様式 8

論 文 内 容 要 旨

報告番号	甲 先 第 297 号	氏 名	神田峻介
学位論文題目	Space- and Time-Efficient String Dictionaries 空間効率と時間効率の良い文字列辞書		

In modern computer science, the management of massive data is a fundamental problem because the amount of data is growing faster than we can easily handle them. Such data are often represented as strings such as documents, Web contents and genomics data; therefore, data structures and algorithms for space-efficient string processing have been developed by many researchers. In this thesis, we focus on a string dictionary that is an in-memory data structure for storing a set of strings. It has been traditionally used to manage vocabulary in natural language processing and information retrieval. The size of the dictionaries is not problematic because of Heaps' Law. However, string dictionaries in recent applications, such as Web search engines, RDF stores, geographic information systems and bioinformatics, need to handle very large datasets. As the space usage of string dictionaries is a significant issue in those applications, it is necessary to develop space-efficient data structures.

If limited to static applications, existing data structures have already achieved very high space efficiency by exploiting succinct data structures and text compression techniques. For example, state-of-the-art string dictionaries can be implemented in space up to 5% of the original dataset size. However, there remain trade-off problems among space efficiency, lookup-time performance and construction costs. For example, Re-Pair that is a powerful compression technique for a text is well used to obtain high space efficiency and fast lookup operations, but the time and space needed during construction are not practical for very large datasets.

In our work, we attempt two approaches to solve the problems. One develops novel string dictionaries based on double-array tries. The double-array trie can support the fastest lookup operation, but its space usage is very large. We propose how to improve the disadvantage. The novel string dictionaries can support very fast lookup operations, although the space usage is somewhat larger compared to existing compressed ones. The other addresses to improve the high construction costs of applying Re-Pair by introducing an alternative compression strategy using dictionary encoding. Our strategy enables lightweight construction, while providing competitive space efficiency and operation speed.

If allowing dynamic operations such as insertion and deletion, there are some sophisticated data structures. While they improve the space efficiency by reducing pointer overheads, they still consume much larger space than the static string dictionaries because some redundancy for dynamic update is needed. In fact, the number of dynamic applications handling very large string dictionaries is fewer than that of static ones; however, a part of applications as in Web crawler and Semantic Web need to compact dynamic string dictionaries. In this thesis, we propose a new data structure through path decomposition that is a technique for constructing cache-friendly trie structures. Although the path decomposition has been utilized to static dictionary implementation, we adopt it for dynamic dictionary construction with a different approach. From experiments, we show that our data structure can implement the smallest dynamic string dictionary.

We also address problems of dynamic double-array trie dictionaries. The dynamic dictionaries are well-used as in stream text processing and search engines, but the space efficiency tends to decrease with key updates. Although we can still maintain high efficiency using existing methods, these methods have practical problems of time and functionality. In this thesis, we present several practical rearrangement methods to solve these problems. We show that the proposed methods can support significantly faster rearrangement via read-world datasets.

As all our methods are heuristic and practical, we sometimes do not provide the theoretical guarantees; however, we always show the practical performances through careful experiments using large real-world datasets. Furthermore, most of our implementations are provided as open source libraries under the MIT License.