STRING ALGORITHMS

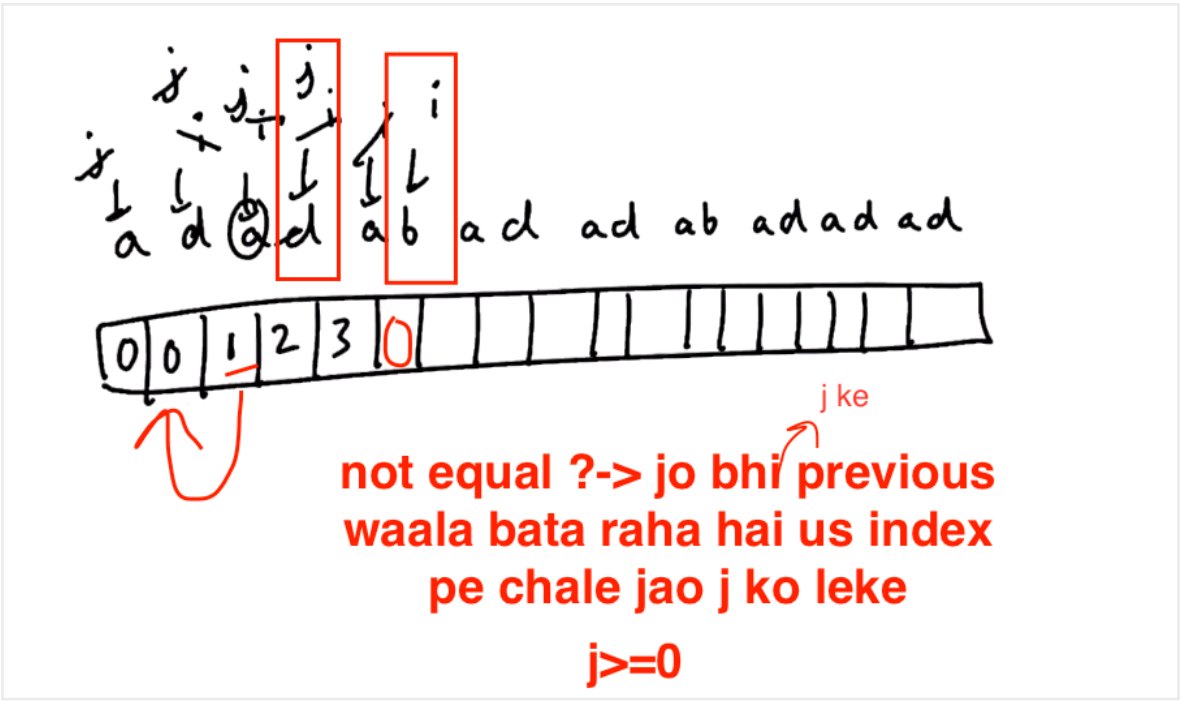
- Does the pattern exists in string ?
- #1 Naive O(M*N)

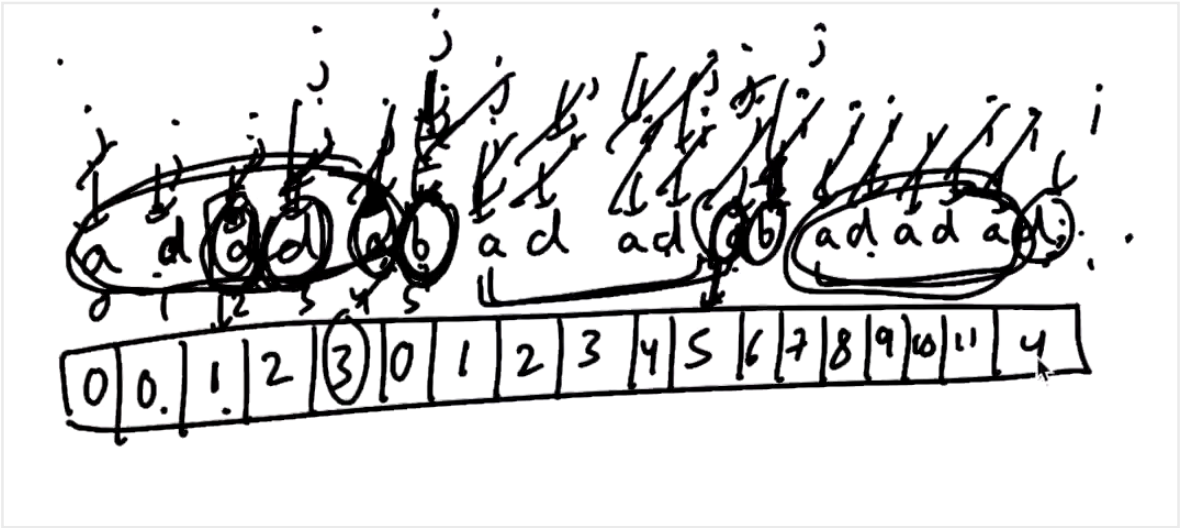
patternmatching.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;

bool isMatching(string s,string p){

    int n = s.length();
    int m = p.length();

    for(int i=0;i<=(n-m);i++){
        bool isFound = true;
        for(int j=0;j<m;j++){

            if(s[i+j] != p[j]){
                isFound = false;
                break;
            }
        }
        if(isFound == true){
            return true;
        }
    }
    return false;
}
int main(){

    string s,p;
    cin >> s;
    cin >> p;

    cout << isMatching(s,p) << endl;
    return 0;
}
```

- #2 KMP O(M+N)

- in the pattern is there a longest prefix which is also a suffix ‘a’ suffix
- we need to know to at every index longest prefix which is also a suffix
- Finding LPS
    - if not match and j==0 i=0 and i++
    - if matches -> ans= j+1 (index value of j +1)
    - if not match and j!=0 :

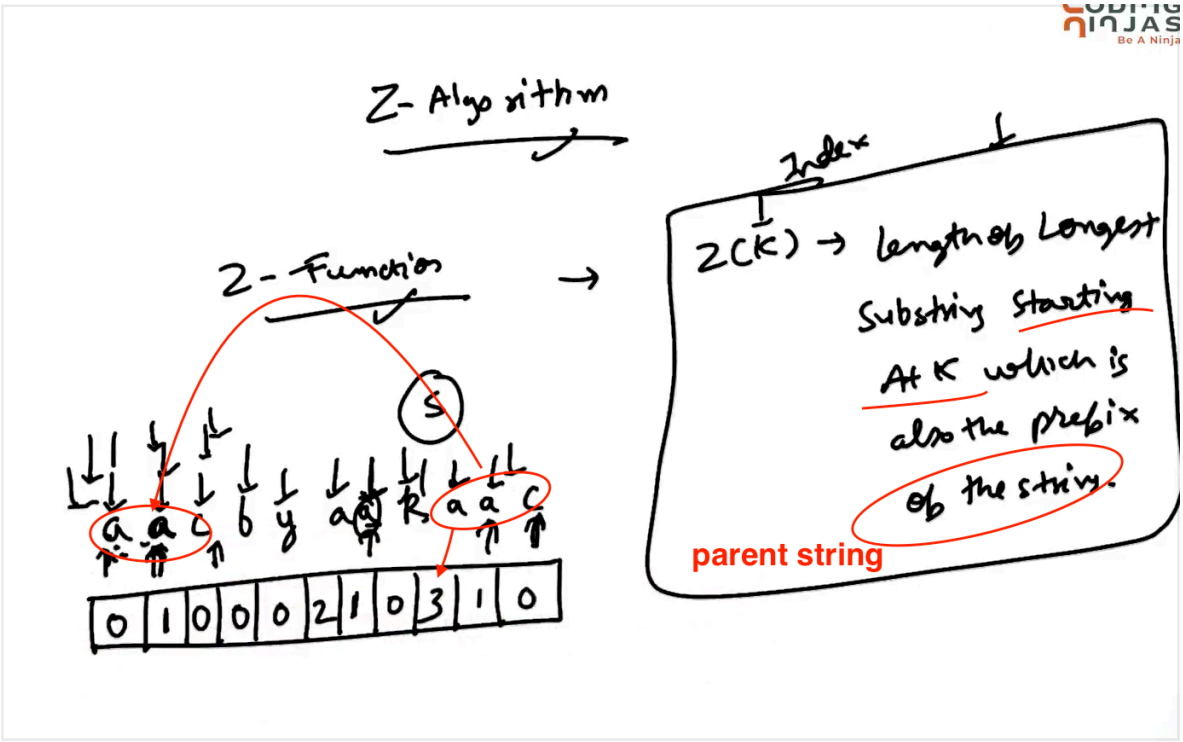



a d a d a b a d a d a b a d a d a b
0 0 1 2 3 0 1 2 3 4 5 6 7 8 9 10 11 4

```cpp
#include<bits/stdc++.h>
using namespace std;


int* getLps(string pattern){

    int len = pattern.length();
    int* lps = new int[len];

    lps[0] = 0;
    int i = 1;
    int j = 0;

    while(i<len){

        if(pattern[i] == pattern[j]){
            lps[i] = j + 1;
            j++;
            i++;
        }else{
            if(j!=0){
                j = lps[j-1];
            }else{
                lps[i] = 0;
                i++;
            }
        }
    }
    return lps;
}
```

```
31 bool kmpSearch(string text,string pattern){
32
33     int lenText = text.length();
34     int lenPat = pattern.length();
35
36     int i = 0;
37     int j = 0;
38
39     int* lps = getLps(pattern);
40     while(i<lenText && j<lenPat){
41
42         if(text[i] == pattern[j]){
43             i++;
44             j++;
45         }else{
46             if(j!=0){
47                 j = lps[j-1];
48             }else{
49                 i++;
50             }
51         }
52     }
53     if(j==lenPat){
54         return true;
55     }
56     return false;
57 }
```

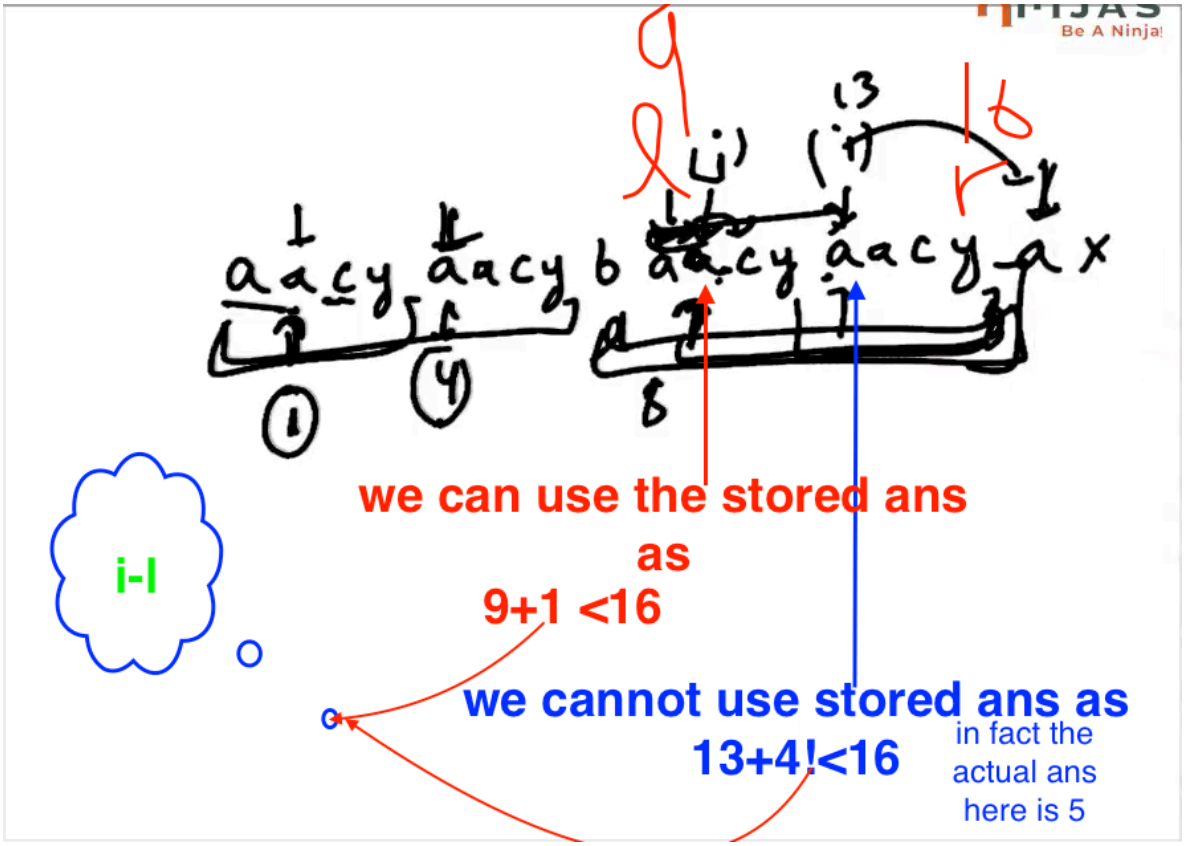
##. Z ALGORITHM



where z[k]== pattern.length-> pattern found

```
#include<bits/stdc++.h>
using namespace std;

void buildZ(int* Z,string str){

}
void searchString(string text,string pattern){
    string str = pattern+"$" +text;
    int n = str.length();
    int* Z = new int[n]();
    buildZ(Z,str);
    for(int i=0;i<n;i++){
        if(Z[i] == pattern.length()){
            cout << i-pattern.length()-1 <<endl;
        }
    }
}
int main(){
    string text = "abcdsafbcdfasbcda";
    string pattern = "bcd";
    searchString(text,pattern);
    return 0;
}
```

index + z[k]<r. you know the answer already its within boundary r



when z[k]<=(r-i) we know the ans as we are ensured ki further elements match nahin honge , jo last ans me hue wohi yahan bhi honge

```
4  void buildZ(int* Z,string str){
5      int l=0;
6      int r =0;
7
8      int n = str.length();
9      for(int i=1;i<n;i++){
10
11         if(i>R){
12             // i does not lie between l and r
13             // Z for this doesnt exist
14             l=i;
15             r=i;
16             while(r<n && str[r-l] == str[r]){
17                 r++;
18             }
19             Z[i] = r-l;
20             r--;
21         }else{
22             int k = i-l;
23             if(Z[k]<=r-i){
24                 // it lies between l and r
25                 // Z will exist previously
26                 Z[i] = Z[k];
```

```
27             }else{
28
29                 //Some part of Z is already included
30                 // You have to start matching further
31                 l=i;
32                 while(r<n && str[r-l] == str[r]){
33                     r++;
34                 }
35                 Z[i] = r-l;
36                 r--;
37             }
38         }
39
40     }
41 }
```