

1. printLCS(int[][] llcs)

i 🡨 n – 1

j 🡨 m – 1

while i > 0 and j > 0 do

while llcs[i][j] == llcs[i-1][j] do

i 🡨 i – 1

while llcs[i][j] == llcs[i][j-1] do

j 🡨 j – 1

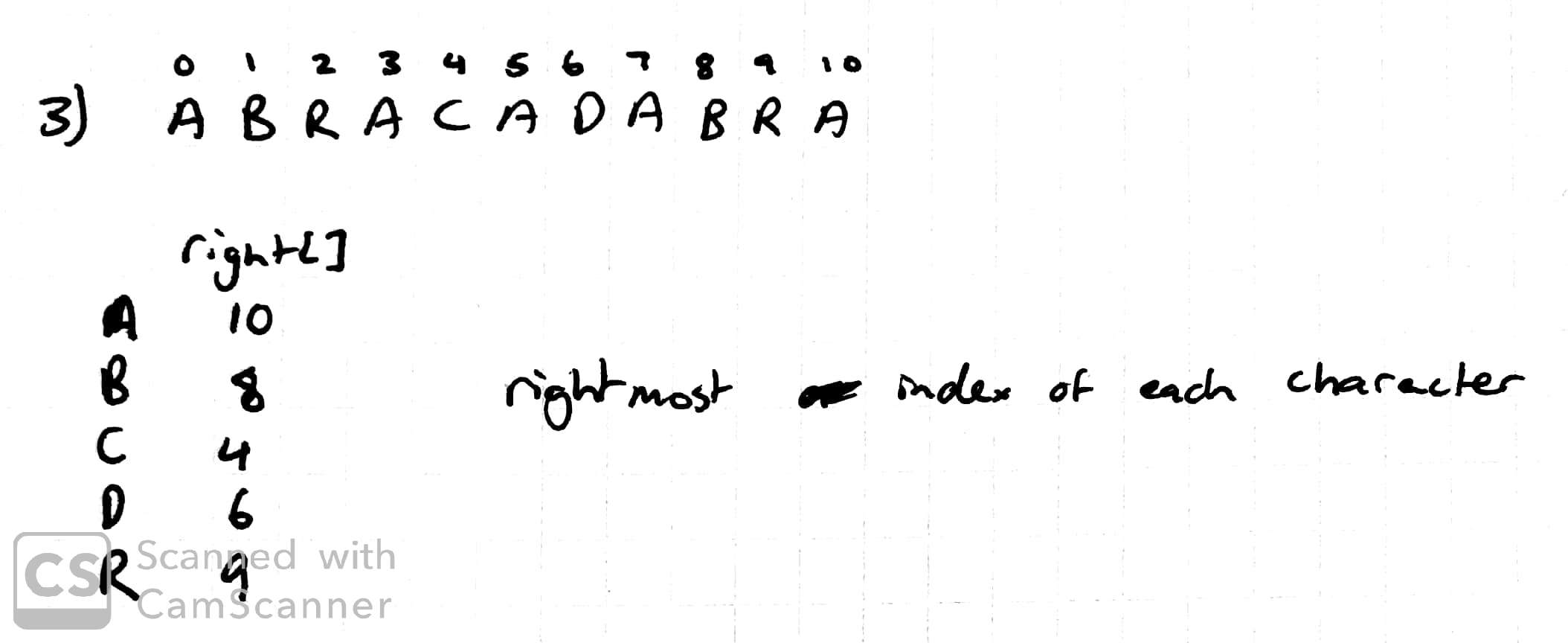
subsetChars.add(y.charAt(j))

i 🡨 i – 1

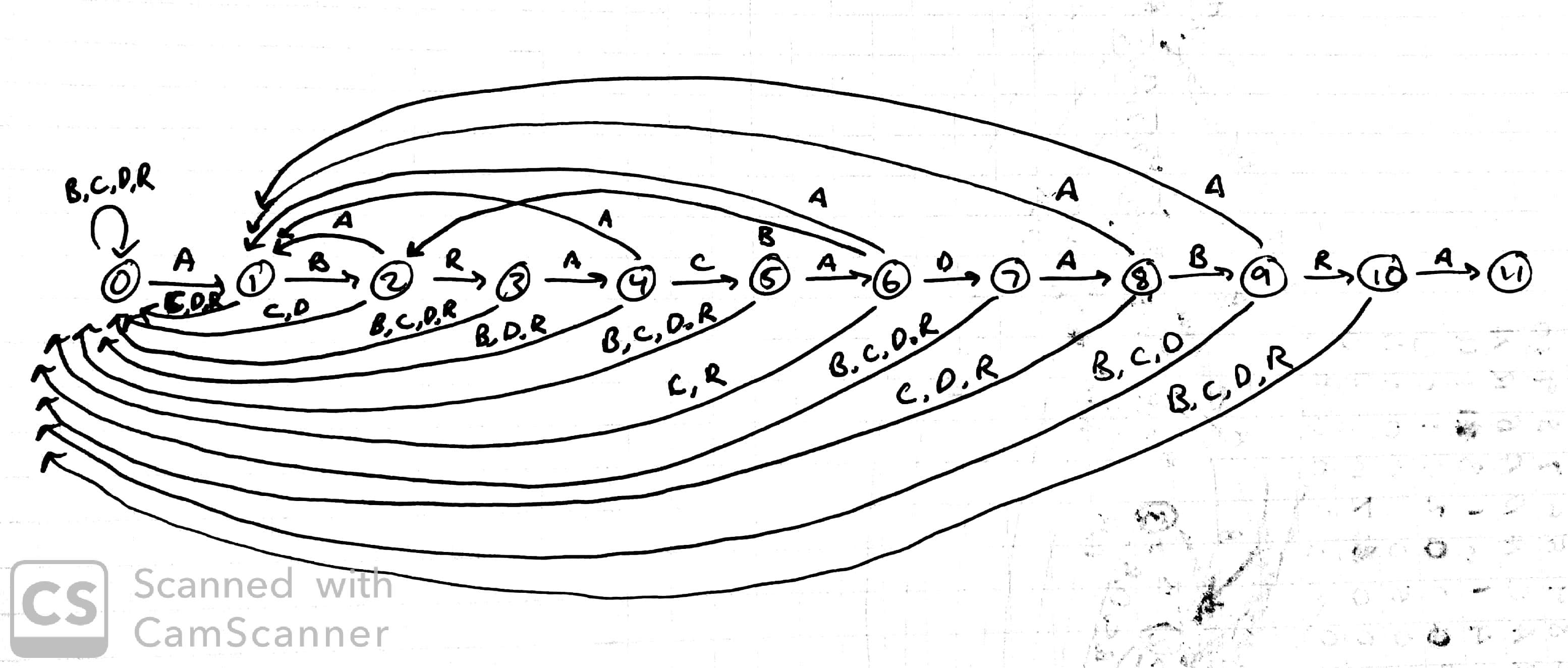
j 🡨 j – 1

setList.reverse()

print (setList)



|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| j | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| charAt(j) | | A | B | R | A | C | A | D | A | B | R | A |
| dfa[][j] | A | 1 | 1 | 1 | 4 | 1 | 6 | 1 | 8 | 1 | 1 | 11 |
| B | 0 | 2 | 0 | 0 | 0 | 0 | 2 | 0 | 9 | 0 | 0 |
| C | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 |
| D | 0 | 0 | 0 | 0 | 0 | 0 | 7 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 |



1. public int search(String txt, int k) {

int N = txt.length();

String txt0 = “”;

String txt1 = “”;

int returnVal = -1;

//Separate string into two (txt0 and txt1), ignoring the kth char

for (int i = 0; i < k; i++) {

txt0 = txt0 + txt[i];

}

for (int i = k+1; i < N; i++) {

txt1 = txt1 + txt[i];

}

//Hash each substring, ignoring the kth char

int txtHash0 = hash(txt0, k-1);

int txtHash1 = hash(txt1, N-k);

//assuming patHash0 and patHash1 correspond to the pattern’s chars, also ignoring the kth char from the first char read in each iteration

if (patHash0 == txtHash0 && patHash1 == txtHash1)

return 0;

//run the rabin-karp algorithm. If txtHash0 == patHash0, check if txtHash1 == txtHash1

//if both match, return the first char index of the matched substring

for (int i = M; i < k-1; i++) {

txtHash0 = (txtHash0 + Q - RM\*txt0.charAt(i-M) % Q) % Q;

txtHash0 = (txtHash0\*R + txt0.charAt(i)) % Q;

if (patHash0 == txtHash0) {

returnVal = i – M + 1;

for (int j = M; j < N; j++) {

txtHash1 = (txtHash1 + Q - RM\*txt1.charAt(j-M) % Q) % Q;

txtHash1 = (txtHash1\*R + txt1.charAt(j)) % Q;

if (patHash1 == txtHash1 && returnVal != -1)

return returnVal;

}

}

}

return N;

}