**RECURSION FUNCTION**

Kendi kendini çağıran fonksiyonlara özyineli (recursive) fonksiyon denilir. Özyineli fonksiyonlar, ileri bilgisayar uygulamalarında çok kullanılır. Bilgisayar biliminin zor sayılan konularından birisidir. Çoğunlukla, döngülerle çözülebilen problemler, özyineli fonksiyonlarla çok daha kolay olarak çözülebilir.

1. ***ÜÇGENLER***

……

…..

…..

int ucgen (int n){

int toplam = 0;

while(n>0){

STANDART FORM

toplam += n;

--n;

}

return toplam;

}

int ucgen (int n){

RECURSIVE FORM

if(n==1){

return 1;

}

else{

return n + ucgen(n-1);

}

}

1. ***FAKTÖRİYEL***

int f (int n){

int s=1;

while(n>0){

STANDART FORM

s\*=n;

n--;

}

return s;

}

int f (int n){

if(n==1){

RECURSIVE FORM

return 1;

}

else{

return n.f(n-1);

}

}

1. ***ANAGRAM***

Anagram, sözcüklerin içinde yer alan harflerin yerini değiştirerek yeni bir kelime öbeği oluşturmaktır. Anagramın olabilmesi için aynı harfleri kullanmak zorunludur. Anagram yaparken anlamlı ya da anlamsız sözcükler oluşturulabilir.

Cat -------< cat , atc , tca , cta , act , tac vb.

anagram(bilgisayar){

b;

} vb. gibi anagramlar üretebiliriz.

anagram(ilgisayar){

string;

}

anagram (int ns){

if(ns == 1){

return;

}

else{

for(int i=0; i<ns; i++){

anagram(ns-1);

if(ns == 2){

system.out.println(ns);

}

}

}

}

1. ***RECURSIVE BİNARY SEARCH***

Sıralanmış bir öğe diziniz olduğunu varsayalım.

İkili arama ile sıralı bir dizinin ortasındaki bir değeri seçeriz, değerimizin seçtiğimiz değerden büyük mü yoksa düşük mü olduğunu buluruz ve sonra değerimizin hangi yarısının parçası değilse onu bırakırız.



İkili arama fikri, dizinin sıralandığı bilgisini kullanmak ve zaman karmaşıklığını O (Log n) 'ye düşürmektir.

int binarySearch(int arr[], int l, int r, int x)

{

    while (l <= r) {

        int m = l + (r - l) / 2;

        if (arr[m] == x)

            return m;

        if (arr[m] < x)

            l = m + 1;

        else

            r = m - 1;

    }

    return -1;

}

1. ***HANOİ TOWERS***

Her hamlede sadece bir disk taşınabilir. Her hamle en üstteki diski direkten alıp diğer bir direğe taşımaktan oluşur. Diğer direkte daha önceden diskler olabilir. Hiçbir disk kendisinden küçük bir diskin üzerine koyulamaz.



hanoi(int n, f, i, t) {

         if (n == 1) {

              System.out.println("Move disk 1 from rod " +  f + " to rod " t);

              return;

         }

else{

hanoi(n-1, f, i, t);

         System.out.println("Move disk " + n + " from rod " +  f + " to rod " + t);

         hanoi(n-1, i, f, t);

}

    }

1. ***MERGE SORT***

Birleştirme sıralaması, böl ve ele geçir yaklaşımını izleyen algoritmadır.



merge (A, B, C) {

int i=0, j=0, r=0;

while (i < A.length && j < B.length){

if(A[i] < B[j]){

C[r] = A[i];

i++;

}

else{

C[r] = B[j];

j++;

}

r++;

}

}