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++:

```cpp
namespace HashSuffixArray {

    const int MAXN = 1 << 21;

    typedef unsigned long long hash;
    const hash BASE = 137;

    int N;
    char * S;
    int sa[MAXN];
    hash h[MAXN], hPow[MAXN];

    #define getHash(lo, size) (h[lo] - h[(lo) + (size)] * hPow[size])

    inline bool sufCmp(int i, int j)
    {
        int lo = 1, hi = min(N - i, N - j);
        while (lo <= hi)
        {
            int mid = (lo + hi) >> 1;
            if (getHash(i, mid) == getHash(j, mid))
                lo = mid + 1;
            else
                hi = mid - 1;
        }
        return S[i + hi] < S[j + hi];
    }

```

```
    void buildSA()
    {
        N = strlen(S);
        hPow[0] = 1;
        for (int i = 1; i <= N; ++i)
            hPow[i] = hPow[i - 1] * BASE;
        h[N] = 0;
        for (int i = N - 1; i >= 0; --i)
            h[i] = h[i + 1] * BASE + S[i], sa[i] = i;

        stable_sort(sa, sa + N, sufCmp);
    }

} // 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:

```
/*
Suffix array O(n lg^2 n)
LCP table O(n)
*/
#include <cstdio>
#include <algorithm>
#include <cstring>

using namespace std;

#define REP(i, n) for (int i = 0; i < (int)(n); ++i)

namespace SuffixArray
{
    const int MAXN = 1 << 21;
    char * S;
    int N, gap;
    int sa[MAXN], pos[MAXN], tmp[MAXN], lcp[MAXN];

    bool sufCmp(int i, int j)
    {
        if (pos[i] != pos[j])
            return pos[i] < pos[j];
        i += gap;
        j += gap;
        return (i < N && j < N) ? pos[i] < pos[j] : i > j;
    }

    void buildSA()
    {
```

```
        N = strlen(S);
        REP(i, N) sa[i] = i, pos[i] = S[i];
        for (gap = 1;; gap *= 2)
        {
            sort(sa, sa + N, sufCmp);
            REP(i, N - 1) tmp[i + 1] = tmp[i] + sufCmp(sa[i], sa[i + 1]);
            REP(i, N) pos[sa[i]] = tmp[i];
            if (tmp[N - 1] == N - 1) break;
        }
    }

    void buildLCP()
    {
        for (int i = 0, k = 0; i < N; ++i) if (pos[i] != N - 1)
        {
            for (int j = sa[pos[i] + 1]; S[i + k] == S[j + k];)
            ++k;
            lcp[pos[i]] = k;
            if (k)--k;
        }
    }
} // end namespace SuffixArray
```

Source: mukel

Tài liệu tham khảo:

- Codeforces

Được cung cấp bởi Wiki.js