. ولی با معیار کمترین میانگین افزونگی، که کد بهینه آن از

$$L^{minave} = arg \min_{L\in\{L_1,L_2\}} \sum_{i=1}^{3} \pi_i \cdot \mathcal{R}(L,P_i) \quad (۶)$$

به دست می‌آید، کد بهینه برابر با $L^{\text{minave}} = L_2 = (2,2,2,2)$ می‌شود.

نکته مهم در مورد مثال ۱ این است که برای به‌دست آوردن کد بهینه با معیار مینیمکس باید همه کدهای فشرده در اختیار باشند. در این مثال ساده که منابع فقط ۴ سمبل دارند ساختن ۲ کد فشرده ۴ تایی کار پیچیده‌ای نیست. ولی در شرایطی که تعداد سمبل‌های منبع زیاد باشد حتما باید از یک روش کارا برای تولید همه کدهای فشرده بهره برد. در پایان همین بخش به مرور کارهای انجام‌شده در این زمینه خواهیم پرداخت.

در مثالی شبیه به آنچه در بالا مطرح شد، فرض کنید این امکان وجود داشته باشد که در بازه‌های زمانی مختلف از کدهای فشرده متفاوتی برای کدگذاری سمبل‌های خروجی استفاده شود و این کدهای فشرده هم همراه با اطلاعات سمبل‌ها ذخیره یا ارسال شوند. در چنین شرایطی روش کارآیی برای نمایش کدهای فشرده با دنباله‌ای از بیت‌ها لازم می‌شود. در بخش سوم مقاله با استفاده از قضیه ۲ روشی برای نمایش کدهای فشرده ارائه خواهد شد (مثال ۵ را ببینید).

علاوه بر محدودیت‌های آنی بودن کد و کوچک بودن مقدار متوسط طول کد، در برخی کاربردها میزان تاخیر کدبرداری هم معیاری برای انتخاب کد است. برای کدهای آنی، تاخیر کدبرداری یک سمبل خروجی منبع به طول‌کد متناظر با آن سمبل وابسته است. بنابراین متوسط تاخیر کدبرداری سمبل‌ها متناسب با متوسط طول‌کد است. ولی اگر طول‌کدهای متناظر با سمبل‌های منبع، تفاوت قابل توجهی با یکدیگر داشته باشند، تاخیر کدبرداری آن‌ها گستره وسیعی خواهد داشت که ممکن است مطلوب نباشد. به طور مشخص در بدترین حالت طول کوتاه‌ترین کلمه‌کد یک کد فشرده می‌تواند مساوی 1 و طول بزرگ‌ترین طول‌کد می‌تواند مساوی $n-1$ باشد (که برای بردار طول‌کد $(1,2,3,\dots,n-2,n-1,n-1)$ رخ می‌دهد) [۴] و برای nهای بزرگ اختلاف آن‌ها قابل توجه است. به طور ویژه در برخی از کاربردهای رایج کدهای بدون‌پیشوند، مانند استانداردهای فشرده‌سازی تصویر و ویدیو (به‌عنوان نمونه JPEG و MP4)، که سرعت کدبرداری از اهمیت ویژه‌ای برخوردار است، باید از کدهایی استفاده شود که بزرگ‌ترین طول‌کد آنها از حد مشخصی (مثلا ۱۶ بیت در JPEG) بیشتر نباشد تا کارایی پردازنده‌ها هنگام کدبرداری پایین نیاید و در نتیجه سرعت کدبرداری کاهش پیدا نکند [۱۲،۱۳]. هرچند چنین محدودیتی به قیمت کاهش نرخ فشرده‌سازی (یا به عبارت دیگر افزایش افزونگی) تمام می‌شود ولی سرعت کدبرداری در این کاربردها اولویت بالایی دارد. فرض کنید ρ نشان‌گر کران بالای طول‌کدها باشد، یعنی همه طول‌کدها باید کوچک‌تر یا مساوی ρ باشند. در این صورت به جای (۳)، مسئله بهینه‌سازی به فرم

$$arg \min_{\substack{L=(\ell_1,\ell_2,\dots,\ell_n) \\ \ell_i\in\mathbb{N}\ ,\ \ell_i\le\rho \\ \sum_{i=1}^{n} 2^{-\ell_i}=1}} \mathcal{R}(L,P) \quad (۷)$$

باید حل شود تا بتوانیم از بین همه کدهای بدون‌پیشوندی که طول‌کدهایشان کوچک‌تر یا مساوی ρ هستند، کد با کمترین افزونگی را پیدا کنیم. این مسئله در [۱۴] مورد بررسی قرار گرفته و الگوریتم حل آن با پیچیدگی خطی ارائه شده است. ولی اگر بردار احتمال سمبل‌ها با زمان تغییر کند یا دقیقا معین نباشد (مانند آنچه در مثال ۱ برای چند توزیع داده‌شده ذکر گردید)، نمی‌توان از الگوریتم مذکور استفاده نمود. به طور مشخص برای انتخاب کد بهینه با معیار مینیمکس، لازم است که ابتدا همه کدهای فشرده‌ای که شرایط روی طول‌کدها را برآورده می‌کنند تولید شوند و سپس با جستجو در بین آن‌ها، بهترین کد انتخاب گردد. به این ترتیب مساله تولید همه کدهای فشرده با یک محدودیت خاص روی طول‌کدها، مورد توجه قرار می‌گیرد.

این مسئله تا کنون به‌طور خاص در مراجع مورد بررسی قرار نگرفته است و قبل از مقاله حاضر راهی جز تولید همه کدهای فشرده (با استفاده از یکی از الگوریتم‌ها مانند [۷، ۱۵-۱۹]) و جداسازی آنها که محدودیت مورد نظر را ارضا کنند وجود نداشته است. راه‌کار پیشنهادی ما برای حل این مسئله آن است که به‌جای کران بالای ρ، یک کران پایین مانند λ روی طول‌کدها در نظر بگیریم و همه کدهای فشرده با این محدودیت را تولید کنیم. چنان‌که در ادامه در قضیه ۵ بخش ۳ خواهیم دید، اعمال شرط λ بر روی کوچکترین طول‌کد در کنار تساوی کرافت می‌تواند مقدار بزرگترین طول‌کد را محدود کند و از اختلاف زیاد بین طول‌کدها جلوگیری کند. قضیه ۵ بیان می‌دارد که

$$\left.\begin{array}{l} \sum_{i=1}^{n} 2^{-\ell_i} = 1 \\ min(\ell_1,\dots,\ell_n) \ge \lambda \end{array}\right\} \Rightarrow max(\ell_1,\dots,\ell_n) \le n - 2^{\lambda} + \lambda, \quad (۸)$$

یعنی اگر همه طول‌کدها بزرگ‌تر یا مساوی λ باشند تساوی کرافت ایجاب می‌کند که همه طول‌کدها کوچک‌تر یا مساوی $n-2^{\lambda}+\lambda$ باشند. بدین ترتیب اختلاف بزرگ‌ترین و کوچک‌ترین طول‌کد (و در نتیجه اختلاف تاخیر کدبرداری آن‌ها) می‌تواند $n-2^{\lambda}$ باشد که با افزایش λ حتی برای nهای بزرگ هم می‌تواند محدود شود.

البته مقدار λ نباید بزرگ‌تر از $\lfloor \log n \rfloor$ انتخاب شود زیرا از $\lambda > \log n$ نتیجه می‌شود $n < 2^{\lambda}$ و $n - 2^{-\lambda} + \lambda < \lambda$ و طبق رابطه بالا هیچ کد فشرده‌ای وجود نخواهد داشت که طول همه کلمه‌کدهایش بزرگ‌تر یا مساوی λ باشد. در نتیجه حتما باید $\lambda \le \lfloor \log n \rfloor$ برقرار باشد. در حالت خاص اگر $\lambda = \lfloor \log n \rfloor$ باشد تنها یک کد فشرده nتایی وجود دارد که این کد، کد هافمن توزیع n تایی هم احتمال است و دارای $2^{\lfloor \log n \rfloor} - n$ کلمه‌کد به طول $\lfloor \log n \rfloor$ و $2n - 2^{\lfloor \log n \rfloor}$ کلمه‌کد به طول $\lfloor \log n \rfloor$ است [۱۰]. در این حالت، اختلاف طول طولانی‌ترین و کوتاه‌ترین کلمه‌کد به حداقل ممکن می‌رسد.

مسئله تولید همه کدهای فشرده n تایی به طور گسترده‌ای مورد مطالعه قرار گرفته است که در ادامه همین بخش به طور مختصر به مرور آن‌ها خواهیم پرداخت. ولی در این مقاله هدف ما تولید دسته خاصی از کدهای فشرده n تایی است که طول همه کلمه‌کدهایشان حداقل λ باشد و تا کنون در پژوهش‌ها به آن اشاره‌ای نشده است. بدیهی‌ترین راه حل برای این مسئله آن است که از الگوریتم‌های موجود برای تولید همه کدهای فشرده n تایی استفاده کنیم و هر کد فشرده‌ای که شرط مورد نظر را ارضا نکرد کنار بگذاریم. ولی با توجه به این که تعداد کدهای فشرده n تایی با افزایش n به صورت نمایی رشد می‌کند (به فرم تقریبی 0.254×1.794^{n} [۱۷])، کارآیی این روش برای مقادیر بزرگ n بسیار پایین خواهد بود. برای حل مسئله مورد بحث، در این مقاله با الهام گرفتن از برخی مفاهیم مطرح شده در [۱۸]، الگوریتمی برای تولید (**به صورت غیرتکراری**) همه کدهای فشرده n تایی که همه طول‌کدهایشان بزرگ‌تر یا مساوی λ باشد ارائه خواهیم نمود.

چنان که قبلا هم اشاره شد هر کد فشرده متناظر یک درخت فشرده و یک صورت از تجزیه عدد یک به توان‌های منفی عدد ۲ است و اینها همه توصیف‌های

مختلف از یک چیز هستند. بنابراین آنچه در ادامه در مورد پژوهش‌های قبلی مطرح می‌شود می‌تواند مربوط به هر یک از توصیف‌ها باشد. دسته‌ای از مقاله‌ها بر مسئله شمارش تعداد کدهای فشرده n تایی متمرکز بوده‌اند. به عنوان چند نمونه می‌توان به [۱۷، ۲۰-۲۶] اشاره کرد. در [۲۰-۲۴] نویسندگان روابطی بازگشتی برای شمارش تعداد کدهای فشرده ارائه نموده‌اند در حالی که [۱۷، ۲۵، ۲۶] تعداد کدهای فشرده را در حالت مجانبی، یعنی هنگامی که n بزرگ شود، تقریب زده‌اند. دسته دیگری از مقاله‌ها مانند [۷، ۱۵-۱۹] روش‌هایی برای تولید همه کدهای فشرده با n مشخص پیشنهاد داده‌اند. تفاوت این مقاله‌ها عمدتا در روش‌های نمایش و توصیف کدهای فشرده است. به عنوان نمونه، [۱۵،۱۹] مستقیما بردارهای طول‌کد را تولید کرده‌اند، اما [۱۶،۱۸] از بردارهای چندگانگی برای نمایش کد استفاده کرده‌اند و در [۱۷] از دنباله تعداد گره‌های هر سطح درخت استفاده شده است. علی‌رغم تفاوت در روش نمایش کدهای فشرده در این مقاله‌ها، همه آن‌ها هدفی یکسان، یعنی تولید همه کدهای فشرده با تعداد کلمه‌کد معین، را دنبال کرده‌اند. همان طور که قبلا اشاره شد هدف نهایی ما در این مقاله تولید همه کدهای فشرده n تایی است که طول‌کدهایشان حداقل λ باشد.

در بخش دوم مقاله پس از معرفی چند نماد و تعریف (به طور خاص بردار چندگانگی متناظر یک کد فشرده و توابع ϕ و ψ) و اثبات چند لم و قضیه، به ارتباط بین مجموعه کدهای فشرده nتایی و مجموعه کدهای فشرده $n+1$تایی خواهیم پرداخت. در بخش سوم رابطه بین دنباله‌های توابع ϕ و ψ و بردارهای چندگانگی را پیدا می‌کنیم و می‌بینیم که چطور هر بردار چندگانگی از اعمال دنباله‌ای از چند تابع ϕ و ψ بر یک بردار چندگانگی خاص به‌دست می‌آید. در بخش چهارم، دستاورد اصلی این پژوهش، یعنی الگوریتم تولید همه کدهای فشرده با محدودیت کران پایین λ روی کوچک‌ترین طول‌کد ارائه می‌شود. بخش پنجم به جمع‌بندی مطالب اختصاص دارد.

۲- ارتباط بین مجموعه کدهای فشرده $n+1$ تایی و n تایی

در این بخش به ارتباط بین مجموعه کدهای فشرده n تایی و مجموعه کدهای فشرده $n+1$ تایی خواهیم پرداخت. برای این کار لازم است ابتدا چند مفهوم و نماد را تعریف کنیم. بخشی از این مفاهیم و نمادها در [۱۸] هم استفاده شده‌اند.

تعریف۱: برای هر بردار طول‌کد nتایی $(\ell_1, \ell_1, \dots, \ell_n)$، مولفه‌های بردار چندگانگی[18] $(m_1, m_2, \dots, m_n)$ به صورت زیر تعریف می‌شوند

$$m_i \triangleq \left|\{\ell_j | \ell_j = i\}\right| \quad \text{for} \quad 1 \le i \le n. \tag{۹}$$

در واقع مقدار m_i نشانگر آن است که چه تعداد طول‌کد مساوی i در بردار طول‌کد وجود دارد و در نتیجه داریم $\sum_{i=1}^{n} m_i = n$. به عنوان مثال بردار چندگانگی متناظر با بردار طول‌کد $(3,3,3,3,3,3,5,5,6,6,6,6)$ مساوی $(0,0,7,0,2,4,0,0,0,0,0,0,0)$ است.

در این مقاله برای سادگی در بیان‌ها و اثبات‌ها، بدون از دست دادن کلیت مسئله، فرض می‌کنیم بردار احتمال سمبل‌های منبع به صورت نزولی مرتب‌شده هستند یعنی $p_1 \ge p_2 \ge \cdots \ge p_n$. بدیهی است که در چنین حالتی از منظر متوسط طول‌کد، ترجیح بر این است که بردار طول‌کدها به صورت صعودی مرتب‌شده باشند، یعنی $\ell_1 \le \ell_2 \le \cdots \le \ell_n$ و کد هافمن هم حتما چنین خاصیتی را دارد. فایده فرض مرتب بودن بردار احتمال سمبل‌ها این است که هر بردار طول‌کد مرتب‌شده $(\ell_1, \ell_1, \dots, \ell_n)$ را می‌توان بدون ابهام و به صورت یکتا، با بردار چندگانگی متناظر آن $(m_1, m_2, \dots, m_n)$ نمایش داد. به بیان دیگر یک تناظر یک به یک بین بردارهای طول‌کد مرتب شده و بردارهای چندگانگی وجود دارد. فرمول‌بندی برخی مسئله‌ها با بردارهای چندگانگی ساده‌تر از بردارهای طول‌کد است. در ادامه مقاله هر جا سخن از بردار طول‌کد به میان می‌آید منظور بردار طول‌کد «مرتب شده» است.

از آنجا که طول همه کلمه‌کدهای یک کد فشرده n تایی کمتر از n است، آخرین عضو بردار چندگانگی همواره برابر با صفر است. با این وجود، برای سادگی نمادها و روابط و این که تعداد مولفه‌های بردار طول‌کد و بردار چندگانگی متناظر آن مساوی باشند، این صفر آخر را از بردار چندگانگی حذف نمی‌کنیم.

تعریف۲: برای هر بردار دلخواه $X = (x_1, x_2, \dots, x_n)$ (که همه مولفه های آن مساوی صفر نباشند) توابع زیر را تعریف می‌کنیم:

$$\alpha(X) \triangleq n, \tag{۱۰}$$

$$\nu(X) \triangleq \sum_{i=1}^{n} x_i, \tag{۱۱}$$

$$\sigma(X) \triangleq \sum_{i=1}^{n} x_i \cdot 2^{-i}, \tag{۱۲}$$

$$\mu(X) \triangleq max\{j | x_j \neq 0\}, \tag{۱۳}$$

$$\rho(X) \triangleq x_{\mu(X)}. \tag{۱۴}$$

اگر L بردار طول‌کد متناظر با بردار چندگانگی M باشد آنگاه $\alpha(M) = \nu(M)$ نشان‌دهنده تعداد طول‌کدها و تعداد مولفه‌های L هستند، $\sigma(M)$ نشان دهنده مجموع کرافت L است، $\mu(M)$ نشان دهنده آخرین (بزرگترین) طول‌کد L است و $\rho(M)$ نشان دهنده تعداد طول‌کدهای مساوی $\mu(M)$ در L می‌باشد. به عنوان مثال، برای بردار چندگانگی $M = (0,3,1,2,0,0)$ که متناظر بردار طول‌کد $L = (2,2,2,3,4,4)$ است داریم $\alpha(M) = \nu(M) = 6$، $\sigma(M) = 1$، $\mu(M) = 4$ و $\rho(M) = 2$.

با توجه به آن چه گفته شد می‌توان تعریف زیر را انجام داد.

تعریف۳: مجموعه همه کدهای فشرده nتایی مرتب شده که همه طول‌کدهایشان بزرگتر یا مساوی λ باشند را با $\mathcal{M}_n(\lambda)$ نمایش می‌دهیم

$$\mathcal{M}_n(\lambda) \triangleq \{(x_1, x_2, \dots, x_n) | x_i = 0 \; for \; 1 \le i < \lambda, \; x_n = 0, x_i \in \mathbb{Z}^+ \; for \; i \ge \lambda, \nu(X) = n, \sigma(X) = 1\}. \tag{۱۵}$$

یعنی یک بردار چندگانگی، عضو $\mathcal{M}_n(\lambda)$ است اگر و تنها اگر هر چهار ویژگی زیر را داشته باشد:

۱) یک بردار nتایی باشد،

۲) $\lambda - 1$ مولفه اول و مولفه آخر (n ام) آن صفر باشند و بقیه مولفه‌ها اعداد صحیح نامنفی باشند،

۳) مجموع مولفه‌های آن مساوی n باشد،

۴) مجموع کرافت بردار طول‌کد متناظر آن مساوی یک باشد.

لازم به تاکید است که $\mathcal{M}_n(1)$ نشانگر مجموعه همه بردارهای چندگانگی متناظر با همه کدهای فشرده nتایی است که هیچ محدودیت خاصی روی طول‌کدهای آن وجود ندارد.

قرارداد: برای واضح‌تر بودن مطالب، در ادامه این مقاله هر جا با یک $n = \alpha(X)$ مشخص از بردار X نام برده می‌شود منظور یک بردار n تایی با مولفه‌های دلخواه به فرم

$$X = (x_1, x_2, \dots, x_{\mu(X)-1}, x_{\mu(X)}, \overbrace{0,0,\dots,0}^{n-\mu(X)}) \tag{۱۶}$$

است و هر جا با یک $n = \alpha(M)$ مشخص از بردار M نام برده می‌شود منظور یک *بردار چندگانگی* n تایی به فرم

[18] Multiplicity vector

$$M = (m_1, m_2, \dots, m_{\mu(M)-1}, m_{\mu(M)}, \overbrace{0,0,\dots,0}^{n-\mu(M)}) \quad (۱۷)$$

است که عضو مجموعه $\mathcal{M}_n(\lambda)$ با λ معلوم است.

مثال۲: برای $n = 6$ و $\lambda = 1$، ۵ کد فشرده با بردار طول‌کدهای $(1,2,3,4,5,5), (1,2,4,4,4,4), (1,3,3,3,4,4), (2,2,2,3,4,4), (2,2,3,3,3,3)$ وجود دارد که متناظر بردارهای چندگانگی در $\mathcal{M}_6(1)$ $= \{(1,1,1,1,2,0), (1,1,0,4,0,0), (1,0,3,2,0,0), (0,3,1,2,0,0), (0,2,4,0,0,0)\}$ هستند. واضح است که برای $\lambda = 2$، فقط ۲ کد فشرده با بردار طول‌کدهای $(2,2,2,3,4,4), (2,2,4,4,4,4)$ وجود دارد و درنتیجه $\mathcal{M}_6(2) = \{(0,3,1,2,0,0), (0,2,4,0,0,0)\}$.

لم ۱: اگر $M \in \mathcal{M}_n(1)$ باشد آنگاه $\rho(M)$ یک عدد زوج مثبت است.

برهان: از آنجا که بردار M در $\mathcal{M}_n(1)$ قرار دارد بنابراین $\sigma(M) = 1$ و $\rho(M) > 0$ است و با توجه به تعریف $\mu(M)$ می‌توان نوشت

$$\rho(M) \cdot 2^{-\mu(M)} + \sum_{i=1}^{\mu(M)-1} m_i \cdot 2^{-i} = 1 \quad (۱۸)$$

و در نتیجه

$$\rho(M) + \sum_{i=1}^{\mu(M)-1} m_i \cdot 2^{\mu(M)-i} = 2^{\mu(M)}. \quad (۱۹)$$

از آنجا که برای $i < \mu(M)$ مقدار $2^{\mu(M)-i}$ و در نتیجه $m_i \cdot 2^{\mu(M)-i}$ عددی زوج است و $2^{\mu(M)}$ هم زوج است، پس مقدار $\rho(M)$ الزاما باید زوج باشد و $m_{\mu(M)} \geq 2$ برقرار است.

تعریف۴: برای هر بردار دلخواه X سه تابع ϕ و ψ و I را به صورت زیر تعریف می‌کنیم

$$\phi(X) \triangleq (x_1, x_2, \dots, x_{\mu(X)-1}, x_{\mu(X)} - 1,2, \overbrace{0,0,\dots,0}^{\alpha(X)-\mu(X)}) \quad (۲۰)$$

$$\psi(X) \triangleq (x_1, x_2, \dots, x_{\mu(X)-1} - 1, x_{\mu(X)} + 2,0, \overbrace{0,0,\dots,0}^{\alpha(X)-\mu(X)}) \quad (۲۱)$$

$$I(X) \triangleq (x_1, x_2, \dots, x_{\mu(X)-1} + 1, x_{\mu(X)} - 2, \overbrace{0,0,\dots,0}^{\alpha(X)-1-\mu(X)}). \quad (۲۲)$$

مثال۳: به عنوان نمونه داریم

$$\varphi\big((1,0,3,2,0,0)\big) = (1,0,3,1,2,0,0)$$
$$\varphi\big((1,1,1,1,2,0)\big) = (1,1,1,1,1,2,0)$$
$$\psi\big((1,0,3,2,0,0)\big) = (1,0,2,4,0,0,0)$$
$$\psi\big((1,1,1,1,2,0)\big) = (1,1,1,0,4,0,0)$$
$$I\big((1,0,2,4,0,0,0)\big) = (1,0,3,2,0,0)$$
$$I\big((1,0,3,1,2,0,0)\big) = (1,0,3,2,0,0).$$

با توجه به تعریف توابع ϕ و ψ و α و σ و ν به راحتی می‌توان موارد مندرج در لم زیر را تصدیق نمود.

لم۲: برای سه تابع ϕ و ψ و I و هر بردار دلخواه X داریم

الف) اگر $\sigma(X) = 1$ باشد آنگاه $\sigma\big(\psi(X)\big) = \sigma\big(\phi(X)\big) = \sigma\big(I(X)\big) = 1$

ب) $\nu\big(\psi(X)\big) = \nu\big(\phi(X)\big) = \nu(X) + 1$ و $\nu\big(I(X)\big) = \nu(X) - 1$

ج) $\alpha\big(\psi(X)\big) = \alpha\big(\phi(X)\big) = \alpha(X) + 1$ و $\alpha\big(I(X)\big) = \alpha(X) - 1$.

لم۳: اگر $M \in \mathcal{M}_n(\lambda)$ باشد آنگاه

الف) $\phi(M) \in \mathcal{M}_{n+1}(\lambda)$

ب) $\psi(M) \in \mathcal{M}_{n+1}(\lambda)$ اگر و تنها اگر $m_{\mu(M)-1} > 0$ باشد.

برهان: طبق تعریف۳، بردار $M = (m_1, \dots, m_n)$ عضو $\mathcal{M}_n(\lambda)$ است اگر و تنها اگر هر چهار شرط $\sigma(M) = 1$، $\nu(M) = n$ و $\alpha(M) = n$ و

$$\begin{cases} m_i = 0 & i < \lambda \quad \text{or} \quad i = n \\ m_i \geq 0 & \lambda \leq i < n \end{cases} \quad (۲۳)$$

برقرار باشند. حال می‌خواهیم نشان دهیم که برای هر $M \in \mathcal{M}_n(\lambda)$، بردارهای $\phi(M)$ و $\psi(M)$ نیز این چهار شرط را برای عضویت در $\mathcal{M}_{n+1}(\lambda)$ خواهند داشت. از لم ۲ می دانیم که سه شرط اول برقرار است. با توجه به تعریف تابع ϕ و لم ۱ که بیان می‌دارد $m_{\mu(M)} \geq 2$، به راحتی می‌توان دید که شرط چهارم نیز برای $\phi(M)$ برقرار است. در پایان، با توجه به تعریف تابع ψ، شرط چهارم برای $\psi(M)$ برقرار است اگر و تنها اگر $m_{\mu(M)-1} - 1 \geq 0$ درست باشد؛ و اثبات کامل است.

لم۴: تابع $I(\cdot)$ دارای خواص زیر است:

الف) اگر $\rho(X) = 2$ باشد آنگاه $X = \phi\big(I(X)\big)$ و اگر $\rho(X) \neq 2$ باشد آنگاه $X = \psi\big(I(X)\big)$. در واقع می‌توان گفت

$$I(X) = \begin{cases} \phi^{-1}(X) & if\ \rho(X) = 2 \\ \psi^{-1}(X) & if\ \rho(X) \neq 2 \end{cases} \quad (۲۴)$$

ب) اگر $M \in \mathcal{M}_n(1)$ باشد آنگاه $I(M) \in \mathcal{M}_{n-1}(1)$.

ج) هر بردار چندگانگی $M \in \mathcal{M}_n(1)$ از اعمال یکی از دو تابع ϕ یا ψ بر یکی از اعضای $\mathcal{M}_{n-1}(1)$ ساخته می‌شود.

برهان:

الف) اگر $\rho(X) = 2$ باشد آنگاه

$$I(X) = (x_1, x_2, \dots, x_{\mu(X)-1} + 1,0, \overbrace{0,0,\dots,0}^{\alpha(X)-1-\mu(X)}) \quad (۲۵)$$

و $\mu(I(X)) = \mu(X) - 1$. بنابراین طبق تعریف تابع $\phi(\cdot)$ خواهیم داشت

$$\phi\big(I(X)\big) = (x_1, x_2, \dots, x_{\mu(X)-1} + 1 - 1,2,0, \overbrace{0,0,\dots,0}^{\alpha(X)-1-\mu(X)}) \quad (۲۶)$$

و برهان برای این حالت کامل است. اگر $\rho(X) > 2$ باشد آنگاه

$$I(X) = (x_1, x_2, \dots, x_{\mu(X)-1} + 1, x_{\mu(X)} - 2, \overbrace{0,0,\dots,0}^{\alpha(X)-1-\mu(X)}) \quad (۲۷)$$

و $\mu(I(X)) = \mu(X)$. بنابراین طبق تعریف تابع $\psi(\cdot)$ خواهیم داشت

$$\psi\big(I(X)\big) = (x_1, x_2, \dots, (x_{\mu(X)-1} + 1) - 1, (x_{\mu(X)} - 2) + 2,0, \overbrace{0,0,\dots,0}^{\alpha(X)-1-\mu(X)}) \quad (۲۸)$$

و برهان برای این حالت هم کامل است.

ب) با توجه به لم۲، تنها دو خاصیت باید بررسی شوند. خاصیت اول آن که باید تمامی‌مولفه‌های بردار $I(M)$ نامنفی باشند؛ با توجه به تعریف تابع $I(\cdot)$ فقط مولفه $m_{\mu(M)} - 2$ جای سئوال دارد که آن هم با توجه به لم ۱ قطعا نامنفی است. خاصیت دوم آن است که آخرین درایه از $I(M)$ باید صفر باشد؛ طبق تعریف $I(\cdot)$، در انتهای $I(M)$ به تعداد $n - 1 - \mu(M)$ صفر وجود دارد و می‌دانیم $\mu(M) \leq n - 1$ است؛ اگر $\mu(M) < n - 1$ باشد آنگاه خاصیت دوم نیز برقرار است؛ اما برای حالت $\mu(M) = n - 1$، تنها بردار چندگانگی با این خاصیت، $M = (\overbrace{1,1,1,\dots,1}^{n-2}, 2,0)$ است که برای آن نیز داریم $I(M) = (\overbrace{1,1,1,\dots,1}^{n-3}, 1 + 1,2 - 2)$. یعنی آخرین درایه صفر است و باز هم خاصیت دوم برقرار است. به این ترتیب اثبات قسمت ب کامل می‌شود.

ج) با توجه به قسمت الف و ب همین لم برهان کامل است.

نکته: لم ۴-ب را می‌توان به این صورت تعمیم داد: اگر $M \in \mathcal{M}_n(\lambda)$ باشد آنگاه اگر $\mu(M) > \lambda$ باشد داریم $I(M) \in \mathcal{M}_{n-1}(\lambda)$، و اگر $\mu(M) = \lambda$ باشد داریم $I(M) \in \mathcal{M}_{n-1}(\lambda - 1)$.

نتیجه این بخش را می‌توان در قضیه زیر خلاصه کرد.

قضیه ۱: با اعمال تابع ϕ بر تک‌تک اعضای مجموعه $\mathcal{M}_n(1)$، و اعمال تابع ψ بر آنهایی که در $m_{\mu(M)-1} > 0$ صدق می‌کنند، همه اعضای مجموعه $\mathcal{M}_{n+1}(1)$ «به صورت غیر تکراری» تولید می‌شوند.

برهان: از لم۳ و لم۴-ج می‌توان نتیجه گرفت که با روش گفته شده در صورت قضیه همه اعضای مجموعه $\mathcal{M}_{n+1}(1)$ تولید می‌شوند. اما برای این که نشان دهیم اعضای این مجموعه به صورت **«غیرتکراری»** تولید می‌شوند کافی است ثابت کنیم که برای هر دو بردار دلخواه X_1 و X_2 همواره داریم $\phi(X_1) \neq \psi(X_2)$. برای این کار باید توجه کرد که $\rho(\phi(X_1)) = 2$ ولی چون طبق تعریف $\rho(X_2) \neq 0$ است پس $\rho(\psi(X_2)) = \rho(X_2) + 2 \neq 2$ و در نتیجه همواره $\phi(X_1) \neq \psi(X_2)$.

بر اساس این قضیه اگر مجموعه $\mathcal{M}_n(1)$ در اختیار باشد می‌توان اعضای $\mathcal{M}_{n+1}(1)$ را از روی آن ساخت. به مثال زیر توجه کنید.

مثال۴: با در اختیار داشتن همه اعضای $\mathcal{M}_6(1)$، یعنی
$\{(1,1,1,1,2,0), (1,1,0,4,0,0), (1,0,3,2,0,0), (0,3,1,2,0,0), (0,2,4,0,0,0)\}$,
می‌توان همه اعضای $\mathcal{M}_7(1)$ را بدون تکرار به دست آورد:

$$\phi((1,1,1,1,2,0)) = (1,1,1,1,1,2,0),$$
$$\phi((1,1,0,4,0,0)) = (1,1,0,3,2,0,0),$$
$$\phi((1,0,3,2,0,0)) = (1,0,3,1,2,0,0),$$
$$\phi((0,3,1,2,0,0)) = (0,3,1,1,2,0,0),$$
$$\phi((0,2,4,0,0,0)) = (0,2,3,2,0,0,0),$$
$$\psi(1,1,1,1,2,0) = (1,1,1,0,4,0,0),$$
$$\psi(1,0,3,2,0,0) = (1,0,2,4,0,0,0),$$
$$\psi(0,3,1,2,0,0) = (0,3,0,4,0,0,0),$$
$$\psi(0,2,4,0,0,0) = (0,1,6,0,0,0,0).$$

۳- رابطه دنباله‌های توابع و بردارهای چندگانگی

در بخش قبل دیدیم که چگونه با در اختیار داشتن همه اعضای مجموعه $\mathcal{M}_{n-1}(1)$ می توان همه اعضای $\mathcal{M}_n(1)$ را ساخت. با این روش اگر مجموعه $\mathcal{M}_{n-1}(1)$ در اختیار نباشد باید ابتدا اعضای آن را بسازیم و برای این کار نیاز به مجموعه $\mathcal{M}_{n-2}(1)$ داریم و قبل از آن باید $\mathcal{M}_{n-3}(1)$ ساخته شده باشد. به راحتی می توان دید که مجموعه $\mathcal{M}_3(1)$ فقط شامل یک بردار چندگانگی $\mathrm{S} = (1,2,0)$ است. بنابراین برای یک n داده‌شده، می‌توان به ترتیب مجموعه‌های $\mathcal{M}_4(1)$ و $\mathcal{M}_5(1)$ و $\mathcal{M}_6(1)$ تا $\mathcal{M}_n(1)$ را از روی قبلی ساخت. بدین ترتیب نقش کلیدی بردار چندگانگی سه‌تایی $\mathrm{S} = (1,2,0)$ در تولید همه کدهای فشرده مشخص می‌شود. در واقع همه بردارهای چندگانگی با اعمال دنباله‌ای از توابع ϕ و ψ بر همین بردار ساخته می‌شوند. با توجه به این که هدف اصلی مقاله تولید کدهای فشرده‌ای است که طول‌کدهایشان کمتر از λ نباشد، در این بخش رابطه دقیق دنباله‌های توابع ϕ و ψ و بردارهای چندگانگی مورد نظر را مشخص می‌کنیم. در ادامه فرض می کنیم $n > 3$ است.

قضیه۲: هر بردار چندگانگی $M \in \mathcal{M}_n(1)$ از اعمال یک و فقط یک دنباله $n-3$ تایی از توابع ϕ و ψ بر $\mathrm{S} = (1,2,0)$ ساخته می‌شود.

برهان: از لم ۴ می‌دانیم که هر بردار چندگانگی در $\mathcal{M}_n(1)$ از اعمال یکی از دو تابع ϕ یا ψ بر بردار چندگانگی $I(M)$ که در $\mathcal{M}_{n-1}(1)$ قرار دارد به دست می‌آید. بر این اساس و با استفاده از استقرا به راحتی می‌توان نشان داد که با اعمال $n-3$ بار متوالی تابع $I(\cdot)$ بر هر بردار چندگانگی $M \in \mathcal{M}_n(1)$ عضوی از $\mathcal{M}_3(1)$ به دست می‌آید. از طرفی $\mathcal{M}_3(1)$ فقط یک عضو دارد که آن هم $\mathrm{S} = (1,2,0)$ است.

نتیجه: با توجه به این که می‌توان دو تابع ϕ و ψ را فقط با یک بیت 1 و 0 نمایش داد، بر اساس قضیه قبلی هر بردار چندگانگی $M \in \mathcal{M}_n(1)$ را هم می‌توان با یک دنباله $n-3$ بیتی نمایش داد.

مثال۵: دنباله $(0,2,4,0,0,0) \in \mathcal{M}_6(1)$ را در نظر بگیرید. بر اساس لم ۴، می‌توان دید این دنباله از اعمال ψ بر $(0,3,2,0,0) \in \mathcal{M}_5(1)$ به دست آمده است. به همین ترتیب، $(0,3,2,0,0)$ از اعمال ϕ بر $(0,4,0,0) \in \mathcal{M}_4(1)$ به دست آمده است که آن هم از اعمال ψ بر $(1,2,0) \in \mathcal{M}_3(1)$ ساخته می‌شود. پس به طور خلاصه، $\psi\left(\phi(\psi(1,2,0))\right) = (0,2,4,0,0,0)$. به این ترتیب می‌توان دنباله تابع $\psi\phi\psi$ یا به طور معادل دنباله باینری 010 را برای نمایش بردار $(0,2,4,0,0)$ به کار برد.

نکته مهم این‌جاست که از اعمال هر دنباله $n-3$ تایی دلخواه از توابع ϕ و ψ بر $\mathrm{S} = (1,2,0)$ لزوما یک عضو $\mathcal{M}_n(1)$ ساخته نمی‌شود. به عنوان مثال اگر دنباله $\phi\psi\psi$ بر $S = (1,2,0)$ اعمال شود بردار $(-1,5,2,0,0,0)$ به دست می‌آید که عضو $\mathcal{M}_6(1)$ نیست. در ادامه خواهیم دید که چه دنباله‌های $n-3$تایی از توابع ϕ و ψ باید بر $\mathrm{S} = (1,2,0)$ اعمال شوند تا فقط اعضا $\mathcal{M}_n(1)$ ساخته شوند و هیچ دنباله دیگری ساخته نشود.

قرارداد: از این پس برای سادگی هر دنباله از توابع ϕ و ψ را به فرم $\psi_{z_k}\phi\psi_{z_{k-1}}\phi \dots \phi\psi_{z_2}\phi\psi_{z_1}$ نمایش می‌دهیم که در آن k یک عدد صحیح مثبت و مقادیر z_i برای $1 \leq i \leq k$ اعداد صحیح نامنفی هستند و ψ_z نشان دهنده یک توالی zتایی از تابع ψ است یعنی $\psi_z = \overbrace{\psi \circ \psi \circ \cdots \circ \psi}^{z}$.

مثال۶: بردار چندگانگی $(1,1,0,1,5,1,0,4,0,0,0,0)$ از اعمال دنباله تابع $\psi\phi\phi\phi\phi\psi\phi\psi\phi\phi$ بر S ساخته می‌شود یعنی $\psi\phi\phi\phi\phi\psi\phi\psi\phi\psi\phi\phi(S) = (1,1,0,1,5,1,0,4)$. این دنباله تابع را می‌توان به فرم قراردادی $\psi_1\phi\psi_0\phi\psi_0\phi\psi_2\phi\psi_1\phi\psi_0\phi\psi_0$ نوشت. به عنوان مثالی دیگر، داریم $\psi\psi\psi\phi\phi\psi\psi\phi\phi\psi(S) = (0,0,3,10,0,0,0,0,0,0,0,0,0,0)$ و این دنباله تابع را به فرم قراردادی $\psi_4\phi\psi_3\phi\psi_1$ نمایش می‌دهیم.

برای سادگی در روابط بعدی، تعریف می‌کنیم $z_0 \triangleq 0$.

لم۵: اگر $z_1, z_2, \dots, z_k$ دنباله‌ای از اعداد صحیح نامنفی باشند آن‌گاه با اعمال دنباله توابع $\psi_{z_k}\phi\psi_{z_{k-1}}\phi \dots \phi\psi_{z_2}\phi\psi_{z_1}$ به $S = (1,2,0)$، بردار $X = (x_1, x_2, \dots, x_{\alpha(X)})$ به دست می‌آید که برای آن داریم $\sigma(X) = 1$ و

$$\alpha(X) = \nu(X) = k + 2 + \sum_{i=1}^{k} z_i, \tag{۲۹}$$

$$\mu(X) = k + 1, \tag{۳۰}$$

$$\begin{cases} x_i = 1 + 2z_{i-1} - z_i & \text{for } \ 1 \leq i \leq k \\ x_{k+1} = 2 + 2z_k & \\ x_i = 0 & \text{for } \ k+2 \leq i \leq \alpha(X). \end{cases} \tag{۳۱}$$

برهان: با توجه به و تعریف ψ_z و توابع ϕ و ψ و لم ۲ می‌توان دریافت که با اعمال دنباله تابع $\phi\psi_z$ بر بردار

$$F = (f_1, f_2, \dots, f_{\mu(F)-1}, f_{\mu(F)}, \overbrace{0,0,\dots,0}^{\alpha(F)-\mu(F)}) \tag{۳۲}$$

به بردار

$$\begin{aligned} G &= \phi\psi_z(F) \\ &= (f_1, f_2, \dots, f_{\mu(F)-1} - z, f_{\mu(F)} + 2z - 1, 2, \overbrace{0,0,\dots,0}^{\alpha(F)-\mu(F)+z}) \end{aligned} \tag{۳۳}$$

می‌رسیم که برای آن داریم

$$\begin{aligned} &\alpha(G) = \alpha(F) + z + 1 \\ &\nu(G) = (F) + z + 1 \\ &\sigma(G) = \sigma(F) \\ &\mu(G) = \mu(F) + 1 \end{aligned} \tag{۳۴}$$

بنابراین با استقرا می‌توان نشان داد که با اعمال دنباله $\phi\psi_{z_{k-1}}\phi \dots \phi\psi_{z_2}\phi\psi_{z_1}$ بر S، به بردار

$$T = (1 - z_1, 1 + 2z_1 - z_2, 1 + 2z_2 - z_3, \dots, 1 + 2z_{k-1}, 2, \overbrace{0,0,0,0,\dots,0}^{1+\sum_{i=1}^{k-1} z_i}) \tag{۳۵}$$

می‌رسیم که برای آن

$$\begin{aligned}\alpha(T)&=\alpha(S)+k-1+\sum_{i=1}^{k-1}z_i\\ \nu(T)&=\nu(S)+k-1+\sum_{i=1}^{k-1}z_i\\ \sigma(T)&=\sigma(S)\\ \mu(T)&=\mu(S)+k-1\end{aligned}\qquad(۳۶)$$

و در نهایت با اعمال ψ_{z_k} بر T، بردار

$$X=(1-z_1,1+2z_1-z_2,1+2z_2-z_3,\dots,1+2z_{k-1}-z_k,2+2z_k,\overbrace{0,0,0,0,\dots,0}^{1+\sum_{i=1}^{k}z_i})\qquad(۳۷)$$

حاصل می‌شود که برای آن

$$\begin{aligned}\alpha(X)&=\alpha(S)+k-1+\sum_{i=1}^{k}z_i\\ \nu(X)&=\nu(S)+k-1+\sum_{i=1}^{k}z_i\\ \sigma(X)&=\sigma(S)\\ \mu(X)&=\mu(S)+k-1\end{aligned}\qquad(۳۸)$$

با توجه به $\sigma(S)=1$ ، $\nu(S)=\alpha(S)=3$ و $\mu(S)=2$، لم اثبات می‌شود.

قضیه۳: شرط لازم و کافی برای آنکه با اعمال دنباله توابع $\psi_{z_k}\phi\psi_{z_{k-1}}\phi\dots\phi\psi_{z_2}\phi\psi_{z_1}$ به $S=(1,2,0)$، یک بردار چندگانگی عضو $\mathcal{M}_n(1)$ تولید شود آن است که

$$\begin{cases}0\le z_i\le 2z_{i-1}+1 & \text{for } 1\le i\le k\\ \sum_{i=1}^{k}z_i=n-k-2\end{cases}\qquad(۳۹)$$

برهان: فرض کنید با اعمال دنباله توابع $\psi_{z_k}\phi\psi_{z_{k-1}}\phi\dots\phi\psi_{z_2}\phi\psi_{z_1}$ به $S=(1,2,0)$ بردار X تولید شود. با توجه به لم۵ داریم $\sigma(X)=1$ و $\alpha(X)=\nu(X)=k+2+\sum_{i=1}^{k}z_i=k+2+(n-k-2)=n$. بنابراین از تعریف۳ چنین بر می‌آید که شرط لازم و کافی برای این که X عضو $\mathcal{M}_n(1)$ باشد آن است که همه مولفه‌های X نامنفی باشند؛ که با توجه به لم۵ و نامنفی بودن z_iها، معادل با آن است که

$$0\le z_i\le 2z_{i-1}+1 \quad \text{for} \quad 1\le i\le k\qquad(۴۰)$$

برقرار باشد.

قضیه۴: شرط لازم و کافی برای آنکه با اعمال دنباله توابع $\psi_{z_k}\phi\psi_{z_{k-1}}\phi\dots\phi\psi_{z_2}\phi\psi_{z_1}$ به $S=(1,2,0)$ یک بردار چندگانگی عضو $\mathcal{M}_n(\lambda)$ تولید شود آن است که

$$\begin{cases}z_i=2^i-1 & \text{for } 1\le i<\lambda\\ 0\le z_i\le 2z_{i-1}+1 & \text{for } \lambda\le i\le k\\ \sum_{i=1}^{k}z_i=n-k-2\end{cases}.\qquad(۴۱)$$

برهان: مانند آنچه در برهان قضیه ۳ مطرح شد اگر با اعمال دنباله توابع $\psi_{z_k}\phi\psi_{z_{k-1}}\phi\dots\phi\psi_{z_2}\phi\psi_{z_1}$ به $S=(1,2,0)$ بردار X تولید شود، شرط لازم و کافی برای اینکه X عضو $\mathcal{M}_n(\lambda)$ باشد آن است که

$$\begin{cases}x_i=0 & \text{for} \quad i<\lambda\\ x_i\ge 0 & \text{for} \quad \lambda\le i\le k,\end{cases}\qquad(۴۲)$$

و این بدان معنی است که باید شرایط زیر برقرار باشد:

$$\begin{cases}z_i=1+2z_{i-1} & for \quad 1\le i\le\lambda-1\\ 0\le z_i\le 2z_{i-1}+1 & for \quad \lambda\le i\le k\end{cases}.\qquad(۴۳)$$

با توجه به $z_0\triangleq 0$ و این که جواب معادله بازگشتی

$$z_i=1+2z_{i-1} \quad \text{for} \quad 1\le i\le\lambda-1\qquad(۴۴)$$

به فرم

$$z_i=2^i-1 \qquad \text{for} \quad 1\le i<\lambda\qquad(۴۵)$$

است برهان کامل می‌شود.

در ادامه به قضیه‌ای خواهیم پرداخت که نشان می‌دهد با در نظر گرفتن کران پایین λ روی کوچک‌ترین طول‌کد، یک کران بالا روی بزرگ‌ترین طول‌کد ایجاد می‌شود و در نتیجه اختلاف بین طول کوتاه‌ترین و طولانی‌ترین کلمه‌کد محدود می‌شود.

قضیه۵: اگر $M\in\mathcal{M}_n(\lambda)$ باشد آنگاه $\mu(M)\le n-2^{-\lambda}+\lambda$. تساوی فقط برای بردار چندگانگی

$$(\overbrace{0,0,\dots,0}^{\lambda-1},2^\lambda-1,\overbrace{1,1,\dots,1}^{n-2^\lambda-1},2)\qquad(۴۶)$$

برقرار می‌شود که متناظر است با بردار طول‌کد

$$(\overbrace{\lambda,\lambda,\dots,\lambda}^{2^\lambda-1},\overbrace{\lambda+1,\lambda+2,\lambda+3,\dots,n-2^\lambda+\lambda}^{n-2^\lambda},n-2^\lambda+\lambda).\qquad(۴۷)$$

برهان: فرض کنید بردار چندگانگی M از اعمال دنباله توابع $\psi_{z_k}\phi\psi_{z_{k-1}}\phi\dots\phi\psi_{z_2}\phi\psi_{z_1}$ بر $S=(1,2,0)$ به دست آمده باشد. با توجه به قضیه قبلی می‌دانیم که $\mu(M)=k+1$ و با توجه به $k=n-2-\sum_{i=1}^{k}z_i$ واضح است که بیشترین مقدار k زمانی به دست می‌آید که $\sum_{i=1}^{k}z_i$ کمترین مقدار ممکن را داشته باشد. مقدار z_i برای $1\le i<\lambda$ که معلوم است، یعنی $z_i=2^i-1$ و

$$\sum_{i=1}^{\lambda-1}z_i=\sum_{i=1}^{\lambda-1}(2^i-1)=2^\lambda-\lambda-1\qquad(۴۸)$$

ولی برای $\lambda\le i\le k$ مقدار z_iها را باید مساوی صفر قرار داد تا کمترین مقدار $\sum_{i=\lambda}^{k}z_i$ یعنی صفر و در نتیجه بیشترین مقدار $\mu(M)$ به دست آید. بنابراین می‌توان نوشت

$$\begin{aligned}\mu(M)&=1+k=1+n-2-\sum_{i=1}^{\lambda-1}z_i-\sum_{i=\lambda}^{k}z_i\\ &=n-1-\left(2^\lambda-\lambda-1\right)-\sum_{i=\lambda}^{k}z_i\\ &\le n-2^{-\lambda}+\lambda\end{aligned}\qquad(۴۹)$$

و تساوی زمانی برقرار می‌شود که دنباله $z_1,z_2,\dots,z_k$ به صورت

$$\overbrace{1,3,7,\dots,2^{\lambda-1}-1}^{\lambda-1},\overbrace{0,0,\dots,0}^{n-2^\lambda}\qquad(۵۰)$$

باشد. با توجه به لم ۵ برهان کامل است.

بیان دیگر قضیه ۵ این است که اگر برای بردار طول‌کد فشرده $(\ell_1,\ell_2,\ell_3,\dots,\ell_n)$ داشته باشیم $\min(\ell_1,\ell_2,\ell_3,\dots,\ell_n)\ge\lambda$، آن‌گاه $\max(\ell_1,\ell_2,\ell_3,\dots,\ell_n)\le n-2^\lambda+\lambda$ بدین ترتیب اختلاف طول کوتاه‌ترین و طولانی‌ترین کلمه‌کد و در نتیجه اختلاف تاخیر کدبرداری آنها حداکثر $n-2^\lambda$ است که با انتخاب مقادیر بزرگ‌تر برای λ کاهش می‌یابد.

۴-تولید همه بردارهای چندگانگی nتایی

در این بخش می‌خواهیم به مسئله اصلی این مقاله یعنی تولید همه بردارهای چندگانگی عضو $\mathcal{M}_n(\lambda)$ بپردازیم. ابتدا حالت ساده $\lambda=1$ را بررسی می‌کنیم. بر اساس قضیه ۳ اگر بخواهیم همه اعضای $\mathcal{M}_n(1)$ را بسازیم باید ابتدا به ازای همه مقادیر $1\le k\le n-1$ همه دنباله‌های k تایی از اعداد صحیح نامنفی $z_1,z_2,\dots,z_k$ را که در شرایط قضیه ۳ صدق کنند تولید کنیم و بعد برای تک‌تک آنها دنباله توابع متناظر به فرم $\psi_{z_k}\phi\psi_{z_{k-1}}\phi\dots\phi\psi_{z_2}\phi\psi_{z_1}$ را بر $S=(1,2,0)$ اعمال کنیم. به جای این کار، می‌توان با استفاده از قضیه ۶

مستقیما همه دنباله‌های اعداد صحیح نامنفی که متناظر عضوی از $\mathcal{M}_n(1)$ هستند را ساخت.

قضیه۶: اگر مقادیر z_i را برای $i = 1,2,\dots$ به ترتیب یکی پس از دیگری چنان تعیین کنیم که همیشه شرط

$$0 \le z_i \le min\left\{2z_{i-1}+1, n-i-2-\sum_{j=1}^{i-1} z_j\right\} \quad (۵۱)$$

برقرار بماند و به ازای کوچکترین i که مقدار قابل قبولی برای z_i وجود نداشته باشد، یعنی

$$min\left\{2z_{i-1}+1, n-i-2-\sum_{j=1}^{i-1} z_i\right\} < 0 \quad (۵۲)$$

برقرار شود، دنباله $z_1, z_2, \dots, z_{i-1}$ را برای تولید یک بردار چندگانگی استفاده کنیم، آنگاه همه بردارهای چندگانگی در $\mathcal{M}_n(1)$ ساخته می‌شوند.

برهان: فرض کنید بردار چندگانگی $\mathcal{M}_n(1)$ از اعمال دنباله توابع $S = (1,2,0)$ بر $\psi_{z_k}\phi\psi_{z_{k-1}}\phi \dots \phi\psi_{z_2}\phi\psi_{z_1}$ ساخته می‌شود. در این صورت دنباله $z_1, z_2, \dots, z_k$ در شرایط قضیه ۴ صدق می‌کند و حتما در نامعادلات این قضیه، یعنی روابط (۴۲)، هم صدق می‌کند. تنها نکته این است که برای چنین دنباله‌ای مقدار z_{k+1} تعریف نشده است. با توجه به

$$\sum_{i=1}^{k} z_i = n-k-2 \quad (۵۳)$$

داریم

$$n-(k+1)-2-\sum_{j=1}^{k} z_i = -1 \quad (۵۴)$$

و در نتیجه باید z_{k+1} در

$$0 \le z_{k+1} \le min\{2z_k+1, -1\} \quad (۵۵)$$

صدق کند که غیر ممکن است. حال به اثبات درستی عکس این مطلب می‌پردازیم.

فرض کنید دنباله $z_1, z_2, \dots, z_k$ در نامعادلات (۵۱) و (۵۲) صدق می‌کند. با توجه به اینکه z_{k+1} تعریف نشده است ولی z_k تعریف شده، حتما داریم

$$\begin{aligned} &min\left\{2z_k+1, n-(k+1)-2-\sum_{j=1}^{k} z_i\right\} < 0 \\ &0 \le min\left\{2z_{k-1}+1, n-k-2-\sum_{j=1}^{k-1} z_i\right\} \end{aligned} \quad (۵۶)$$

و چون $z_k \ge 0$ است پس

$$n-(k+1)-2-\sum_{j=1}^{k} z_i < 0 \quad (۵۷)$$

و

$$n-k-3-\sum_{j=1}^{k-1} z_i < z_k \quad (۵۸)$$

و در نتیجه

$$n-k-2-\sum_{j=1}^{k-1} z_i \le z_k\,. \quad (۵۹)$$

از طرفی می‌دانیم که

$$z_k \le n-k-2-\sum_{j=1}^{k-1} z_i \quad (۶۰)$$

و در نهایت داریم

$$z_k = n-k-2-\sum_{j=1}^{k-1} z_i\,. \quad (۶۱)$$

در نتیجه دنباله $z_1, z_2, \dots, z_k$ همه شرایط قضیه ۴ را داراست و اگر دنباله توابع $\psi_{z_k}\phi\cdots\phi\psi_{z_2}\phi\psi_{z_1}$ بر روی $S = (1,2,0)$ اعمال شود حتما یک عضو از $\mathcal{M}_n(1)$ حاصل می‌شود.

مثال۷: برای تولید همه اعضای $\mathcal{M}_6(1)$ طبق جدول زیر دنباله‌های قابل قبول از اعداد صحیح نامنفی را می‌سازیم. در مرحله اول برای مقدار z_1 فقط دو انتخاب 0 و 1 وجود دارد. بنابراین فقط دو زیر دنباله از اعداد صحیح نامنفی ساخته می‌شود، (0) و (1). در مرحله بعد برای (0) باید z_2 در محدوده $0 \le z_2 \le \min\{1,2\}$ صدق کند و برای (1) باید z_2 در $0 \le z_2 \le \min\{3,1\}$ صدق کند. بدین ترتیب زیر دنباله‌های (0,0)، (0,1)، (1,0)، (1,1) ساخته می‌شوند که باید در مرحله بعد برای آنها محدوده قابل قبول z_3 تعیین شود. به طور مشخص برای زیردنباله (1,1) باید $0 \le z_3 \le \min\{1,-1\}$ برقرار باشد و بنابراین هیچ مقدار قابل قبولی برای z_3 وجود ندارد و خود دنباله (1,1) می‌تواند یک بردار چندگانگی در $\mathcal{M}_6(1)$ بسازد. در جدول ۱ مراحل تولید همه دنباله‌های اعداد صحیح نامنفی و همه اعضای $\mathcal{M}_6(1)$ نشان داده شده است. دنباله‌هایی که با رنگ خاکستری مشخص شده‌اند کامل هستند و هیچ عدد دیگری نمی‌توان و نباید به آنها اضافه کرد.

جدول ۱: تولید دنباله توابع متناظر با $\mathcal{M}_6(1)$

زیر دنباله اعداد نامنفی	i	حداکثر مقدار قابل قبول z_i	فرم قراردادی دنباله توابع متناظر	دنباله توابع	بردار چندگانگی متناظر
()	1	1			
(0)	2	min{1,2}			
(1)	2	min{3,1}			
(0,0)	3	min{1,1}			
(0,1)	3	min{1,0}			
(1,0)	3	min{1,0}			
(1,1)	3	min{1,−1}	$\psi_1\phi\psi_1$	$\psi\phi\psi$	(0,2,4,0,0,0)
(0,0,0)	4	min{1,0}			
(0,0,1)	4	min{1,−1}	$\psi_1\phi\psi_0\phi\psi_0$	$\psi\phi\phi$	(1,1,0,4,0,0)
(0,1,0)	4	min{1,−1}	$\psi_0\phi\psi_1\phi\psi_0$	$\phi\psi\phi$	(1,0,3,2,0,0)
(1,0,0)	4	min{1,−1}	$\psi_0\phi\psi_0\phi\psi_1$	$\phi\phi\psi$	(0,3,1,2,0,0)
(0,0,0,0)	5	min{1,−1}	$\psi_0\phi\psi_0\phi\psi_0\phi\psi_0$	$\phi\phi\phi$	(1,1,1,1,2,0)

۵-الگوریتم تولید همه کدهای فشرده nتایی با محدودیت روی طول‌کدها و کارایی آن

با برهانی شبیه به آنچه برای قضیه ۶ ارائه شد می‌توان الگوریتم تولید دنباله‌های اعداد نامنفی که بر اساس آنها می‌توان یک بردار چندگانگی در $\mathcal{M}_n(\lambda)$ ساخت را به صورت زیر در نظر گرفت:

مقادیر z_i را برای $i = 1,2,\dots$ به ترتیب یکی پس از دیگری چنان تعیین می‌کنیم که همیشه شرط

$$\begin{aligned} &z_i = 2^i - 1 && 1 \le i < \lambda \\ &0 \le z_i \le min\left\{2z_{i-1}+1, n-i-2-\sum_{j=1}^{i-1} z_i\right\} && i \ge \lambda \end{aligned} \quad (۶۲)$$

برقرار باشد. شبه کد این الگوریتم در شکل زیر مشخص شده است. در این

شبه کد منظور از $Sum(sequence)$ و $Last(sequence)$ به ترتیب مجموع اعضا و آخرین عضو $sequence$ است. پس از تولید همه دنباله‌های $\boldsymbol{z_1, z_2, \cdots}$، می‌توان با توجه به مطالب بخش ۳ (تناظر بین دنباله توابع و بردارهای چندگانگی) همه بردارهای چندگانگی عضو $\mathcal{M}_n(\lambda)$ را تولید نمود.

شبه‌کد تولید همه دنباله‌های $\boldsymbol{z_1, z_2, \cdots}$ متناظر با اعضای $\boldsymbol{\mathcal{M}_n(\lambda)}$

Input: n
Z_sequences $\leftarrow [2^1-1, 2^2-1, 2^3-1, \ldots, 2^{\lambda-1}-1]$
for i in $[1,2,\ldots,n-2]$ **do**
  for z_seq in Z_sequences **do**
  $u = \min(1 + 2 * Last(z\_seq), n-i-2-Sum(z\_seq))$
  if $u \geq 0$ **then**
    for z in $[0,1,2,\ldots,u]$ **do**
      Add [z_seq, u] **to** Z_sequences
    Remove z_seq **from** Z_sequences
Output: Z_sequences

مثال۸: وقتی الگوریتم برای تولید اعضای $\mathcal{M}_{13}(3)$ اجرا شود دنباله‌های اعداد صحیح نامنفی و بردارهای چندگانگی متناظر آنها طبق جدول ۲ به دست می‌آیند. همانطور که از قضیه ۵ انتظار داشتیم بزرگ‌ترین مقدار $\mu(\cdot)$ برای این بردارهای چندگانگی $13 - 2^3 + 3 = 8$ است که در آخرین سطر جدول دیده می‌شود.

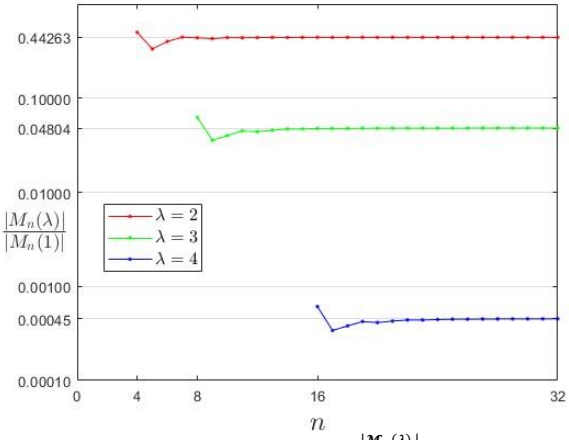
جدول۲: دنباله‌های اعداد و بردارهای چندگانگی مربوط به $\mathcal{M}_{13}(3)$

دنباله $z_1, z_2, \cdots$	بردار چندگانگی
(1,3,4)	(0,0,3,10,0,0,0,0,0,0,0,0,0)
(1,3,1,2)	(0,0,6,1,6,0,0,0,0,0,0,0,0)
(1,3,2,1)	(0,0,5,4,4,0,0,0,0,0,0,0,0)
(1,3,3,0)	(0,0,4,7,2 ,0,0,0,0,0,0,0,0)
(1,3,0,1,1)	(0,0,7,0,2,4,0,0,0,0,0,0,0)
(1,3,1,0,1)	(0,0,6,3,0,4,0,0,0,0,0,0,0)
(1,3,1,1,0)	(0,0,6,2,3,2,0,0,0,0,0,0,0)
(1,3,2,0,0)	(0,0,5,5,1,2,0,0,0,0,0,0,0)
(1,3,0,0,0,1)	(0,0,7,1,1,0,4,0,0,0,0,0,0)
(1,3,0,0,1,0)	(0,0,7,1,0,3,2,0,0,0,0,0,0)
(1,3,0,1,0,0)	(0,0,7,0,3,1,2,0,0,0,0,0,0)
(1,3,1,0,0,0)	(0,0,6,3,1,1,2,0,0,0,0,0,0)
(1,3,0,0,0,0,0)	(0,0,7,1,1,1,1,2,0,0,0,0,0)

قبل از ارائه الگوریتم پیشنهادی این بخش، می‌بایست ابتدا با استفاده از مطالب بخش ۴ و قضیه ۶ (یا یکی دیگر از روش‌های موجود مانند [۱۵، ۱۸]) همه اعضای $\mathcal{M}_n(1)$ و در نتیجه همه کدهای فشرده nتایی را تولید نمود و بعد از بین آن‌ها، کدهایی که همه طول‌کدهایشان بزرگتر یا مساوی مقدار λ است را جدا کرد. واضح است که این‌کار باعث اتلاف منابع می‌شود. ولی با استفاده از الگوریتم پیشنهادی این بخش می‌توان منابع محاسباتی را فقط برای ساخت کدهای موردنظر (متناظر با اعضای $\mathcal{M}_n(\lambda)$) مصرف کرد و با سرعت بیشتری همه آن‌ها را به‌دست آورد. بدیهی است که زمان اجرای هر یک از دو روش مذکور متناسب با تعداد کدهایی است که با آنها ساخته می‌شود. بر این اساس نسبت

$$\frac{|\mathcal{M}_n(\lambda)|}{|\mathcal{M}_n(1)|} \qquad (۶۳)$$

نشان‌گر نسبت *زمان اجرای* دو روش مذکور (با استفاده از الگوریتم پیشنهادی این بخش و بدون استفاده از آن) است. در شکل ۱ مقدار این نسبت به ازای سه مقدار مختلف $\lambda = 2,3,4$ و $2^\lambda \leq n \leq 32$ در مقیاس لگاریتمی رسم شده‌است (در بخش ۱ دیدیم که باید $\lambda \leq \lfloor \log n \rfloor$ برقرار باشد). همان‌طور که انتظار می‌رفت به ازای هر n مشخص، هرچه مقدار λ بزرگتر باشد این نسبت کوچکتر می‌شود و کارایی الگوریتم پیشنهادی بیشتر نمایان می‌گردد. نکته دیگر اینکه به ازای هر λ مشخص، با افزایش n این نسبت به مقدار ثابتی میل می‌کند. این مقادیر ثابت برای $\lambda = 2,3,4$ به ترتیب تقریبا مساوی ۰/۴۴۲۶۳، ۰/۰۴۸۰۴ و ۰/۰۰۰۴۵ هستند. این بدان معناست که مثلا اگر $\lambda = 3$ باشد برای n به‌اندازه کافی بزرگ، یعنی $2^\lambda \leq n$، *زمان اجرا* با الگوریتم پیشنهادی تقریبا ۵ درصد زمان اجرا بدون الگوریتم پیشنهادی است یا به‌عبارت دیگر *سرعت اجرا* با الگوریتم پیشنهادی تقریبا ۲۰ برابر است. برای $\lambda = 4$ این نسبت سرعت به حدود ۲۲۰۰ می‌رسد.



شکل ۱: نسبت $\frac{|\mathcal{M}_n(\lambda)|}{|\mathcal{M}_n(1)|}$ به‌عنوان نسبت *زمان اجرای* تولید کدهای‌فشرده با استفاده از الگوریتم پیشنهادی و بدون استفاده از آن به‌ازای سه مقدار مختلف $\boldsymbol{\lambda = 2,3,4}$ و $\boldsymbol{2^\lambda \leq n \leq 32}$.

مثال۹: برای کدگذاری حروف زبان فارسی (۳۲ حرف به اضافه همزه) باید به دنبال کدهای ۳۳ تایی مناسب باشیم. به طور کلی ۳۳۸۱۸۷۹۴ کد فشرده ۳۳ تایی وجود دارد که بهترین آنها (کد هافمن) به حداقل متوسط طول‌کد ۴/۷۵۸۲ می‌رسد. با این کد اختلاف طول طولانی‌ترین و کوتاه‌ترین کلمه‌کد مساوی ۵-۲=۷ است. همان طور که در جدول۳ مشخص شده با افزایش مقدار λ تعداد کدهای ممکن کاهش می‌یابد و چون انتخاب‌ها محدودتر می‌شوند متوسط طول‌کد افزایش می‌یابد، ولی همان‌طور که انتظار می‌رفت اختلاف طول بزرگ‌ترین و کوچک‌ترین کلمه‌کد و همچنین واریانس طول‌کدها کاهش می‌یابد.

به‌طور خاص برای $\lambda = 4$ فقط ۱۵۲۹۸ کد فشرده وجود دارد که متوسط طول‌کد بهترین آنها ۴/۸۹۵۶ فقط ۰/۱۴ بیت بیشتر از متوسط طول‌کد هافمن است ولی درعوض پراکندگی طول‌کدهای آن به طرز قابل توجهی کمتر است. در حالت غایی $\lambda = 5$ فقط یک کد فشرده با بردار چندگانگی (0,0,0,0,31,2) وجود دارد که متوسط طول کد آن ۵/۰۰۶۹ بیت است. در ستون آخر جدول ۳ نسبت سرعت تولید کدها با الگوریتم پیشنهادی و بدون آن (یعنی $\frac{|\mathcal{M}_n(\lambda)|}{|\mathcal{M}_n(1)|}$) مشخص شده‌است.

طبق [۲۷] مقدار احتمال ۳۲ حرف زبان فارسی به اضافه نشانه همزه به صورت زیر در نظر گرفته شده است

۰/۱۹۹۸,۰/۰۵۴۹,۰/۰۰۸۳,۰/۰۶۱۹,۰/۰۰۱۸,۰/۰۱۳۳,۰/۰۰۴۶,۰/۰۱۱۹,
۰/۰۱۵۹,۰/۰۹۰۳,۰/۰۰۲۶,۰/۰۵۹۰,۰/۰۲۷۲,۰/۰۰۱۲,۰/۰۲۱۸,۰/۰۱۷۲,
۰/۰۰۴۶,۰/۰۰۲۰,۰/۰۰۳۷,۰/۰۰۱۴,۰/۰۰۸۶,۰/۰۰۱۴,۰/۰۰۹۵,۰/۰۰۷۸,
۰/۰۰۴۰,۰/۰۰۹۲,۰/۰۳۳۱,۰/۰۷۱۵,۰/۰۸۹۷,۰/۰۷۳۳,۰/۰۸۱۶,۰/۰۰۵۰,
۰/۰۰۱۹.

۵- نتیجه‌گیری

در این مقاله به مسئله تولید دسته خاصی از کدهای فشرده، که در اینجا به اختصار آنها را کدهای «*مطلوب*» می‌نامیم، پرداخته شد. کدهای *مطلوب* کدهای فشرده‌ای هستند که (برای λ مشخص) همه کلمه‌کدهای آنها λ بیت یا

طولانی‌تر باشند. دیدیم که برای چنین کدهایی بزرگترین طول کد حداکثر $n-2^{\lambda}+\lambda$ (و در نتیجه اختلاف بزرگ‌ترین و کوچک‌ترین طول‌کد حداکثر $n-2^{\lambda}$) می‌شود و حتی برای مقادیر بزرگ n هم می‌توان با انتخاب λ به اندازه کافی بزرگ (با قید کوچکتر یا مساوی $\lfloor \log n \rfloor$) این محدودیت روی طول‌کدها را ایجاد کرد و کدبرداری را با کارایی بالاتر انجام داد.

با توجه به اینکه اگر طول‌کدهای یک کد بدون پیشوند در اختیار باشد ساختن کلمه‌کدها با یک الگوریتم سرراست قابل انجام است، مسئله اصلی مقاله به تولید همه بردار طول‌کدها و به طور معادل تولید همه بردارهای چندگانگی کدهای *مطلوب* تبدیل می‌شود. همچنین دیدیم که چگونه با اعمال توابع ϕ و ψ بر بردارهای چندگانگی کدهای *مطلوب* n تایی می‌توان بردارهای چندگانگی کدهای *مطلوب* $n+1$ تایی را به صورت غیرتکراری ساخت. ثابت شد که هر بردار چندگانگی از اعمال دنباله‌ای از توابع ϕ و ψ بر یک بردار چندگانگی خاص یعنی $S=(1,2,0)$ به دست می‌آید. همچنین شرط لازم و کافی برای این که یک دنباله از توابع ϕ و ψ منجر به تولید یک کد *مطلوب* شود به دست آمد. هر دنباله از توابع ϕ و ψ با یک دنباله محدود از اعداد صحیح نامنفی توصیف شد و درنهایت الگوریتمی برای تولید همه دنباله‌های اعداد که متناظر یک کد *مطلوب* هستند ارائه گردید. کارایی و اهمیت این الگوریتم در آن است که درحین اجرای آن هیچ کدی که *مطلوب* نباشد تولید نمی‌شود و هیچ کد *مطلوبی* هم بیش از یک بار تولید نمی‌شود.

بدون استفاده از الگوریتم پیشنهادی این امکان وجود داشت که ابتدا همه کدهای فشرده n تایی را با استفاده از یکی از روش‌های موجود تولیدکرده و سپس کدهای مطلوب را از بین آنها جدا کنیم. با این‌کار مقداری از منابع محاسباتی هدر می‌رود. از آن‌جا که زمان اجرای الگوریتم‌ها متناسب با تعداد کدهایی است که باید با آنها ساخته شود، کارایی الگوریتم پیشنهادی را می‌توان با نسبت تعداد کدهای مطلوب n تایی به تعداد کل کدهای n تایی سنجید. به ازای $\lambda=2,3,4$ ، با افزایش n این نسبت به سمت مقادیر ثابت ۰/۴۴۲۶۳، ۰/۰۴۸۰۴ و ۰/۰۰۰۴۵ میل می‌کند. به‌طور خاص به ازای $\lambda=3$ با استفاده از الگوریتم پیشنهادی منابع محاسباتی لازم برای تولید کدهای مطلوب تقریبا ۵ درصد حالتی است که از آن استفاده نشود.

جدول۳: انتخاب کد برای الفبای فارسی با ملاحظه اختلاف طول کلمات کد

مقدار λ	تعداد کدهای ممکن	بردار چندگانگی بهینه	اختلاف بزرگ‌ترین و کوچکترین طول‌کد	واریانس طول‌کدهای بهینه	حداقل متوسط طول‌کد ممکن	نسبت سرعت تولید کدهای فشرده مورد نظر (با محدودیت λ روی طول‌کدها) با استفاده از الگوریتم پیشنهادی و بدون استفاده از آن
۱	۳۳۸۱۸۷۹۴	(0,1,0,3,8,19,2)	۵	۰/۹۴۵۱	۴/۷۵۸۲	۱
۲	۱۴۹۶۹۲۳۹	(0,1,0,3,8,19,2)	۵	۰/۹۴۵۱	۴/۷۵۸۲	۲/۲۵
۳	۱۶۲۴۷۳۱	(0,0,2,2,12,15,2)	۴	۰/۸۷۱۲	۴/۸۳۵۶	۲۰/۸۱
۴	۱۵۲۹۸	(0,0,0,1,29,1,2)	۳	۰/۲۹۷۳	۴/۸۹۵۶	۲۲۱۰/۶۶

مراجع

[۱] ثریا عمویی، کمال میرزایی، « فشرده‌سازی تصویر توسط چندی‌سازی برداری مبتنی بر الگوریتم کرم شب‌تاب بهبودیافته »، مجله مهندسی برق دانشگاه تبریز، جلد ۴۹، شماره ۲، صفحات ۶۹۳-۷۰۷، ۱۳۹۸.

[۲] محمود طوماری، سپیده جباری، « فشرده‌سازی سیگنال‌های ژنوم با کمک حسگری فشرده و کاربرد آن در مقایسه دنباله‌های ژنی»، مجله مهندسی برق دانشگاه تبریز، جلد ۴۹، شماره ۱، صفحات ۳۰۷-۳۱۶، ۱۳۹۸.

[۳] مریم مگری، هادی گرایلو، « فشرده‌سازی سیگنال‌های الکترومایوگرام مبتنی بر هموارسازی به کمک تکنیک پیش‌تاکید-واتاکید»، مجله مهندسی برق دانشگاه تبریز، جلد ۴۹، شماره ۴، صفحات ۱۸۳۷-۱۸۴۸، ۱۳۹۸.

[4] T. M. Cover, J. A. Thomas, "Elements of information theory", Wiley, New York, 1991.

[5] Y. G. Chen, C. Elsholtz, L. L. Jiang, "Egyptian fractions with restrictions", Acta Arithmetica, vol. 154, no. 2, pp. 109–123, 2012.

[6] F. N. Castro, "On the Equation $\sum_{i=1}^{n}\frac{1}{x_i}=1$ in Distinct Odd or Even Numbers", Contributions to Discrete Mathematics, vol. 11, no. 2, pp. 63-78, 2017.

[7] J. Leeuwen, "On the construction of Huffman trees", Proceedings of Third International Colloquium on Automata, Languages and Programming, pp. 382-410, 1976.

[8] J. Rissanen, "Minimax codes for finite alphabets", IEEE Transactions on Information Theory, vol. 24, no. 3, pp. 389–392, 1978.

[9] A. Mahmood, A. B. Wagner, "Minimax Rate-Distortion", IEEE Transactions on Information Theory, vol. 69, no. 12, pp. 7712–7737, 2023.

[10] M. Khosravifard, H. Saidi, M. Esmaeili, T.A. Gulliver, "The minimum average code for finite memoryless monotone sources", IEEE Transactions on Information Theory, vol. 53, no. 3, pp. 957–977, 2007.

[11] H. Narimani, M. Khosravifard, "A new code for encoding all monotone sources with a fixed large alphabet size", IEEE Transactions on Information Theory, vol. 66, no. 3, pp. 1474–1481, 2020.

[12] S. Banchhor, R. Gajjala, Y. Sabharwal, S. Sen, "Generalizations of Length Limited Huffman Coding for Hierarchical Memory Settings", arXiv preprint arXiv:2010.05005, 2021.

[13] G. Lakhani, "Modified JPEG Huffman coding", IEEE Transactions on Image Processing, vol. 12, no. 2, pp.159-169, 2003.

[14] L. L. Larmore, D. S. Hirschberg, "A fast algorithm for optimal length-limited Huffman codes", Journal of the ACM, vol. 37, no. 3, pp. 464–473, 1990.

[15] M. Khosravifard, M. Esmaeili, H. Saidi, T. Aron Gulliver, "A tree based algorithm for generating all possible binary compact codes with *N* codewords," IEICE Transactions Fundamentals, vol. E86-A, no. 10, pp. 2510-2516, 2003.

[16] S. Even, A. Lempel, "Generation and enumeration of all solutions of the characteristic sum condition", Information and Control, vol. 21, pp. 476-482, 1972.

[17] P. Flajolet, H. Prodinger, "Level number sequences for trees", Discrete Mathematics, vol. 65, no. 2, pp. 149–156, 1987.

[18] H. Narimani, M. Khosravifard, "The supertree of the compact codes", Proceedings of International Symposium on Telecommunications (IST), pp. 649-655, 2008.

[19] D. Hoffman, P. Johnson, N. Wilson, "Generating Huffman sequences", Journal of Algorithms, vol. 54, pp. 115-121, 2005.

[20] E. Norwood, "The number of different possible compact codes", IEEE Transactions on Information Theory, vol. 13, no. 4, pp. 613-616, 1967.

[21] C. Elsholtz, C. Heuberger, H. Prodinger, "The number of Huffman codes, compact trees, and sums of unit fractions", IEEE Transactions on Information Theory, vol. 59, no. 2, pp. 1065-1075, 2013.

[22] C. Elsholtz, C. Heuberger C, D. Krenn, "Algorithmic counting of nonequivalent compact Huffman codes", Applicable Algebra in Engineering, Communication and Computing, Jan. 2023 Available online at: https://doi.org/10.1007/s00200-022-00593-0.

[23] J. Paschke, J. Burkert, R. Fehribach, "Computing and estimating the number of n-ary Huffman sequences of a specified length", Discrete Mathematics, vol. 311, pp. 1-7, 2011.

[24] K. Hashimoto, K. Iwata, H. Yamamoto, "Enumeration and Coding of Compact Code Trees for Binary AIFV Codes", Proceedings IEEE International Symposium on Information Theory (ISIT), pp. 1527-1531, Jul. 2019.

[25] J. Komlos, W. Moser, T. Nemetz, "On the asymptotic number of prefix codes", Mitteilungen aus dem Mathematischen Seminar Giessen, Heft 165, Teil III, pp. 35-48, 1984.

[26] D. Krenn, S. Wagner, "Compositions into powers of *b*: asymptotic enumeration and parameters", Algorithmica, vol. 75, pp. 606-631, 2016.

[۲۷] مهرزاد منصوری «بررسی بسامد نویسه‌های فارسی و مناسبت جایگاه آنها بر صفحه کلید رایانه‌ها»، مجله زبان‌شناسی و گویش‌های خراسان، شماره ۷، صفحات ۱۰۹-۱۲۹ ، پاییز و زمستان ۱۳۹۱