SA

```cpp
struct SA{
    vector<int> sa, rk, oldrk, id, key1, cnt, ht;
    vector<vector<int>> st;
    int i, m = 127, p, w;
    bool cmp(int x, int y, int w) {
        return oldrk[x] == oldrk[y] && oldrk[x + w] == oldrk[y + w];
    }// key1[i] = rk[id[i]]（作为基数排序的第一关键字数组）
    int n;
    SA(string s)
    {
        n = s.size() - 1;
        oldrk.resize(2 * n + 5);
        sa.resize(n + 2);
        rk.resize(n + 2);
        id.resize(n + 2);
        key1.resize(n + 2);
        cnt.resize(max(n + 5, 130ll));
        for (i = 1; i <= n; ++i) ++cnt[rk[i] = s[i]];
        for (i = 1; i <= m; ++i) cnt[i] += cnt[i - 1];
        for (i = n; i >= 1; --i) sa[cnt[rk[i]]--] = i;
        for (w = 1;; w <<= 1, m = p) {  // m=p 就是优化计数排序值域
            for (p = 0, i = n; i > n - w; --i) id[++p] = i;
            for (i = 1; i <= n; ++i)
                if (sa[i] > w) id[++p] = sa[i] - w;
            fill(cnt.begin(), cnt.end(), 0);
            for (i = 1; i <= n; ++i) ++cnt[key1[i] = rk[id[i]]];
            // 注意这里px[i] != i，因为rk没有更新，是上一轮的排名数组

            for (i = 1; i <= m; ++i) cnt[i] += cnt[i - 1];
            for (i = n; i >= 1; --i) sa[cnt[key1[i]]--] = id[i];
            for(int i = 1 ; i <= n ; i++)
            {
                oldrk[i] = rk[i];
            }
            for (p = 0, i = 1; i <= n; ++i)
                rk[sa[i]] = cmp(sa[i], sa[i - 1], w) ? p : ++p;
            if (p == n) {
                break;
            }
        }
        // height数组构建
        ht.resize(n + 2);
        int k = 0;
        for(int i = 1 ; i <= n ; i++ )
        {
            k = max(k - 1, 0ll);
            if(rk[i] == 1) continue;
            int j = sa[rk[i] - 1];
            while(s[i + k] == s[j + k]) k++;
            ht[rk[i]] = k;
        }

        // LCPst表构建
        st.resize(24);
        st[0].resize(n + 5);
        for(int i = 1 ; i <= n ; i++ )
        {
            st[0][i] = ht[i];
        }
        for(int j = 1 ; j <= 22 ; j++ )
        {
            st[j].resize(n + 5);
            for(int i = 1 ; i + (1 << j) - 1 <= n ; i++ )
            {
                st[j][i] = min(st[j - 1][i], st[j - 1][i + (1ll << j - 1)]);
            }
        }


    }
    int LCP(int u, int v)
    {
        if(u == v) return n - u + 1;
        if(rk[u] > rk[v]) swap(u, v);
        int l = rk[u] + 1, r = rk[v];
        int len = __lg(r - l + 1);
        return min(st[len][l], st[len][r - (1 << len) + 1]);
    }
};
```