P1:

a)

bbacbbcb

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| a | 0 | 0 | 3 | 0 | 0 | 0 | 3 | 0 |
| b | 1 | 2 | 2 | 1 | 5 | 6 | 2 | 7/match |
| c | 0 | 0 | 0 | 4 | 0 | 0 | 7 | 0 |

b)

bbacbbcb

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | match |
|  | 0 | 1 | 0 | 0 | 1 | 2 | 0 | 1 |

c)

P2:

This can be down by using KMP-automaton method, searching pattern in string that is repeated once. For example, if inputs are “alloy” and “loyal”, the algorithm should return true if we can find “alloy” as a substring of “loy**alloy**al”.

bool shift(string w, string x){

//KMP(T,P) is the same algorithm as discussed in class.

if (KMP(w, x+x) == false){

return false;

}

return true;

}

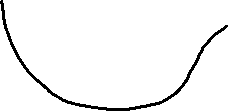
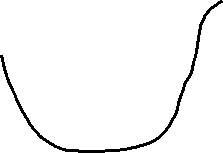
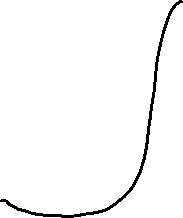
If we can find in the repeated string, it means the tails of x is the start of w, and the beginning of x is the tail of w. Since w and x are of the same length, length of matched head plus tail is the length of x. In the example at the beginning, “al” is the tail of x and head of w, “loy” is the tail of w and head of x. We can shift “loy” part, and we get xfrom w.

The running time of this algorithm is .

P3:

a)

tendergender



b)

the repeated pattern is at index 1 and index 7 with length 5.

c)

P4

a)

Assume towards contradiction that for every compression algorithm A and there is an input for which . Then we can compress and get with . We should be able to infinitely repeat this process and get the compress length real small. However, since the length is integer, the length will keep decreasing and eventually become negative, which creates a contradiction. Hence the original statement is true.

b)

P5:

a)

T=bananablues

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| letter | a | b | e | l | n | s | u |
| frequency | 3 | 2 | 1 | 1 | 2 | 1 | 1 |

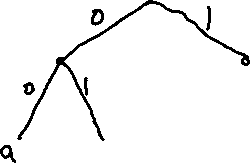


Table:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| letter | a | b | e | l | n | s | u |
| code | 00 | 010 | 0110 | 0111 | 10 | 110 | 111 |

Code: 010 00 10 00 10 00 010 0111 111 0110 110

010001000100001001111110110110

b)

100010001001011001111011011111110010

Decodes to:

10 00 10 00 10 010 110 0111 10 110 111 111 10 010

nananbslnsuunb