education for the real world

Brute Force

Chapter 3

Brute force method:

- It is a straight forward approach to solving a problem usually directly based on the problems statement and definitions of the concepts involved.
- It is one of the easiest to apply.
- Ex:

1. Computing a^n
2. Computing n!
3. Sequential search

Selection Sort

```
Algorithm SelectionSort(A[0….n-1])
// Sorts given array using selection sort.
// Input: An array A[0…..n-1] orderable elements.
// Output: An array A[0…..n-1] sorted in ascending order.
for i←0 to n-2 do
    min←i
    for j←i+1 to n-1 do
        if A[j]<A[min]
            min←j
    swap A[i] and A[min]
```

Analysis:

- Input size: number of elements n.
- Basic operation: Comparision A[j] < A[min]

$$C(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1$$

$$= \sum_{i=0}^{n-2} [(n-1) - (i+1) + 1]$$

$$= \sum_{i=0}^{n-2} (n-1-i)$$

$$= \frac{n(n-1)}{2}$$

Bubble Sort

```
ALGORITHM  BubbleSort(A[0…n-1])
//Sorts the array using bubble sort.
//Input: An array A[0….n-1] of orderable elements.
//Output: An Array A[0….n-1] in ascending order.
for i←0 to n-2 do
    for j←0 to n-2-i do
        if A[j+1]<A[j]
            swap A[j] and A[j+1]
```

Bubble Sort

- Swaps

 - Worst case and Best Case

- Improvement to Bubble sort algorithm by introducing exchange variable

Sequential Search

```
ALGORITHM  SequentialSearch(A[0….n-1],k)
// Searchs the array using Sequential Search method.
//Input: An array A[0….n-1] of elements and a key element k which is to be
//searched.
//Output: if found, returns the position where the element found else returns -1.
i=0
while  i < n  and  A[i]≠ k do
      i=i+1
if i< n
      return i
return -1
```

Sequential Search

```
ALGORITHM  SequentialSearch(A[0….n-1],k)
// Searchs the array using Sequential Search method.
//Input: An array A[0….n-1] of elements and a key element k which is to be
//searched.
//Output: if found, returns the position where the element found else returns -1.
A[n]← k  // Sentinel element
i=0
while A[i]≠ k do
    i=i+1
if i< n
    return i
return -1
```

Matrix Multiplication

```
//Multiplication of 2 nxn matrices
//Input :Matrices A and B.
//Output: C=A*B
for i←0 to n-1 do
    for j←0 to n-1 do
        C[i,j] ←0
        for k←0 to n-1 do
            C[i,j] ← C[i,j] + A[i,k] * B[k,j]
return C.
```

String Matching

```
ALGORITHM  BruteForceStringMatching(T[0…n-1],p[0…m-1])
//Implements String matching
//Input: text array T of n characters, and pattern array P of m characters.
//Output: Position of first character of pattern if successful otherwise -1
for i←0 to n-m do
    j←0
    while j<m and P[j]=T[i+j]
        j←j+1
    if j=m
        return i
return -1
```

Tracing of String Matching

Text : “WAIT AND WATCH”

Pattern : “WAT”

i	j	P[j]	T[i+j]
0	0	W	W
	1	A	A
	2	T	I
1	0	W	T
2	0	W	
3	0	W	A
4	0	W	N
5	0	W	D
6	0	W	
7	0	W	W
8	0	W	A
9	0	W	T
	1	A	C
	2	T	H
	3		

j=m=3

Return 9

i.e. The Starting position of the substring in the given string.

Pattern P is present in the text T starting at position 9.

The End

Thank You