

Mảng hậu tố (Suffix Array)

Mảng hậu tố (Suffix Array)

Mảng Hậu Tố là một CTDL giúp **sort** các **hậu tố** theo **thứ tự từ điển**.

Mảng này chứa các số nguyên, khởi đầu của các hậu tố.

Có 2 cách để xây dựng một mảng hậu tố:

1. **Thuật toán không xác định**: Sử dụng thuật toán **Rabin-Karp** và kiểm tra nếu một hậu tố có thứ tự từ điển nhỏ hơn một hậu tố khác, tìm **mảng tiền tố chung lớn nhất (LCP)**, sau đó sử dụng **Tìm Kiếm Nhị Phân** và **hàm băm (Hash)** và kiểm tra ký tự tiếp theo sau **LCP** của chúng.

Code C++:

```

1 namespace HashSuffixArray {
2
3     const int MAXN = 1 << 21;
4
5     typedef unsigned long long hash;
6     const hash BASE = 137;
7
8     int N;
9     char * S;
10    int sa[MAXN];
11    hash h[MAXN], hPow[MAXN];
12
13    #define getHash(lo, size) (h[lo] - h[(lo) + (size)] * hPow[size])
14
15    inline bool sufCmp(int i, int j)
16    {
17        int lo = 1, hi = min(N - i, N - j);
18        while (lo <= hi)
19        {
20            int mid = (lo + hi) >> 1;
21            if (getHash(i, mid) == getHash(j, mid))
22                lo = mid + 1;
23            else
24                hi = mid - 1;
25        }
26        return S[i + hi] < S[j + hi];
27    }
28
29 }

```

```

29
30     void buildSA()
31     {
32         N = strlen(S);
33         hPow[0] = 1;
34         for (int i = 1; i <= N; ++i)
35             hPow[i] = hPow[i - 1] * BASE;
36         h[N] = 0;
37         for (int i = N - 1; i >= 0; --i)
38             h[i] = h[i + 1] * BASE + S[i], sa[i] = i;
39
40         stable_sort(sa, sa + N, sufCmp);
41     }
42 } // end namespace HashSuffixArray

```

2. **Thuật toán xác định:** Sort log(Độ dài lớn nhất) bước, với bước thứ i (tính từ 0), sort chúng theo 2^i ký tự đầu tiên và đưa hậu tố có cùng tiền tố với 2^i ký tự vào cùng một bucket.

Code:

```

1  /*
2  Suffix array  $O(n \lg^2 n)$ 
3  LCP table  $O(n)$ 
4  */
5  #include <cstdio>
6  #include <algorithm>
7  #include <cstring>
8
9  using namespace std;
10
11 #define REP(i, n) for (int i = 0; i < (int)(n); ++i)
12
13 namespace SuffixArray
14 {
15     const int MAXN = 1 << 21;
16     char * S;
17     int N, gap;
18     int sa[MAXN], pos[MAXN], tmp[MAXN], lcp[MAXN];
19
20     bool sufCmp(int i, int j)
21     {
22         if (pos[i] != pos[j])
23             return pos[i] < pos[j];
24         i += gap;
25         j += gap;
26         return (i < N && j < N) ? pos[i] < pos[j] : i > j;
27     }
28
29     void buildSA()
30     {
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```
31     N = strlen(S);
32     REP(i, N) sa[i] = i, pos[i] = S[i];
33     for (gap = 1;; gap *= 2)
34     {
35         sort(sa, sa + N, sufCmp);
36         REP(i, N - 1) tmp[i + 1] = tmp[i] + sufCmp(sa[i], sa[i + 1]);
37         REP(i, N) pos[sa[i]] = tmp[i];
38         if (tmp[N - 1] == N - 1) break;
39     }
40 }
41
42 void buildLCP()
43 {
44     for (int i = 0, k = 0; i < N; ++i) if (pos[i] != N - 1)
45     {
46         for (int j = sa[pos[i] + 1]; S[i + k] == S[j + k];)
47             ++k;
48         lcp[pos[i]] = k;
49         if (k)--k;
50     }
51 }
52 } // end namespace SuffixArray
```

Source: [mukel](#) 

Tài liệu tham khảo:

- [Codeforces](#) 

Được cung cấp bởi [Wiki.js](#)