Linear and Binary Search

Dr. Animesh Chaturvedi

Assistant Professor: IIIT Dharwad

Young Researcher: Heidelberg Laureate Forum

Postdoc: King's College London & The Alan Turing Institute
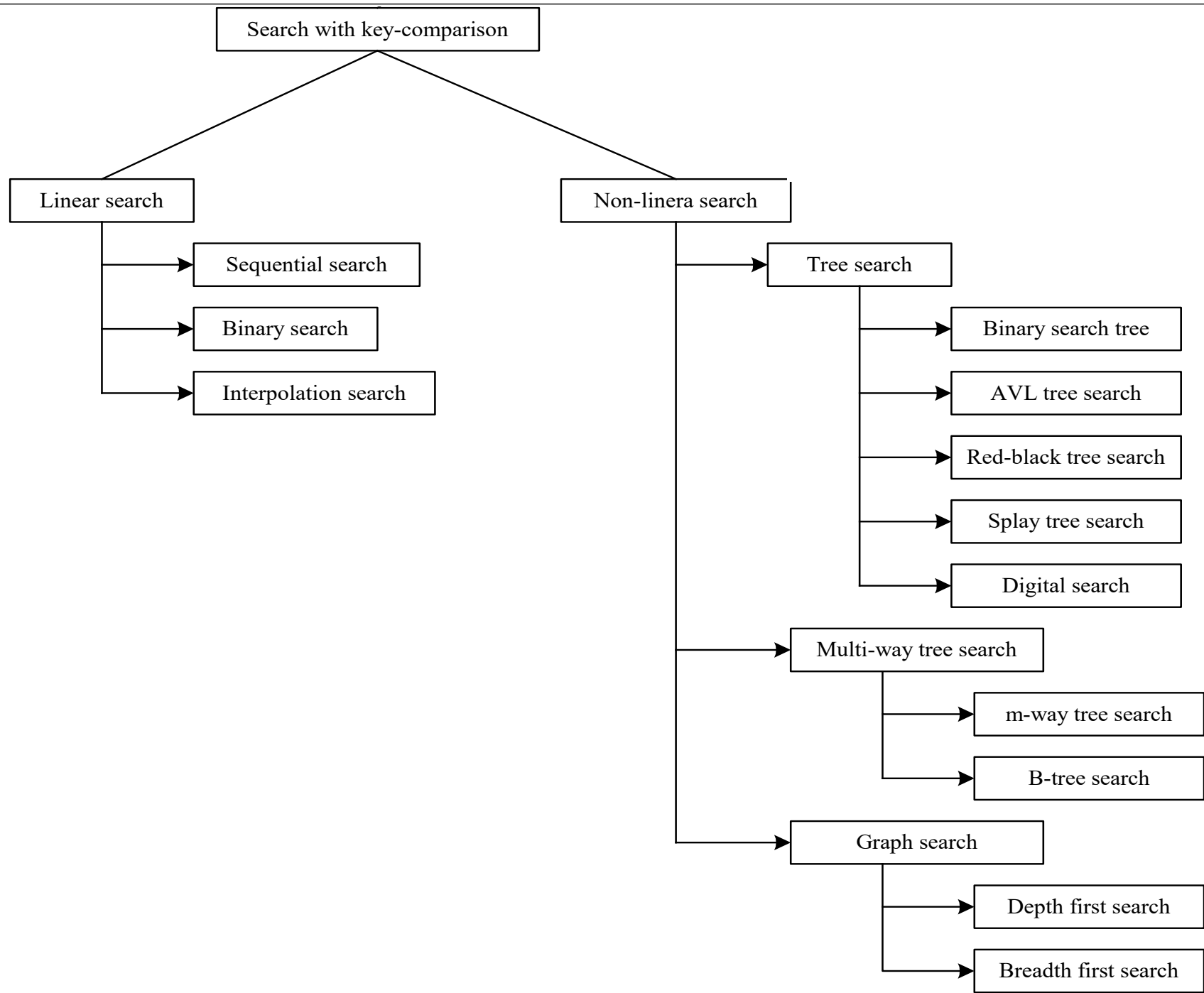
PhD: IIT Indore MTech: IIITDM Jabalpur

```
Search with key-comparison
├── Linear search
│   ├── Sequential search
│   ├── Binary search
│   └── Interpolation search
└── Non-linera search
    ├── Tree search
    │   ├── Binary search tree
    │   ├── AVL tree search
    │   ├── Red-black tree search
    │   ├── Splay tree search
    │   └── Digital search
    ├── Multi-way tree search
    │   ├── m-way tree search
    │   └── B-tree search
    └── Graph search
        ├── Depth first search
        └── Breadth first search
```

Linear Search

Flowchart: Sequential Search with Array
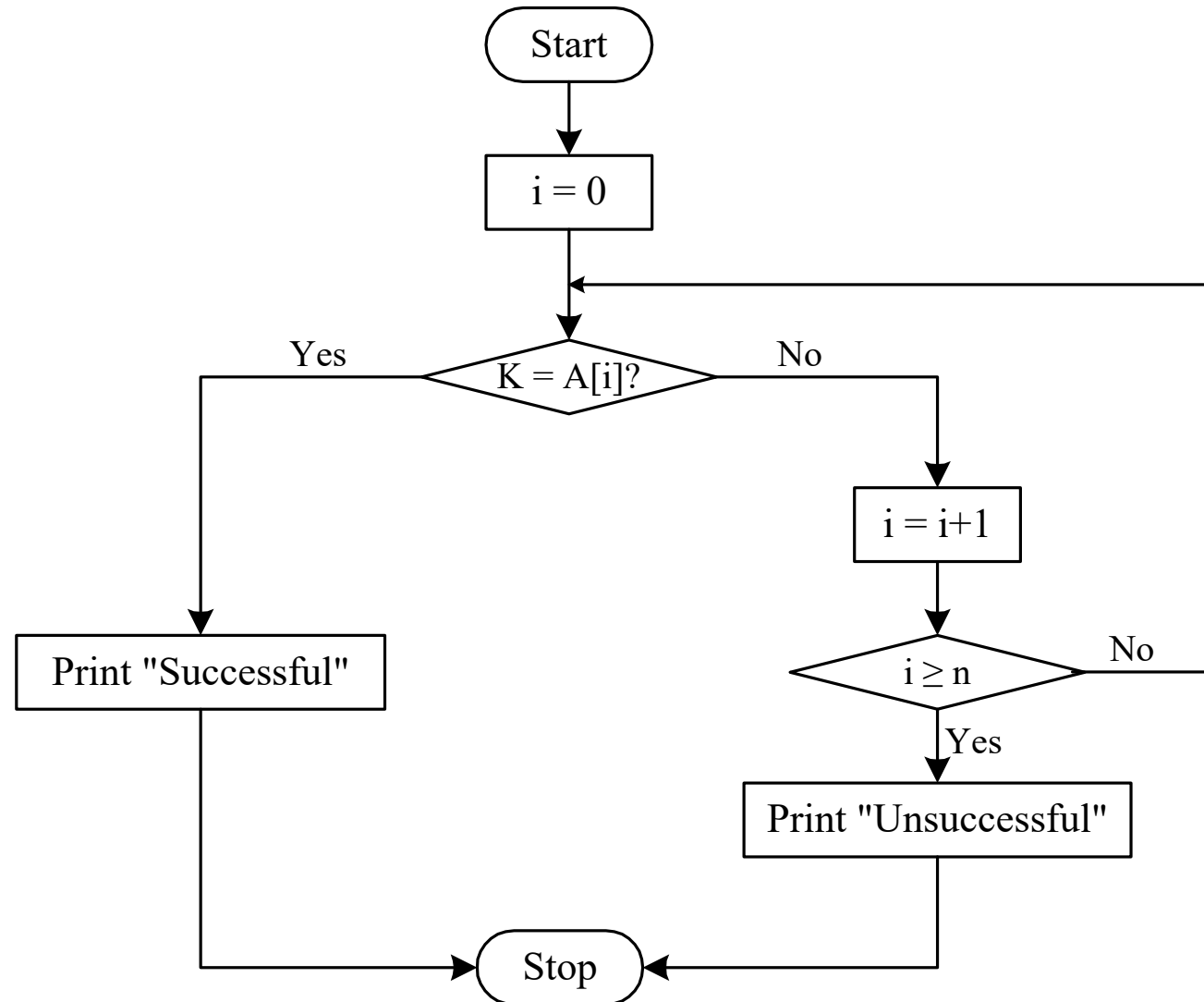
Example: Sequential Search with Array

```c
int main()
{
    int A[10], i, n, K, flag = 0;
    printf("Enter the size of an array: ");
    scanf("%d",&n);

    printf("Enter the elements of the array: ");
    for(i=0; i < n; i++)
        scanf("%d",&A[i]);
    printf("Enter the number to be searched: ");
    scanf("%d",&K);
    for(i=0;i<n;i++){
        if(a[i] == K){
            flag = 1; break;
        }
    }
    if(flag == 0)
        printf("The number is not in the list");
    else
        printf("The number is found at index %d",i);
    return 0;
}
```
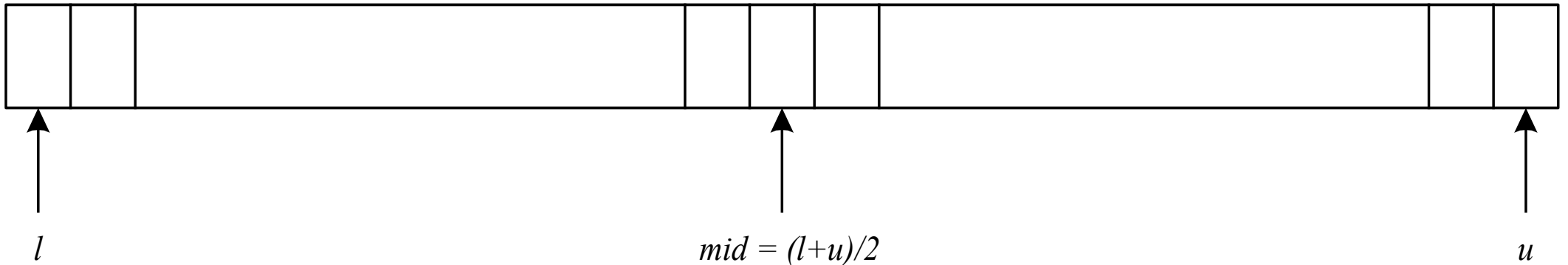
Complexity Analysis

- Case 1: The key matches with the first element
  - $T(n) = 1$

- Case 2: Key does not exist
  - $T(n) = n$

- Case 3: The key is present at any location in the array
  - $T(n) = (n+1)/2$

Complexity Analysis

Case	Number of key comparisons	Asymptotic complexity	Remark
Case 1	$T(n) = 1$	$T(n) = O(1)$	Best case
Case 2	$T(n) = n$	$T(n) = O(n)$	Worst case
Case 3	$T(n) = \dfrac{n+1}{2}$	$T(n) = O(n)$	Average case

Binary Search

Binary Search

(a) An ordered array of elemnets with index values l, u and mid

Binary Search

mid

l

u = mid-1

u

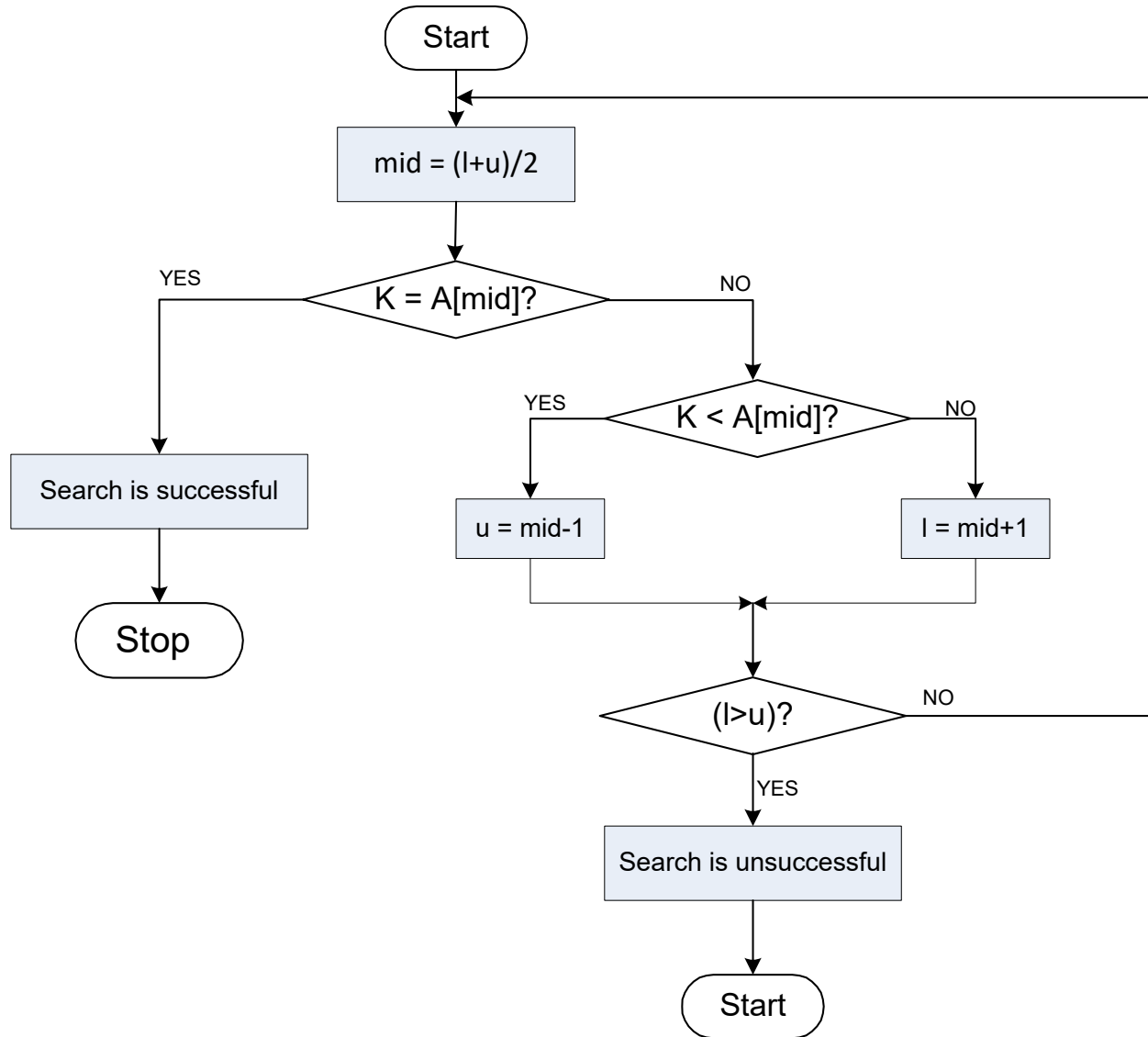Serach this half the same way
if *K < A[mid]*
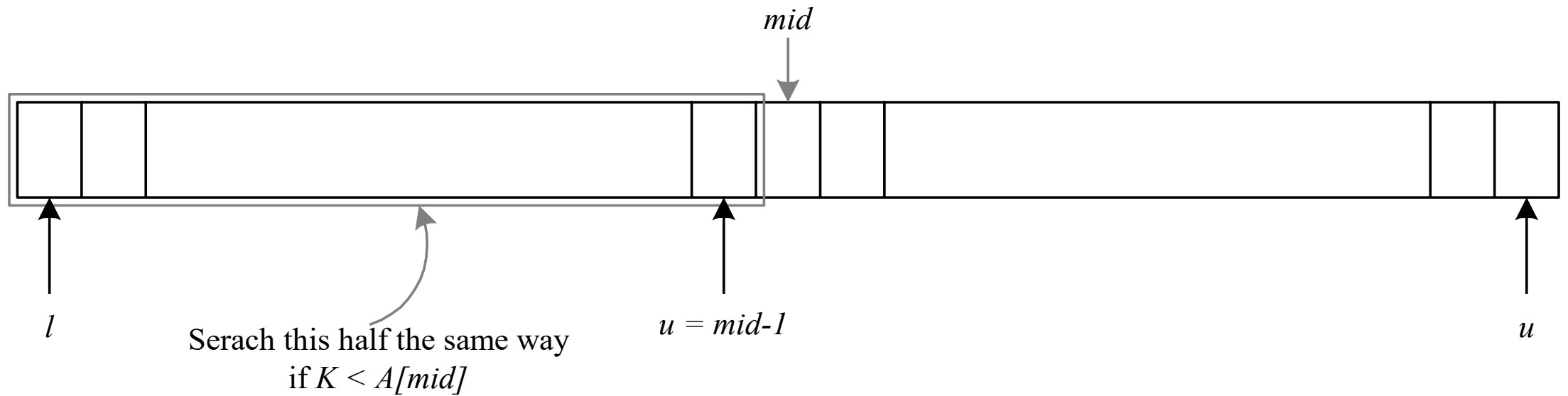
(b) Search the entire list turns into the searching of left-half only

Binary Search
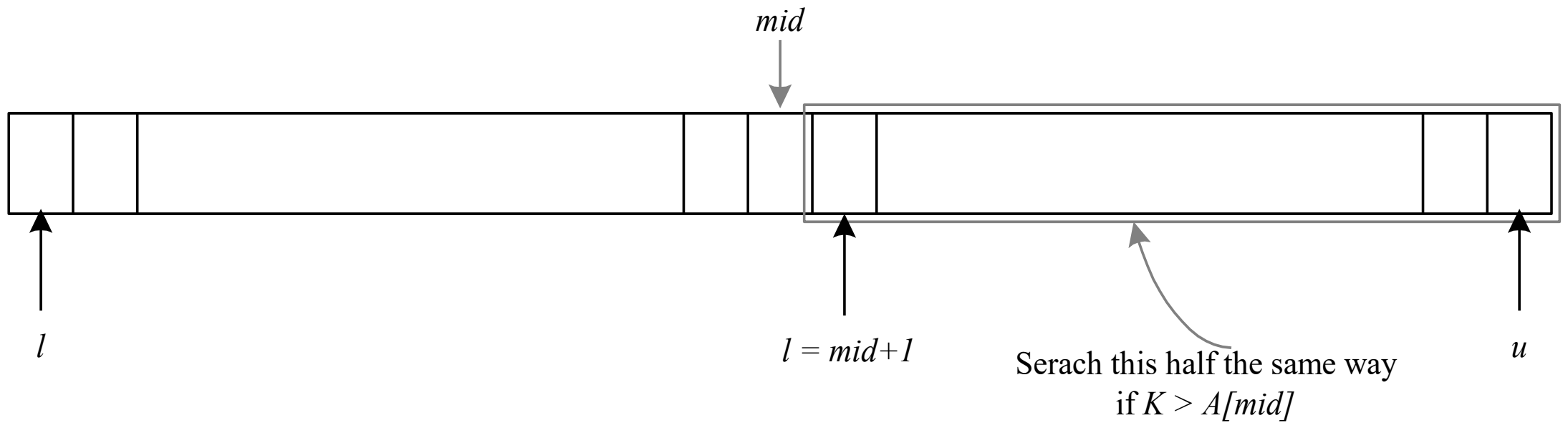
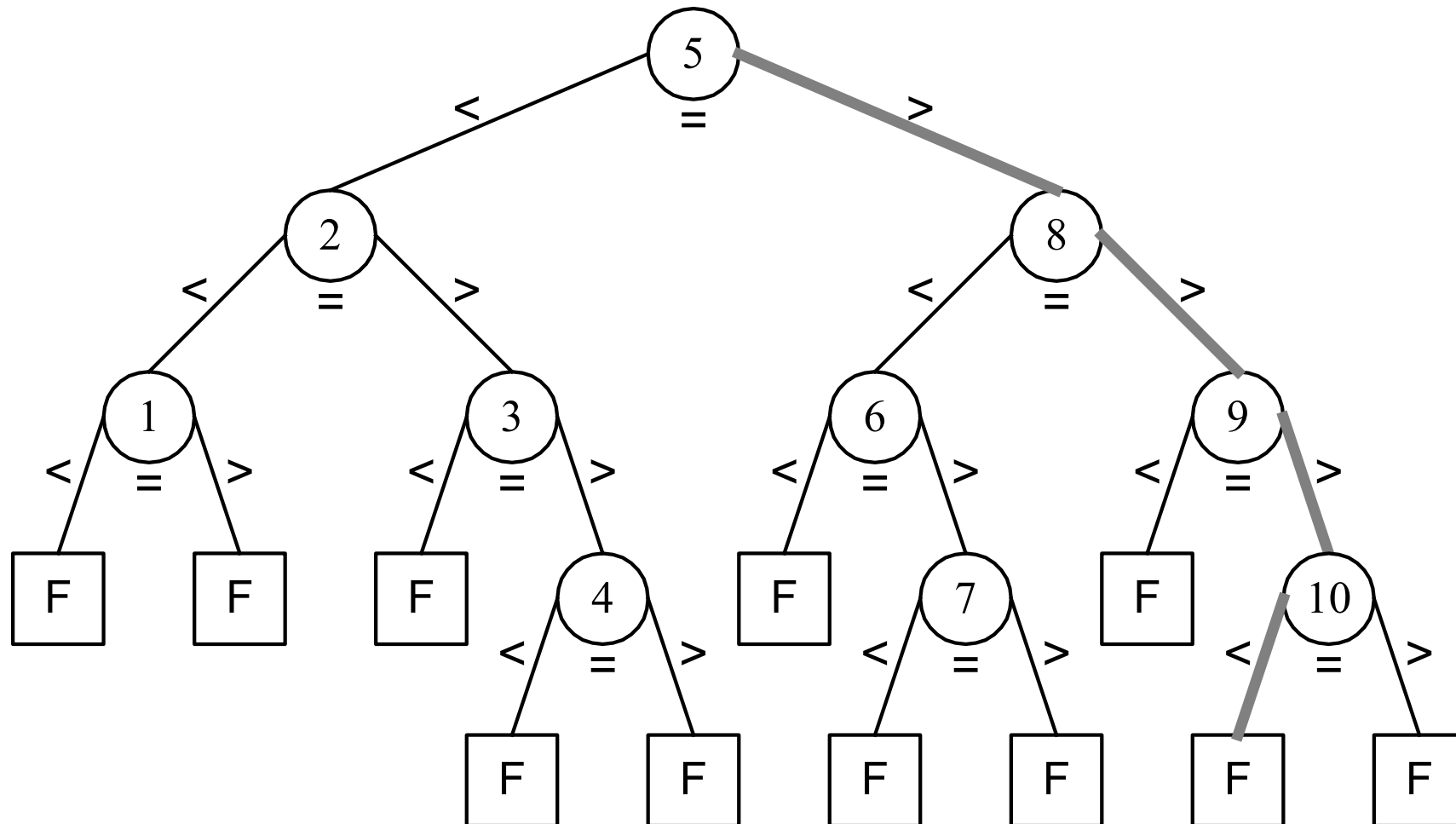
(c) Search the entire list turns into the searching of right-half only

Flowchart: Binary Search with Array

Binary Search

Binary Search (Iteration)

```c
#include <stdio.h>

int main()
{   int i, l, u, mid, n, K, data[100];

    printf("Enter number of elements\n");
    scanf("%d",&n);

    printf("Enter %d integers in sorted
                            order\n", n);

    for (i = 0; i < n; i++)
        scanf("%d",&array[i]);

    printf("Enter value to find\n");
    scanf("%d", &K);

    l = 0;
    u = n - 1;
    mid = (l+u)/2;

    while (l <= u) {
        if (data[mid] < K)
            l = mid + 1;
        else if (data[mid] == K) {
            printf("%d found at index %d.\n",
                            search, mid+1);
            break;
        }
        else
            u = mid - 1;
        mid = (l + u)/2;
    }
    if (l > u)
        printf("%d is not present in the list.\n",
                                        K);

    return 0;
}
```

Binary Search (Recursion)

```c
#include<stdio.h>
int main(){
    int data[100],i, n, K, flag, l, u;
    printf("Enter the size of an array: ");
    scanf("%d",&n);
    printf("Enter the elements of the
            array in sorted order: " );

    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    printf("Enter the number to be search: ");
    scanf("%d",&K);

    l=0,u=n-1;
    flag = binarySearch(data,n,K,l,u);
    if(flag==0)
        printf("Number is not found.");
    else
        printf("Number is found.");

    return 0; }
```

```c
int binary(int a[],int n,int K,int l,int u){

    int mid;

    if(l<=u){
        mid=(l+u)/2;
        if(K==a[mid]){
            return(1);
        }
        else if(K<a[mid]){
            return binarySearch(a,n,K,l,mid-1);
        }
        else
            return binarySearch(a,n,m,mid+1,u);
    }
    else return(0);

}
```

Sequential Search with Linked List

Sequential Search with Linked List

DATA	LINK

(a) Structure of a node in the linked list

H

Header

Search begins here

Search at an intermediate node:

Search stops here if key matches else move to its immediate next node

Search unsuccessfully ends here
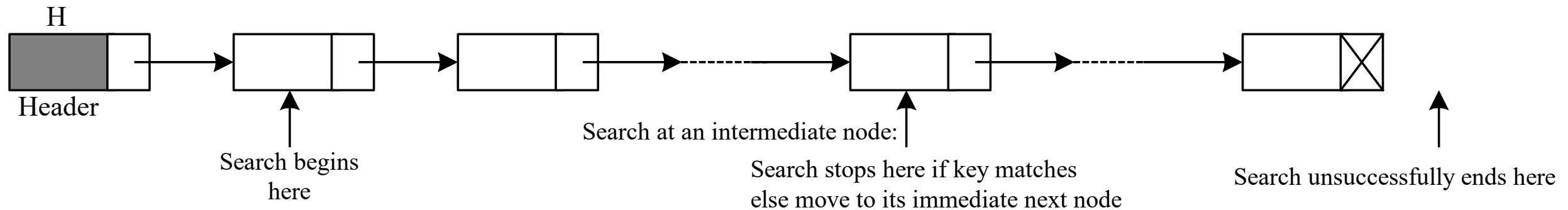
(b) Linear search on a linked list

Flow Chart: Sequential Search with LL
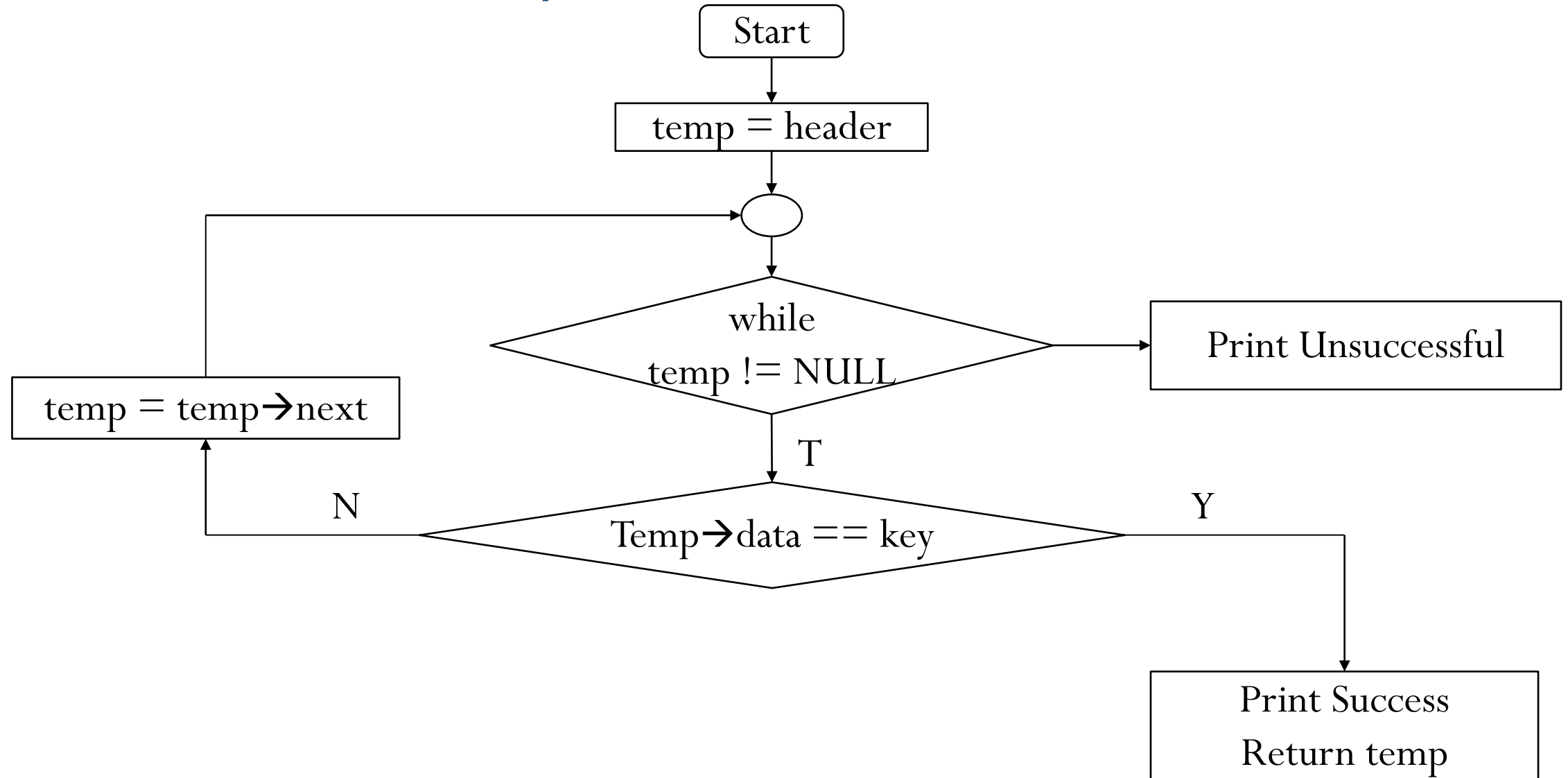
Sequential Search with Linked List

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

struct node
{   int data;
    struct node *next; };


int main()
{   struct node *header = NULL;
    int K, n;

    printf("Enter the number of nodes: ");
    scanf("%d", &n);
    printf("\nDisplaying the list\n");
    generate(header, num);
    printf("\nEnter key to search: ");
    scanf("%d", &key);
    searchBinary(header, K);
    delete(header);

    return 0;   }
```

Linear Search with Linked List

```c
void generate(struct node *head, int n)
{   int i;
    struct node *temp;

    for (i = 0; i < num; i++)
    {   temp = (struct node *)malloc(sizeof(struct node));
        temp->data = rand() % n;
        if (*header == NULL)
        {   *header = temp;
            temp->next = NULL;
        }
        else
        {   temp->next = header;
            header = temp;
        }
        printf("%d  ", temp->data);
    }
}
```

Linear Search with Linked List

```c
void searchBinary(struct node *temp, int K)
{   while (temp != NULL)
    {   if (temp->data == K)
        {   printf("key found\n");
            return;
        }
        else temp = temp->next;
    }
    printf("Key not found\n");
}


void delete(struct node *header)
{   struct node *temp;
    temp = header;
    while (temp != NULL)
    {   temp = temp->next;
        free(header);
        header = temp;
    }
}
```

Complexity Analysis

Case	Number of key comparisons	Asymptotic complexity	Remark
Case 1	$T(n) = 1$	$T(n) = O(1)$	Best case
Case 2	$T(n) = \dfrac{n+1}{2}$	$T(n) = O(n)$	Average case
Case 3	$T(n) = n$	$T(n) = O(n)$	Worst case

References

- Debasis Samanta, Computer Science & Engineering, Indian Institute of Technology Kharagpur, Spring-2017, Programming and Data Structures.
  https://cse.iitkgp.ac.in/~dsamanta/courses/pds/index.html

ขอบคุณ
Thai

Grazie
Italian

תודה רבה
Hebrew

ಧನ್ಯವಾದಗಳು
Kannada

धन्यवादः
Sanskrit

Ευχαριστώ
Greek

Thank You
English

Gracias
Spanish

Спасибо
Russian

Obrigado
Portuguese

شكراً
Arabic

https://sites.google.com/site/animeshchaturvedi07

Merci
French

多謝
Traditional Chinese

धन्यवाद
Hindi

Danke
German

多谢
Simplified Chinese

நன்றி
Tamil
Tamil

ありがとうございました
Japanese

감사합니다
Korean