Strings

Suffix Array – Definição e Construção

Prof. Edson Alves - UnB/FGA

2019

Sumário

Definição

Suffix Array

- Seja S uma string de tamanho N
- O i-ésimo sufixo de S é a substring que inicia no índice i e termina no índice N, isto é, $S[i..N]$
- Um vetor de sufixos (*suffix array*) $s_A(S)$ de S é um vetor de inteiros que representam os índices iniciais i dos prefixos de S, após a ordenação lexicográfica dos mesmos
- Os vetores de sufixos são usados em problemas que envolvem buscas em strings
- Estes vetores foram propostos por Udi Manber e Gene Myers

Exemplo de *suffix array*

i	$S[i..N]$
1	"abacaxi"
2	"bacaxi"
3	"acaxi"
4	"caxi"
5	"axi"
6	"xi"
7	"i"

j	$s_A[j]$	$S[s_A[j]..N]$
1	1	"abacaxi"
2	3	"acaxi"
3	5	"axi"
4	2	"bacaxi"
5	4	"caxi"
6	7	"i"
7	6	"xi"

Construção de $s_A(S)$ com complexidade $O(N^2 \log N)$

- A construção de $s_A(S)$ diretamente de sua definição tem complexidade $O(N^2 \log N)$, onde N é o tamanho da string S
- Primeiramente é preciso construir um vetor ps de pares $(S[i..N], i)$
- Em seguida, este vetor deve ser ordenado
- O algoritmo de ordenação tem complexidade $O(N \log N)$, e como as comparações entre as substrings tem complexidade $O(N)$, a complexidade da solução é $O(N^2 \log N)$
- Observe que, após ordenado o vetor ps, o vetor de sufixos $s_A(S)$ é composto apenas pelos índices (segundo elemento de cada par), não sendo necessário armazenar os prefixos explicitamente
- Assim a complexidade de memória é $O(N)$

Construção *naive* de $s_A(S)$

```
5  vector<int> suffix_array(const string& s)
6  {
7      using si = pair<string, int>;
8
9      vector<si> ss(s.size());
10
11     for (size_t i = 0; i < s.size(); ++i)
12         ss[i] = si(s.substr(i), i);
13
14     sort(ss.begin(), ss.end());
15
16     vector<int> sa(s.size());
17
18     for (size_t i = 0; i < s.size(); ++i)
19         sa[i] = ss[i].second;
20
21     return sa;
22 }
```

Observações sobre a construção *naive* de $s_A(S)$

- Embora a construção apresentada seja de fácil entendimento e implementação, ela não é aplicável em strings grandes ($N \geq 10^4$)
- É possível construir $s_A(S)$ com complexidade $O(N \log N)$, porém tanto a implementação é mais sofisticada
- Além disso, a terminologia e os conceitos utilizados para esta redução na complexidade não são triviais
- Estes conceitos e esta construção serão apresentados a seguir

Construção do vetor de sufixos em $O(N \log N)$

Conceitos preliminares

Referências

1. CP Algorithms. Suffix Array, acesso em 06/09/2019.
2. **CROCHEMORE**, Maxime; **RYTTER**, Wojciech. *Jewels of Stringology: Text Algorithms*, WSPC, 2002.
3. **HALIM**, Steve; **HALIM**, Felix. *Competitive Programming 3*, Lulu, 2013.