| **NAME:** | Kunal Bhatia |
| --- | --- |
| **UID:** | 2021300010 |
| **SUBJECT** | Design and Analysis of Algorithms |
| **EXPERIMENT NO:** | 10 |
| **AIM:** | To implement Naive and Rabin Carp string matching algorithms |
| **Algorithm:** | **Naive string matching Algorithm**  1. n ← length [T]  2. m ← length [P]  3. for s ← 0 to n -m  4. do if P [1.....m] = T [s + 1....s + m]  5. then print "Pattern occurs with shift" s  **Rabin Karp string matching Algorithm**  1. n ← length [T]  2. m ← length [P]  3. h ← dm-1 mod q  4. p ← 0  5. t0 ← 0  6. for i ← 1 to m  7. do p ← (dp + P[i]) mod q  8. t0 ← (dt0+T [i]) mod q  9. for s ← 0 to n-m  10. do if p = ts  11. then if P [1.....m] = T [s+1.....s + m]  12. then "Pattern occurs with shift" s  13. If s < n-m  14. then ts+1 ← (d (ts-T [s+1]h)+T [s+m+1])mod q |
| **Code** | **Naive string matching algorithm**  **Source Code**  #include <iostream>  #include <string>  using namespace std;  void naiveSearch(string pattern, string text)  {  int patternLength = pattern.length();  int textLength = text.length();  int i, j;    for (i = 0; i <= textLength - patternLength; i++) {  for (j = 0; j < patternLength; j++) {  if (text[i + j] != pattern[j])  break;  }  if (j == patternLength)  cout << "\nPattern found at index " << i << endl;  }  }    int main()  {  string text;  string pattern;    cout<<"\nEnter the string :";  getline(cin, text);  cout<<"\nEnter the pattern you want to search :";  getline(cin, pattern);  naiveSearch(pattern, text);  cout<<endl;  return 0;  } |
| **Output 1:** |  |
| **Code Part 2:** | **Rabin Karp Algorithm**  **Source Code**  #include <stdio.h>  #include <string.h>  #define d 256  #define q 101  int rabin\_karp(char\* text, char\* pattern) {  int text\_length = strlen(text);  int pattern\_length = strlen(pattern);  int i, j;  int pattern\_hash = 0;  int text\_hash = 0;  int h = 1;    for (i = 0; i < pattern\_length - 1; i++) {  h = (h \* d) % q;  }    for (i = 0; i < pattern\_length; i++) {  pattern\_hash = (d \* pattern\_hash + pattern[i]) % q;  text\_hash = (d \* text\_hash + text[i]) % q;  }    for (i = 0; i <= text\_length - pattern\_length; i++) {    if (text\_hash == pattern\_hash) {    for (j = 0; j < pattern\_length; j++) {  if (text[i+j] != pattern[j]) {  break;  }  }    if (j == pattern\_length) {  return i;  }  }    if (i < text\_length - pattern\_length) {  text\_hash = (d \* (text\_hash - text[i] \* h) + text[i+pattern\_length]) % q;    if (text\_hash < 0) {  text\_hash += q;  }  }  }    return -1;  }  int main() {  char text[1000], pattern[1000];    printf("\nEnter the string : ");  fgets(text, 1000, stdin);  printf("\nEnter the pattern to search for : ");  fgets(pattern, 1000, stdin);    text[strcspn(text, "\n")] = 0;  pattern[strcspn(pattern, "\n")] = 0;    int result = rabin\_karp(text, pattern);  if (result == -1) {  printf("\nPattern not found in text.\n");  } else {  printf("\nPattern found in text starting at index %d.\n\n", result);  }      return 0;  } |
| **Output 2:** |  |
| **Conclusion:** | Thus we have implemented Naive and Rabin Karp String matching algorithms |