CSCI2100 Data Structures
Text Processing

Irwin King

king@cse.cuhk.edu.hk
http://www.cse.cuhk.edu.hk/~king

Department of Computer Science & Engineering
The Chinese University of Hong Kong

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Outline

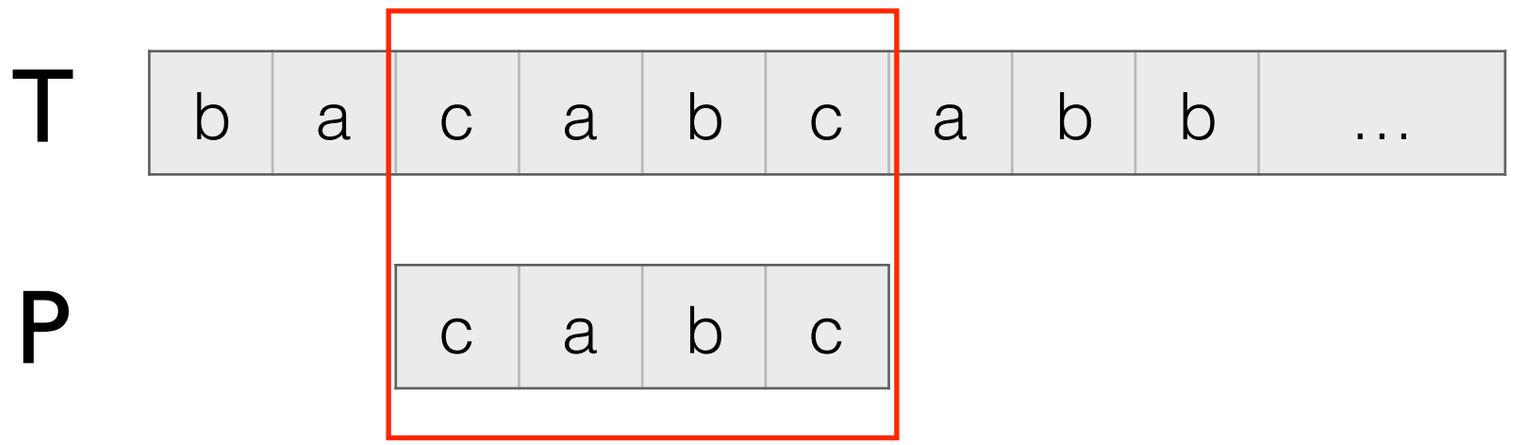
- String Matching with Finite Automata
- Basic Library Functions of String

Resources: http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15251-s08/Site/Materials/Lectures/Lecture05/lecture05.pdf
http://en.cppreference.com/w/c/string/byte
http://www.tutorialspoint.com/c_standard_library/index.htm

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

String Matching

- Given a text array T[1, …, n] and a pattern array P[1, …, m].
- The String Matching Problem is to find all the occurrence of P in T.



CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Automata

- A finte automaton is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$
- Q is the set of states
- Σ is the alphabet
- $\delta : Q \times \Sigma \rightarrow Q$ is the transition function
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ the set of accept states

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Construct an Automata

- For the pattern “ababc”
- Alphabet: {a, b, c}
- States: {“a”, “ab”, “aba”, “abab”, “ababc”, q0}
- Start state: q0
- Accept States: “ababc”

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Construct an Automaton

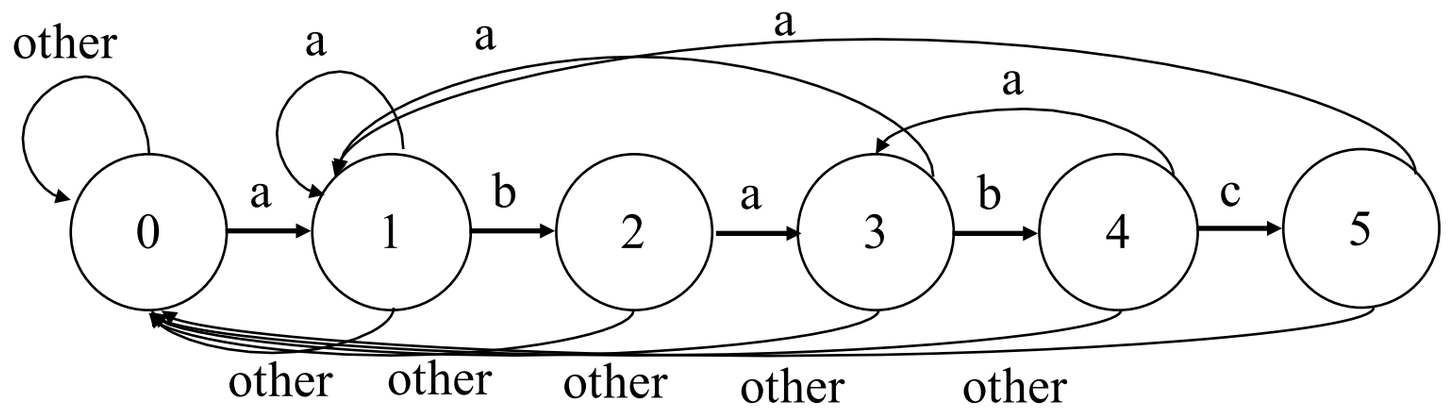
- For the pattern "ababc"
- Transition function:

δ	a	b	c
q0	a	q0	q0
a	a	ab	q0
ab	aba	q0	q0
aba	a	abab	q0
abab	aba	q0	ababc
ababc	a	q0	q0

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Construct an Automaton

- For the pattern "ababc":
- We could construct the automaton according to the transition function table.

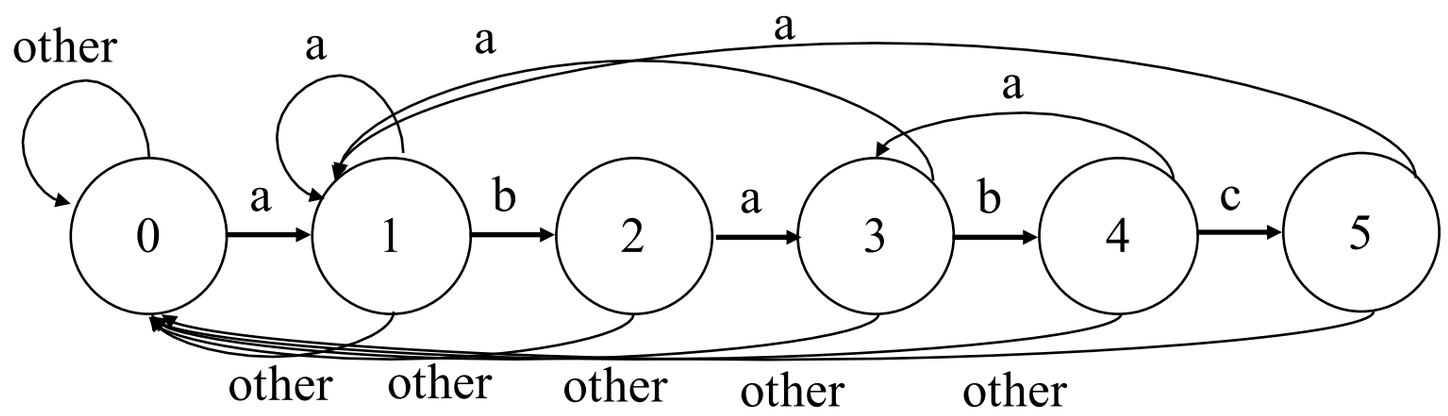


- For simplicity, "0" = q0, "1" = "a", "2" = "ab", "3" = "aba", "4" = "abab", "5" = "ababc".
- Then we use this automaton to find all the occurrences of "ababc" in string "ababacababcba".

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

String Matching with Automata

Pattern: ababc

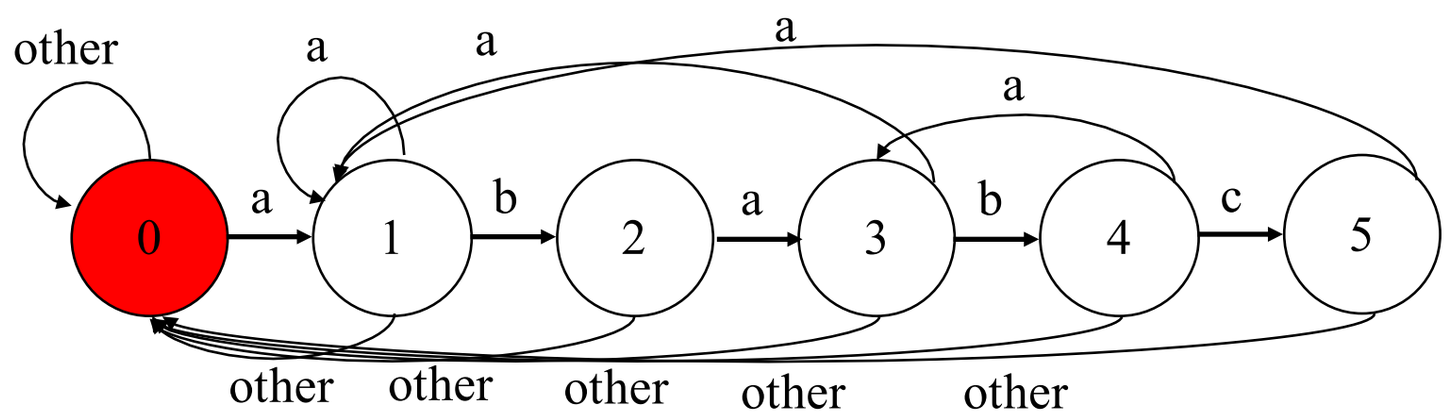


i	-	1	2	3	4	5	6	7	8	9	10	11	12	13
T[i]	-	a	b	a	b	a	c	a	b	a	b	c	b	a
State	0													

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

String Matching with Automata

Pattern: ababc

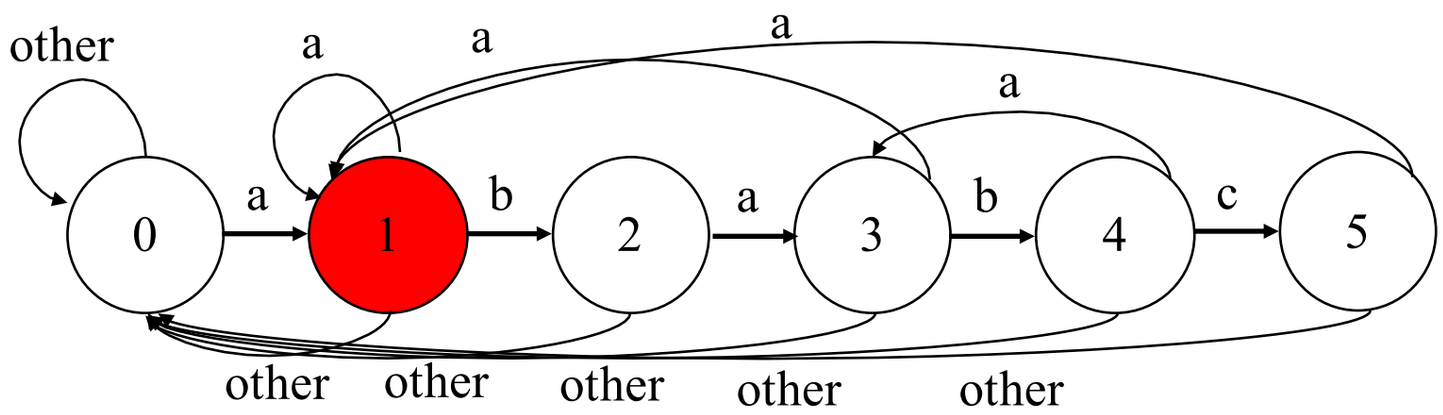


i	-	1	2	3	4	5	6	7	8	9	10	11	12	13
T[i]	-	a	b	a	b	a	c	a	b	a	b	c	b	a
State	0													

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

String Matching with Automata

Pattern: ababc

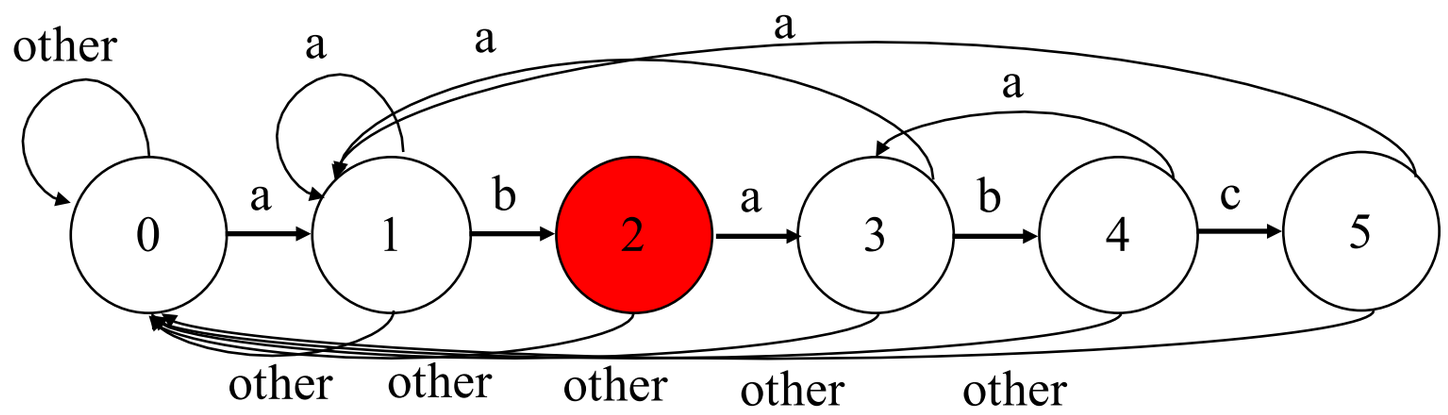


i	-	1	2	3	4	5	6	7	8	9	10	11	12	13
T[i]	-	a	b	a	b	a	c	a	b	a	b	c	b	a
State	0	1												

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

String Matching with Automata

Pattern: ababc

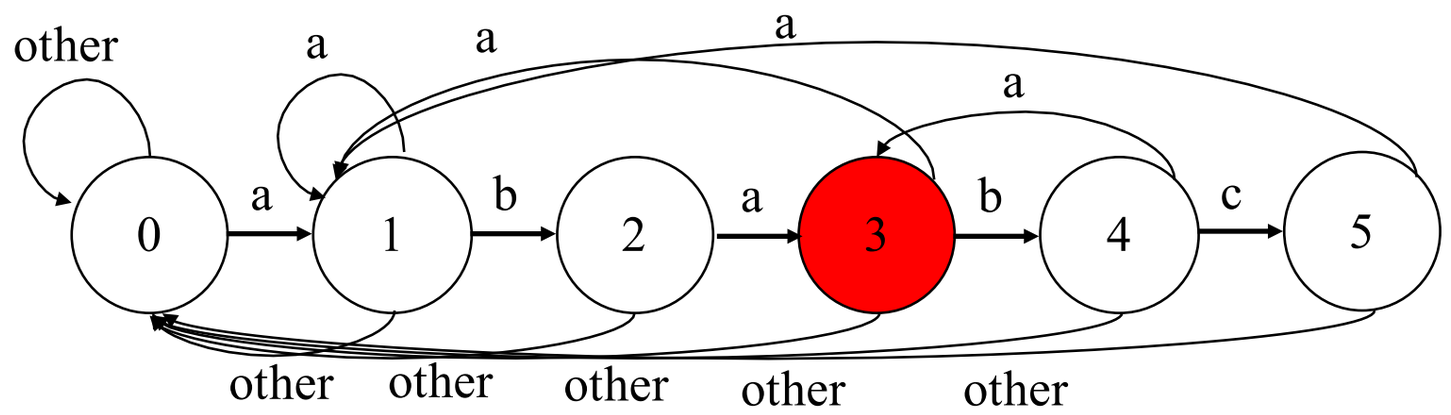


i	-	1	2	3	4	5	6	7	8	9	10	11	12	13
T[i]	-	a	b	a	b	a	c	a	b	a	b	c	b	a
State	0	1	2											

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

String Matching with Automata

Pattern: ababc

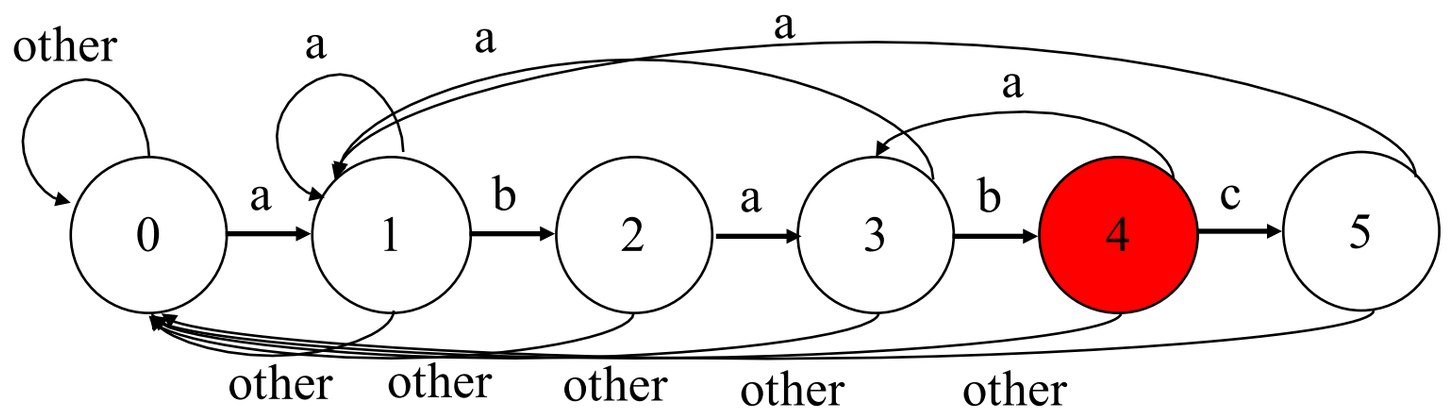


i	-	1	2	3	4	5	6	7	8	9	10	11	12	13
T[i]	-	a	b	a	b	a	c	a	b	a	b	c	b	a
State	0	1	2	3										

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

String Matching with Automata

Pattern: ababc

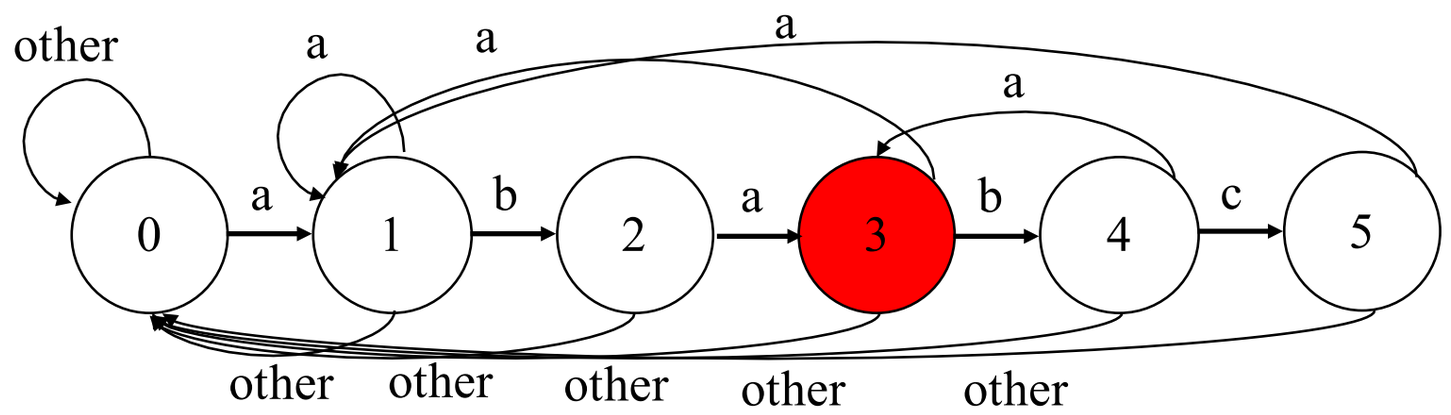


i	-	1	2	3	4	5	6	7	8	9	10	11	12	13
T[i]	-	a	b	a	b	a	c	a	b	a	b	c	b	a
State	0	1	2	3	4									

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

String Matching with Automata

Pattern: ababc

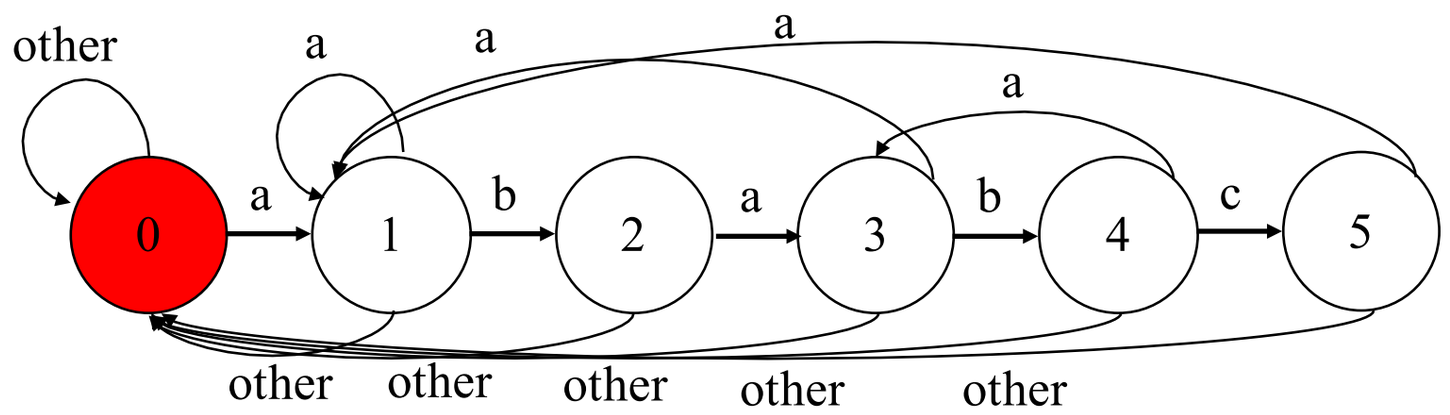


i	-	1	2	3	4	5	6	7	8	9	10	11	12	13
T[i]	-	a	b	a	b	a	c	a	b	a	b	c	b	a
State	0	1	2	3	4	3								

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

String Matching with Automata

Pattern: ababc

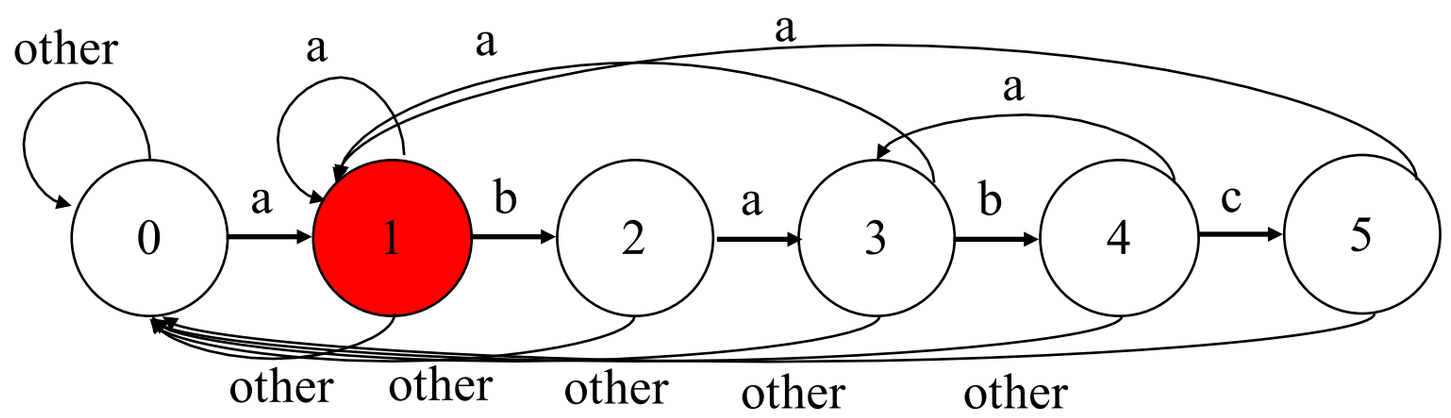


i	-	1	2	3	4	5	6	7	8	9	10	11	12	13
T[i]	-	a	b	a	b	a	c	a	b	a	b	c	b	a
State	0	1	2	3	4	3	0							

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

String Matching with Automata

Pattern: ababc

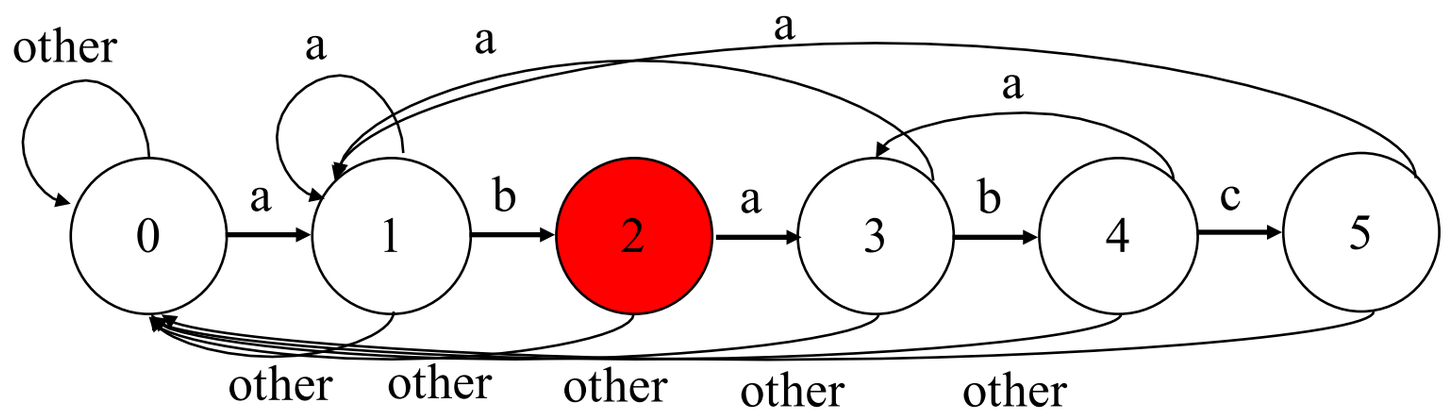


i	-	1	2	3	4	5	6	7	8	9	10	11	12	13
T[i]	-	a	b	a	b	a	c	a	b	a	b	c	b	a
State	0	1	2	3	4	3	0	1						

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

String Matching with Automata

Pattern: ababc

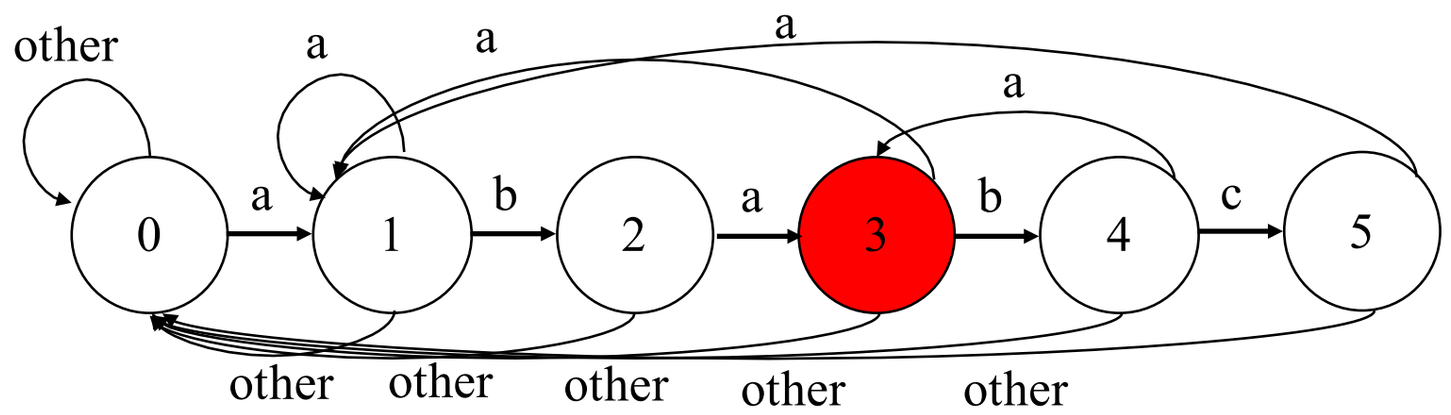


i	-	1	2	3	4	5	6	7	8	9	10	11	12	13
T[i]	-	a	b	a	b	a	c	a	b	a	b	c	b	a
State	0	1	2	3	4	3	0	1	2					

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

String Matching with Automata

Pattern: ababc

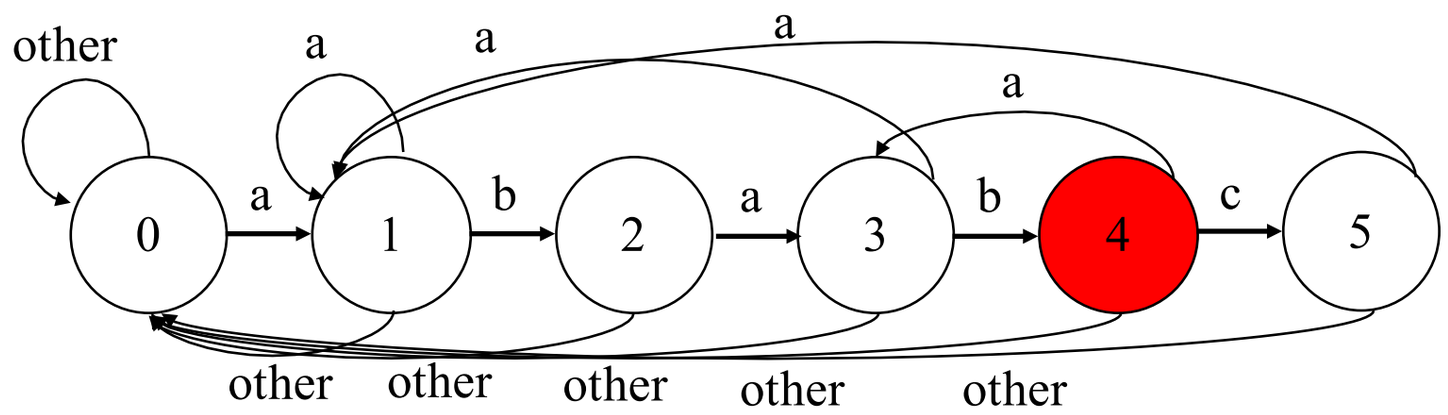


i	-	1	2	3	4	5	6	7	8	9	10	11	12	13
T[i]	-	a	b	a	b	a	c	a	b	a	b	c	b	a
State	0	1	2	3	4	3	0	1	2	3				

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

String Matching with Automata

Pattern: ababc

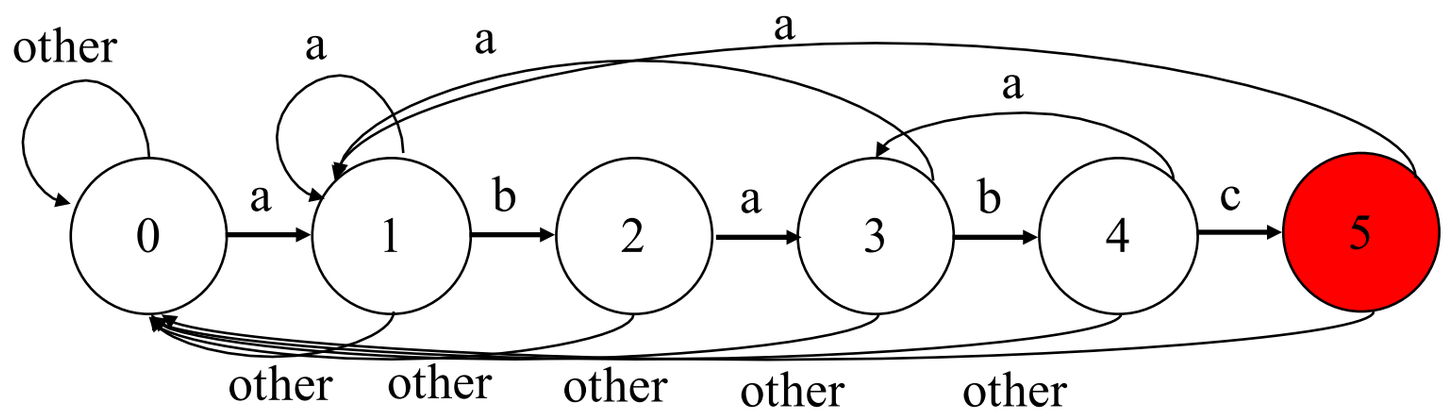


i	-	1	2	3	4	5	6	7	8	9	10	11	12	13
T[i]	-	a	b	a	b	a	c	a	b	a	b	c	b	a
State	0	1	2	3	4	3	0	1	2	3	4			

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

String Matching with Automata

Pattern: ababc



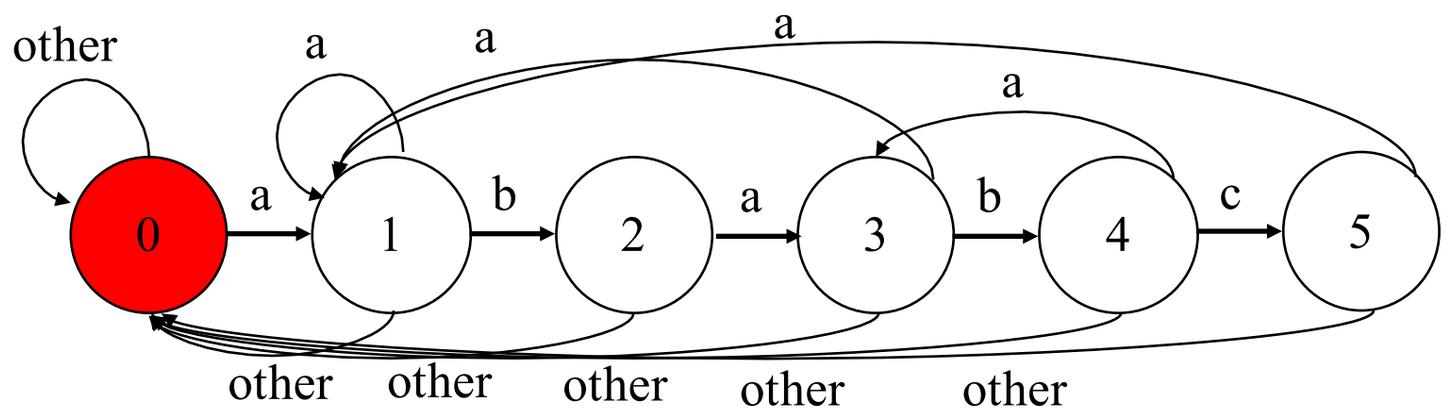
i	-	1	2	3	4	5	6	7	8	9	10	11	12	13
T[i]	-	a	b	a	b	a	c	a	b	a	b	c	b	a
State	0	1	2	3	4	3	0	1	2	3	4	5		

Find the pattern!

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

String Matching with Automata

Pattern: ababc

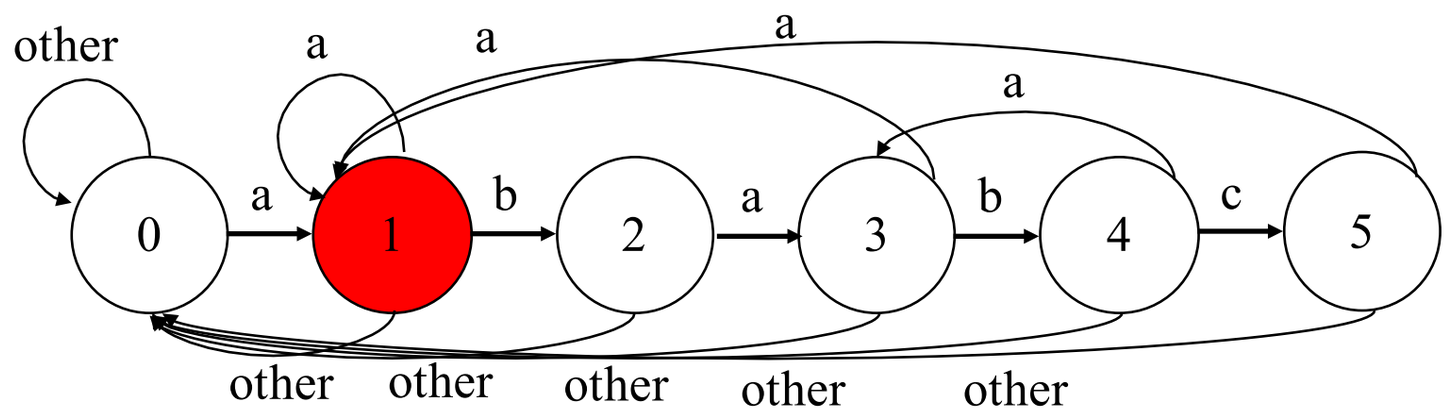


i	-	1	2	3	4	5	6	7	8	9	10	11	12	13
T[i]	-	a	b	a	b	a	c	a	b	a	b	c	b	a
State	0	1	2	3	4	3	0	1	2	3	4	5	0	

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

String Matching with Automata

Pattern: ababc



i	-	1	2	3	4	5	6	7	8	9	10	11	12	13
T[i]	-	a	b	a	b	a	c	a	b	a	b	c	b	a
State	0	1	2	3	4	3	0	1	2	3	4	5	0	1

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Basic Library Functions of String

Declarations:

size_t strlen(const char *str);

void *memcpy(void *dest, const void *src, size_t n);

char *strncat(char *dest, const char *src, size_t n);

int memcmp (const void *str1, const void *str2, size_t n);

void *memchr(const void *str, int c, size_t n);

char *strpbrk(const char *str1, const char *str2);

char *strstr(const char *haystack, const char *needle)

int atoi (const char *str);

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Basic Library Functions of String

Declaration:

size_t strlen(const char *str);

Description: This function computes the length of the string **str** up to, but not including the terminating null character.

Parameters:

- **str:** This is the string whose length is to be found.

Return: This function returns the length of string.

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Basic Library Functions of String

Declaration:

size_t strlen(const char *str);

Example:

```
#include <stdio.h>
#include <string.h>

int main () {

   char *str = "Data Structure!";
   int len = 0;

   len = (int)strlen(str);
   printf("Length of \"%s\" is %d.\n", str, len);

   return(0);
}
```

Output → Length of "Data Structure!" is 15.

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Basic Library Functions of String

Declaration:

void *memcpy(void *dest, const void *src, size_t n);

Description: This function copies **n** characters from memory area **src** to memory area **dest**.

Parameters:

- **dest:** This is pointer to the destination array where the content is to be copied, type-casted to a pointer of type void*.
- **src:** This is pointer to the source of data to be copied, type-casted to a pointer of type void*.
- **n:** This is the number of bytes to be copied.

Return: This function returns a pointer to destination, which is **dest**.

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Basic Library Functions of String

Declaration:

void *memcpy(void *dest, const void *src, size_t n);

Example:

```
#include <stdio.h>
#include <string.h>

int main () {

   const char src[50] = "Data Structure";
   char dest[50] = "\0";

   printf("Before memcpy dest = %s\n", dest);
   memcpy(dest, src, strlen(src)+1);
   printf("After memcpy dest = %s\n", dest);

   return(0);
}
```

Output

```
Before memcpy dest =
After memcpy dest = Data Structure
```

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Basic Library Functions of String

Declaration:

char *strncat(char *dest, const char *src, size_t n);

Description: This function appends the string pointed to by **src** to the end of the string pointed to by **dest** up to **n** characters long.

Parameters:

- **dest:** This is pointer to the destination array, which should contain a C string, and should be large enough to contain the concatenated resulting string which includes the additional null-character.
- **src:** This is the string to be appended.
- **n:** This is the maximum number of characters to be appended.

Return: This function returns a pointer to the resulting string dest.

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Basic Library Functions of String

Declaration:

char *strncat(char *dest, const char *src, size_t n);

Example:

```
#include <stdio.h>
#include <string.h>

int main () {

   char src[50] = "Structure!";
   char dest[50] = "Data ";

   strncat(dest, src, strlen(src));

   printf("Final destination string : \"%s\".\n", dest);

   return(0);
}
```

Output

Final destination string : "Data Structure!".

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Basic Library Functions of String

Declaration:

int memcmp (const void *str1, const void *str2, size_t n);

Description: This function compares the first **n** bytes of memory area **str1** and memory area **str2**.

Parameters:

- **str1:** This is the pointer to a block of memory.
- **str2:** This is the pointer to a block of memory.
- **n:** This is the number of bytes to be compared.

Return:

If return value < 0 then it indicates str1 is less than str2.
If return value > 0 then it indicates str2 is less than str1.
If return value = 0 then it indicates str1 is equal to str2.

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Basic Library Functions of String

Declaration:

int memcmp (const void *str1, const void *str2, size_t n);

Example:

```
#include <stdio.h>
#include <string.h>

int main () {
   char *str1 = “abcdef”;
   char *str2 = “ABCDEF”;
   int ret;
   ret = memcmp(str1, str2, 6);
   if(ret > 0) {
      printf("str2 is less than str1.”);
   } else if(ret < 0) {
      printf("str1 is less than str2.”);
   } else {
      printf("str1 is equal to str2.”);
   }
   return(0);
}
```

Output →

```
str2 is less than str1.
```

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Basic Library Functions of String

Declaration:

void *memchr(const void *str, int c, size_t n);

Description: This function searches for the first occurrence of the character **c** (an unsigned char) in the first **n** bytes of the string pointed to, by the argument **str**.

Parameters:

- **str:** This is the pointer to the block of memory where the search is performed.
- **c:** This is the value to be passed as an int, but the function performs a byte per byte search using the unsigned char conversion of this value.
- **n:** This is the number of bytes to be analyzed.

Return: This function returns a pointer to the matching byte or NULL if the character does not occur in the given memory area.

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Basic Library Functions of String

Declaration:

void *memchr(const void *str, int c, size_t n);

Example:

```
#include <stdio.h>
#include <string.h>

int main () {

   const char str[] = "//Data Structure + Algorithm!";
   const char ch = '+';
   char *ret;

   ret = memchr(str, ch, strlen(str));

   printf("String after %c is %s\n", ch, ret);

   return(0);
}
```

Output → String after + is + Algorithm!

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Basic Library Functions of String

Declaration:

char *strpbrk(const char *str1, const char *str2);

Description: This function finds the first character in the string **str1** that matches any character specified in **str2**. This does not include the terminating null-characters.

Parameters:

- **str1:** This is the C string to be scanned.
- **str2:** This is the C string containing the characters to match.

Return: This function returns a pointer to the character in str1 that matches one of the characters in str2, or NULL if no such character is found.

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Basic Library Functions of String

Declaration:

char *strpbrk(const char *str1, const char *str2);

Example:

```
#include <stdio.h>
#include <string.h>

int main () {

   const char *str1 = "Data Structure 2100A";
   const char *str2 = "5B2";
   char *ret;

   ret = strpbrk(str1, str2);
   if(ret) {
      printf("First matching character: %c\n", *ret);
   } else {
      printf("Character not found");
   }
   return(0);
}
```

Output → First matching character: 2

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Basic Library Functions of String

Declaration:

char *strstr(const char *haystack, const char *needle)

Description: This function finds the first occurrence of the substring **needle** in the string **haystack**. The terminating '\0' characters are not compared.

Parameters:

- **haystack:** This is the main C string to be scanned.
- **needle:** This is the small string to be searched within haystack string.

Return: This function returns a pointer to the first occurrence in haystack of any of the entire sequence of characters specified in needle, or a null pointer if the sequence is not present in haystack.

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Basic Library Functions of String

Declaration:

char *strstr(const char *haystack, const char *needle)

Example:

```
#include <stdio.h>
#include <string.h>

int main () {

   const char haystack[20] = "Data Structure2100A";
   const char needle[10] = "Structure";
   char *ret;

   ret = strstr(haystack, needle);

   printf("The substring is: %s.\n", ret);
   }
   return(0);
}
```

Output → The substring is: Structure2100A.

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Basic Library Functions of String

Declaration:

int atoi (const char *str);

Description: This function converts the string argument **str** to an integer (type int).

Parameters:

- **str:** This is the string representation of an integral number.

Return: This function returns the converted integral number as an int value. If no valid conversion could be performed, it returns zero.

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Basic Library Functions of String

Declaration:

int atoi (const char *str);

Example:

```
#include <stdio.h>
#include <stdlib.h>

int main () {

   int val;
   char *str = “666999”;

   val = atoi(str);
   printf("String value = %s, Int value = %d\n", str, val);

   return(0);
}
```

Output → String value = 666999, Int value = 666999

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.

Thank You!

CSCI2100 Data Structures, The Chinese University of Hong Kong, Irwin King, All rights reserved.