Computing Laboratory

Sorting Techniques

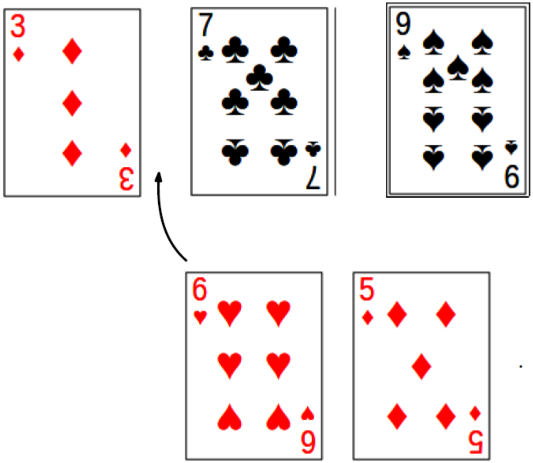
Malay Bhattacharyya

Assistant Professor

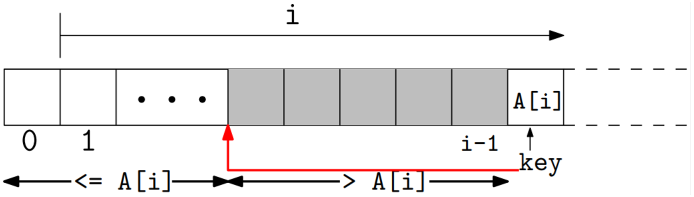
Machine Intelligence Unit
Indian Statistical Institute, Kolkata

December, 2021

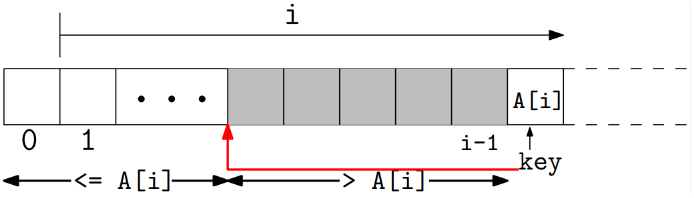
Insertion sort



Insertion sort



Insertion sort

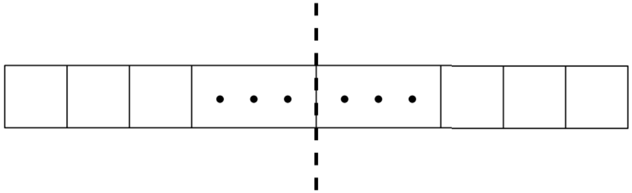


```
for(i = 1; i < n; i++){
    key = A[i];
    /* Find the right place for A[i] in A[0 ... i-1] */
    for(j = i-1; j >= 0 && A[j] > key; j--)
        A[j+1] = A[j];
    A[j+1] = key;
}
```

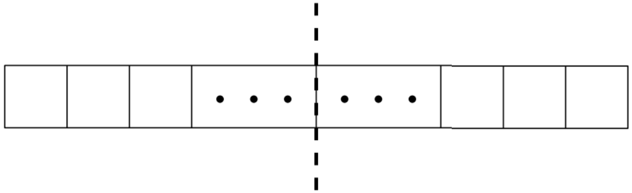
Bubble sort

```
for(i = 0; i < n-1; i++)
    for(j = 0; j < n-i-1; j++)
        if(A[j+1] < A[j])
            swap(A, j, j+1);
```

Merge sort



Merge sort



```
void msort(int *A, int beginning, int end){
     if(beginning < end){
        middle = (beginning + end) / 2;
        msort(A, beginning, middle);
        msort(A, middle+1, end);
        merge(A, beginning, middle, end);
     }
}
```

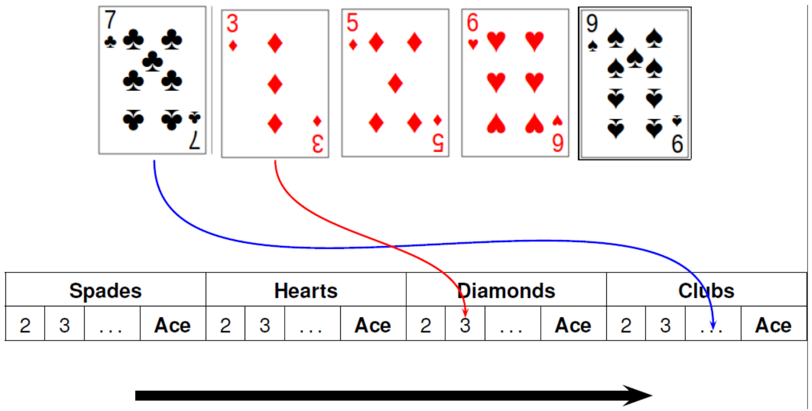
Merge sort

```
void merge(A, b, m, e){
     int i, j, k;
     /* allocate space for auxiliary array B */
     for(i = b, j = m+1, k = 0; i <= m && j <= e; )
         if(A[i] < A[j])
            B[k++] = A[i++];
         else if(A[i] > A[j])
            B[k++] = A[j++];
         else
            B[k++] = A[i++], B[k++] = A[j++];
     while(i <= m) B[k++] = A[i++];
     while(j <= e) B[k++] = A[j++];
     assert(...);
}
```

Heap sort

```
void heapsort(void *a, int N, size_t element_size,
            int (*comparator)(void *, int, int)){
    int k;
    HEAP h;
    h.element_size = element_size;
    h.num_allocated = h.num_used = N;
    h.array = a;
    h.comparator = comparator;
    /* Make heap out of array */
    for(k = N/2; k >= 1; k--)
        swapDown(&h, k);
    /* Sort by successive deleteMax */
    while(h.num_used > 1){
        swap(&h, 1, h.num_used); // move max to end
        h.num_used--;
        swapDown(&h, 1);
    }
}
```

Bucket sort



Let us understand: function pointers

- Declaring function pointers

  `<return type> (* <function name>) ( <parameter list> )`

  The brackets around the function name are important!!!
  Example:

  `int *aFunction(int), *(*aFunctionPointer)(int);`

- Using function pointers

  `(*f)(...)`

- Setting function pointer variables / passing function pointers asarguments: simply use the name of the function
  Example:

  `aFunctionPointer = aFunction;`

Let us understand: void pointers

- Pointers to a generic chunk of memory
- Programmer needs to know the actual type in order to do anything useful
- Useful for writing generic (type independent) code

```
char c, *cp;
char *str, **sp;
int i, *ip;
float f;

void *voidPointer;
```

```
voidPointer = (void *) &c;
* ((char *) voidPointer) = 'A';
printf("%c\n", c);

voidPointer = (void *) &i;
* ((int *) voidPointer) = 10;
printf("%d\n", i);
```

Generic sort routine

```
#include<stdlib.h>

void qsort(void *base, size_t nmemb, size_t size,
           int (*compar)(const void *, const void *));
```

Comparator routine: examples

```
int compare_int (void *elem1, void *elem2){
    int *ip1 = elem1;
    int *ip2 = elem2;
    return *ip1 - *ip2;
    /* Or more explicitly:
       int i1 = *((int *) elem1);
       int i2 = *((int *) elem2);
       return i1 - i2;
     */
}

int compare_strings (void *elem1, void *elem2){
    char **s1 = elem1; // Alt.: char *s1 = *((char **) elem1);
    char **s2 = elem2; // Alt.: char *s2 = *((char **) elem2);
    return strcmp (*s1, *s2);   // Alt.: return strcmp(s1, s2);
}
```

Using qsort

```
char **strings;
int *a;
int num_strings, N;

qsort(a, N, sizeof(int), compare_int);
qsort(strings, num_strings, sizeof(char *),
                                    compare_strings);
```

Timsort

Timsort is a highly optimized and stable version of merge sort that effectively combines insertion sort.

Timsort was implemented by Tim Peters in 2002 for use in the Python programming language.

- `https://en.wikipedia.org/wiki/Timsort`
- `https://realpython.com/sorting-algorithms-python/#pythons-built-in-sorting-algorithm`

Problems – Day 13

1. The password file on a GNU/Linux machine looks like the example available from `https://www.dropbox.com/s/7x17j7z3alfkfdc/etc-passwd.txt?dl=0`. This file lists the accounts of MTCS students of ISI from the 2017-19 batch. It consists of multiple columns, separated by '`:`'. Columns 1,3, and 5 correspond respectively to a student's roll number, user ID and name.
   Write a C program to read and store this information in the form of an array of structures, with each structure representing a single student.
   Now write appropriate comparator functions so that the `qsort` function can be used to sort the array in
   - ascending order of roll numbers;
   - descending order of roll numbers;
   - alphabetic order by name.