ARRAY & STRINGS

Date: / / /

Binary Search:- Search in sorted elements.

A	4	8	10	15	18	21	24	27	29	33	34	37	39	41	43
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Search for key = 18, we need three vars l, h, & mid.

$$mid = \left\lfloor \frac{l+h}{2} \right\rfloor$$

$l = 0$, $h = 14$

$$mid = \left\lfloor \frac{0+14}{2} \right\rfloor = 7$$

check A[mid] is key element. Yes terminate

(*) If no whether "key" element is bigger or smaller than A[mid]

In this case smaller hence new high will be

$h = mid - 1$

$= 7 - 1 = 6$

$$mid = \left\lfloor \frac{0+6}{2} \right\rfloor = 3$$

same checks but this time A[mid] is greater hence shift low

$l = mid + 1 = 3 + 1 = 4$

$$mid = \left\lfloor \frac{6+4}{2} \right\rfloor = 5$$

Same chek

if(A[5] == mid) (*)

$h = 4$, $\left\lfloor \frac{l+4}{2} \right\rfloor = \frac{4+4}{2}$

$= 4$.

Date: / /

Another example key = 34

l	h	mid	
0	14	7	
8	14	11	
8	10	9	
10	10	10	→ Terminate key found

Another example key = 25 (* not in array)

l	h	mid	
0	14	7	
0	6	3	
4	6	5	
6	6	6	
7	6		→ ⊗ terminate low has became greater than high.

```
Algo BinSearch(l, h, key) {
    while(l <= h) {
        mid = ⌊(l+h)/2⌋
        if(key == A[mid])
            return mid
        else if(key < A[mid])
            h = mid-1
        else
            l = mid+1
    }
    return -1;
}
```

```
Algo RecBinSearch(l,h,key){
  if(l<=h)
    mid = [l+h/2]
    if(key==A[mid])
  else if(key<A[mid])
    return RecBin(l,mid-1,key)
  else
    return RecBin(mid+1,h,key)
  }
```

(*) Right working Recursive Binary Search Yourself.

Date: / / /

Question :- Check for anagram? Anagram example (decimal & medical) (Verbose & observe)

Assuming all the strings are lower case.

```
int main( ){
char A[] = "decimal";
char B[] = "medical";
int l1 = strlen(A);
int l2 = strlen(B)
if(l1 != l2){
  cout << "Not Anagram"
}
else {
for(i=0; i<l1; i++){
HashArr[A[i]-97]
  ++=1;
}
```

```
HashArr[26] = [0,00,...0]

for(i=0, i<l1; i++){
H[B[i]-97]--=1
if((H[B[i]-97)<0){
    printf("Not Anagram
  }  break;
}
}
```

(97) Ascii for char meter "a".

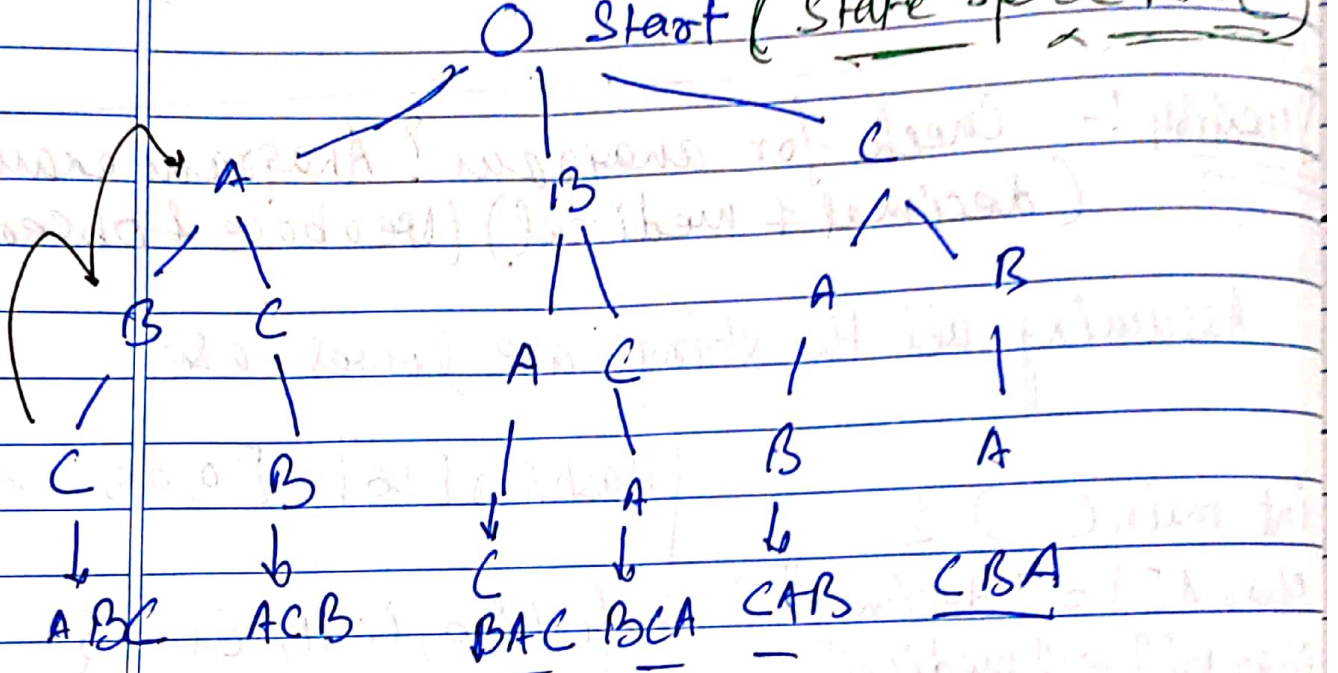
Date / /

Permutation String :-

S = | A | B | C | \0 |

Permutations

1. ABC
2. ACB
3. BAC
4. BCA
5. CAB
6. CBA



[Going back and track called BACK TRACKING]

* All possible permutations = "BRUTE FORCE"

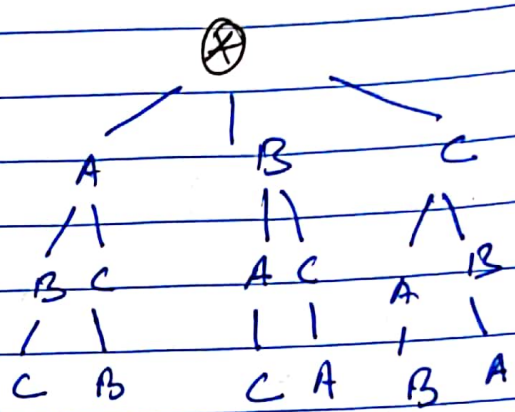
* We can use [Recursion] to achieve backtracking.

* Approach for RECURSION →

Date: / / /

We have S [A|B|C|D] ; k is level tracker for tree level A [0|0|0|0] → status of characters is that selected or not selected.
& final Res [| | |] .
0 1 2 3

(i) stacking array k=0, i=0
(A) A [0|0|0]
take (A) increment (i)




```
void perm(char S[], int k)
static int A[10] = {0};
static char Res[10];
 int i;
if(S[k] == '\0') {
   Res[k] == '\0';
   print(Res)
} else {
for(i=0; S[i] = '\0'; i++){
{  if(A[i] == 0) {
   Res[k] = S[i]
    A[i] = 1;
   perm(S, k+1)
    A[i] = 0;
} ⇒ if
} ⇒ for
} ⇒ else
} ⇒ function close.
```

(8) Make trace tree yourself & execute

```
[ char S[] = "ABC"
  perm(S, 0); ]
```

CALLING || perm.