# suffixarray

suffixarrayb通过使用内存中的后缀树实现了对数级时间消耗的子字符串搜索。

用法举例：

// 创建数据的索引  
index := suffixarray.New(data)// 查找切片s  
offsets1 := index.Lookup(s, -1) // 返回data中所有s出现的位置  
offsets2 := index.Lookup(s, 3) // 返回data中最多3个所有s出现的位置

## type Index

// Index implements a suffix array for fast substring search.  
type Index struct {  
 data []byte  
 sa ints // suffix array for data; sa.len() == len(data)  
}

Index类型实现了用于快速子字符串搜索的后缀数组。

## func New

// New creates a new Index for data.  
// Index creation time is O(N) for N = len(data).  
func New(data []byte) \*Index {  
 ix := &Index{data: data}  
 if len(data) <= maxData32 {  
 ix.sa.int32 = make([]int32, len(data))  
 text\_32(data, ix.sa.int32)  
 } else {  
 ix.sa.int64 = make([]int64, len(data))  
 text\_64(data, ix.sa.int64)  
 }  
 return ix  
}

使用给出的[]byte数据生成一个Index，时间复杂度O(Nlog(N))。

## func (\*Index) Bytes

// Bytes returns the data over which the index was created.  
// It must not be modified.  
//  
func (x \*Index) Bytes() []byte {  
 return x.data  
}

返回创建x时提供的[]byte数据，注意不能修改返回值。

## func (\*Index) Read

// Read reads the index from r into x; x must not be nil.  
func (x \*Index) Read(r io.Reader) error

从r中读取一个index写入x，x不能为nil。

## func (\*Index) Write

// Write writes the index x to w.  
func (x \*Index) Write(w io.Writer) error

将x中的index写入w中，x不能为nil。

## func (\*Index) Lookup

func (x \*Index) Lookup(s []byte, n int) (result []int)

返回一个未排序的列表，内为s在被索引为index的切片数据中出现的位置。如果n<0，返回全部匹配；如果n==0或s为空，返回nil；否则n为result的最大长度。时间复杂度O(log(N)\*len(s) + len(result))，其中N是被索引的数据的大小。

source := []byte("hello world, hello china")  
index := suffixarray.New(source)  
offsets := index.Lookup([]byte("hello"), -1)  
sort.Ints(offsets)  
fmt.Printf("%v", offsets) //输出：[0 13]

## func (\*Index) FindAllIndex

func (x \*Index) FindAllIndex(r \*regexp.Regexp, n int) (result [][]int)

返回一个正则表达式r的不重叠的匹配的经过排序的列表，一个匹配表示为一对指定了匹配结果的切片的索引（相对于x.Bytes())。如果n<0，返回全部匹配；如果n==0或匹配失败，返回nil；否则n为result的最大长度。