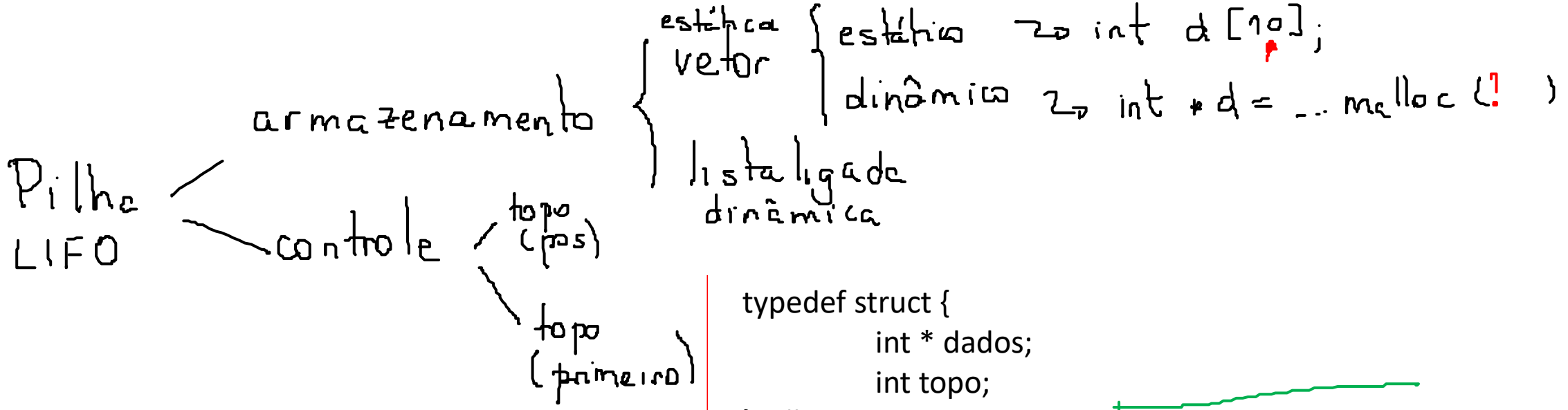
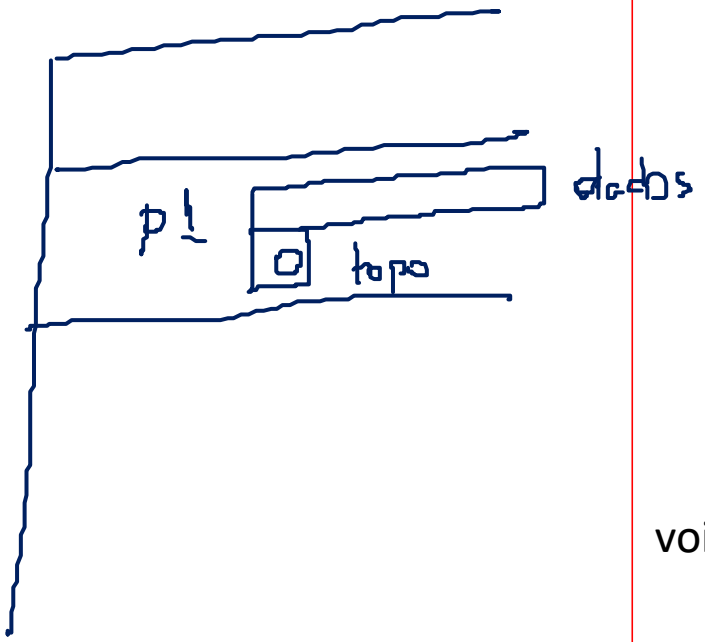
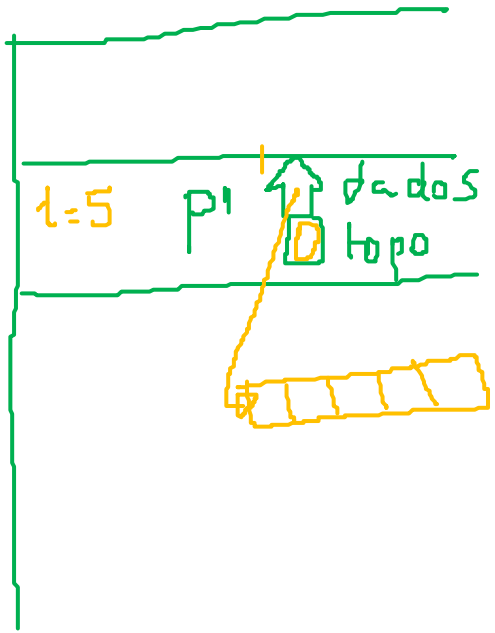


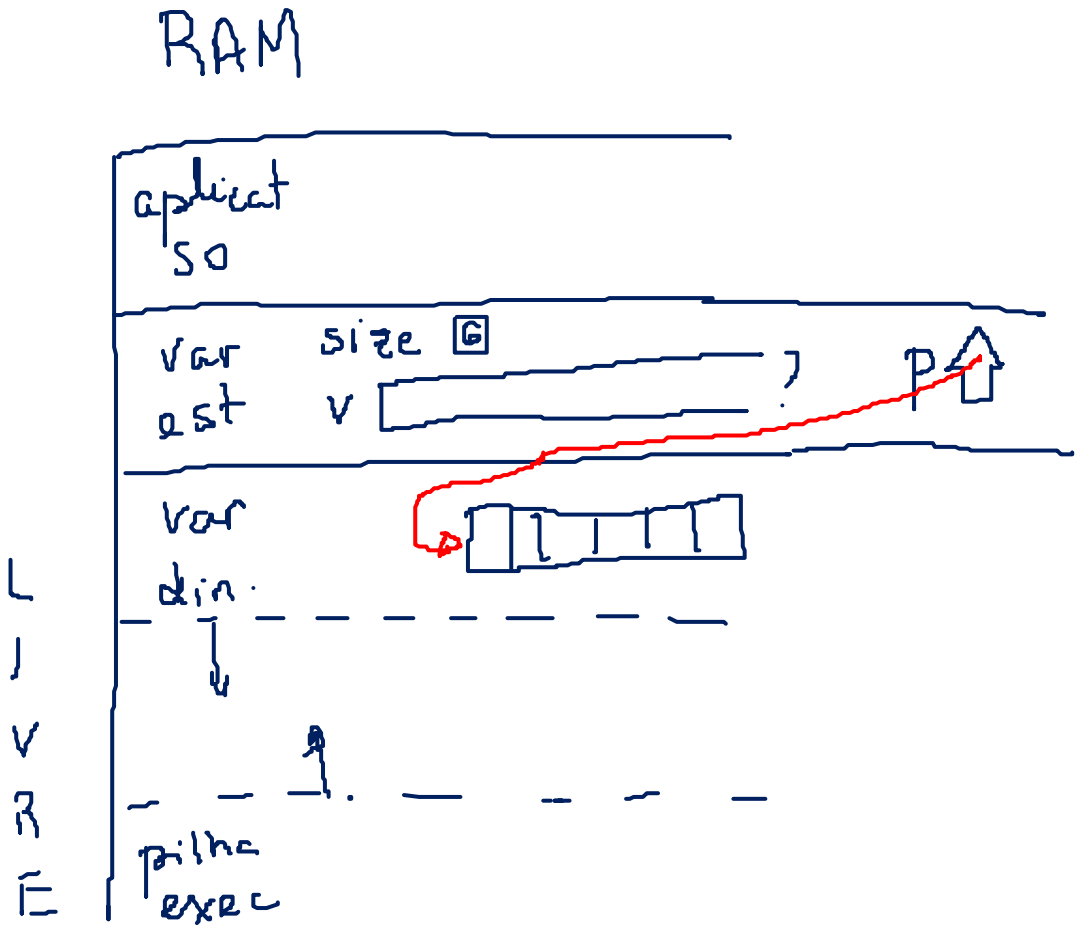

```
typedef struct {
        int dados[10];
        int topo;
} pilha;


void inicia_pilha (pilha *p) {
        p->topo = 0;
}

int main() {
        pilha p1;
        inicia_pilha(&p1);
```




```
typedef struct {
        int * dados;
        int topo;
} pilha;


void inicia_pilha (pilha *p, int t) {
        p->topo = 0;
        p->dados = (int *) malloc (t * sizeof(int));
}
```



```c
int main () {
        int size;
        int v[size]; /* definido em tempo de compilação*/
        int *p;
        printf ("\nsize: %d\n", size); //typo
        scanf ("%d", &size);
        p = (int*) malloc (size * sizeof(int));
        return 0;
}
```



Recursão ——causa——→ repetição ← laços

início
fim → passo

↳ chamada de um procedimento por ele mesmo

1=) critério de parada (fim)

2=) reduzir o problema (passo)

3:) chamada inicial (início)

indução finita

⇒ quando não recursão

Fatorial:

$n! = 1 \times 2 \times 3 \times 4 \times \ldots \times n$

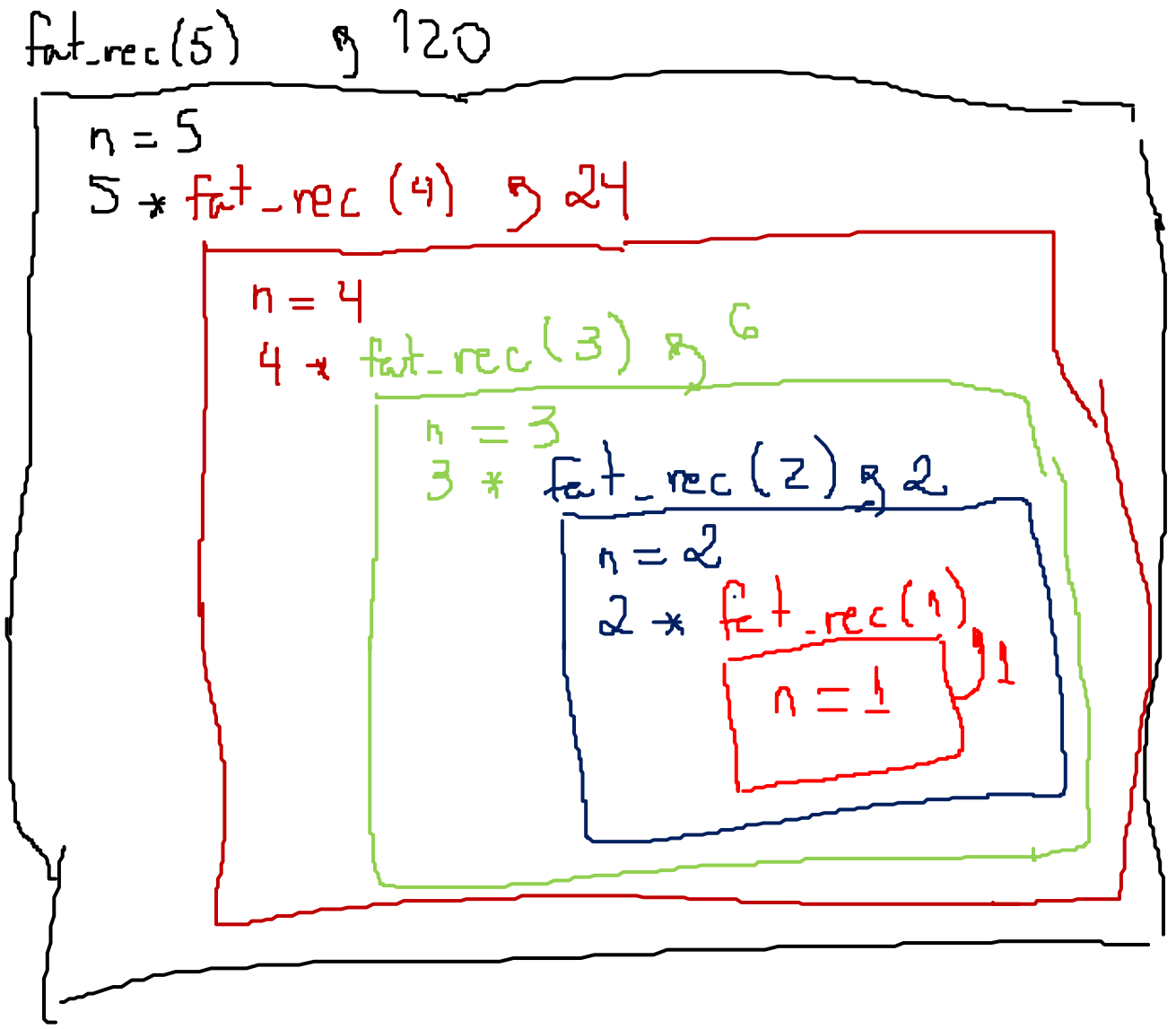
$5! = 5 \times \underbrace{4 \times 3 \times 2 \times 1}_{4!}$

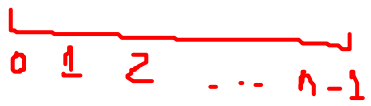
$5! = 5 \times 4!$

$4! = 4 \times \underbrace{3 \times 2 \times 1}_{3!}$

```
int fat_it (int n) {
    int i, f;
    for (f = 1, i = 2; i <= n; i++)
        f = f * i;
    return f;
}

int fat_rec (int n) {
    if (n == 0 || n == 1)
        return 1;
    return n * fat_rec (n-1);
}
```

```
int fat_rec (int n) {
        if (n==0 || n==1)
                return 1;
        return n*fat_rec(n-1);
}
int main () {
        printf ("fatorial de 5 = %d", fat_rec(5));
        return 0;
}
```



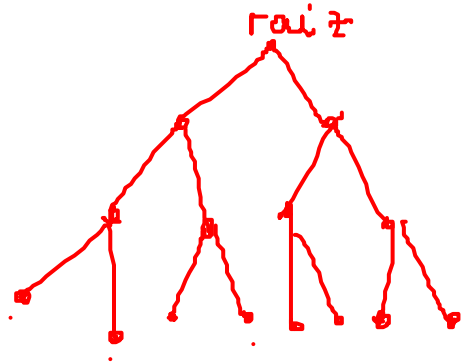
vetor: 

fim de 0 a n-1

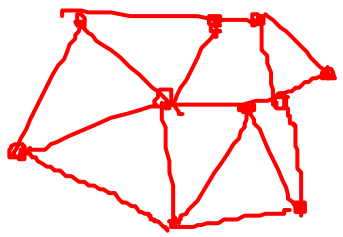
// 
primeiro

while inicio até fim

árvore binária



grafos



exibir um vetor:

```
for (i = 0 ; i < T ; i++)
    printf ("%d", v[i]);
```

```
void mostra_v_rec1 (int v[], int i, int T){
    if (i < T) {
        printf ("%d", v[i]);
        mostra_v_rec1 (v, i+1, T);
    }
}
```

```
: mostra_v_rec1 (v, 0, T);
```

v = [8, 7, 3, 4] mostra (v, 3) } v[3]

i = 3 : mostra (v, 2) } v[2]

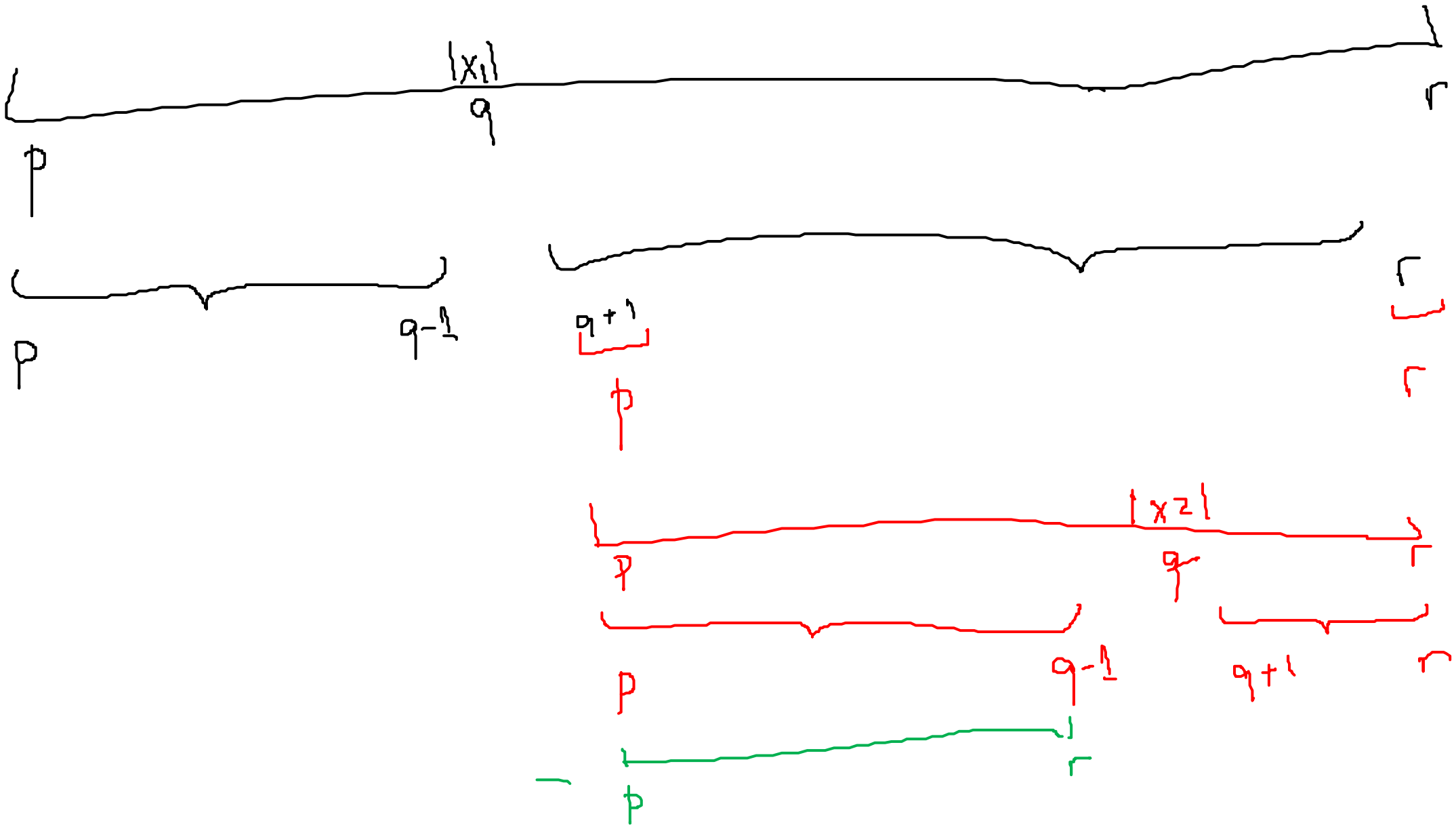
i = 2 : mostra (v, 1) } v[1]

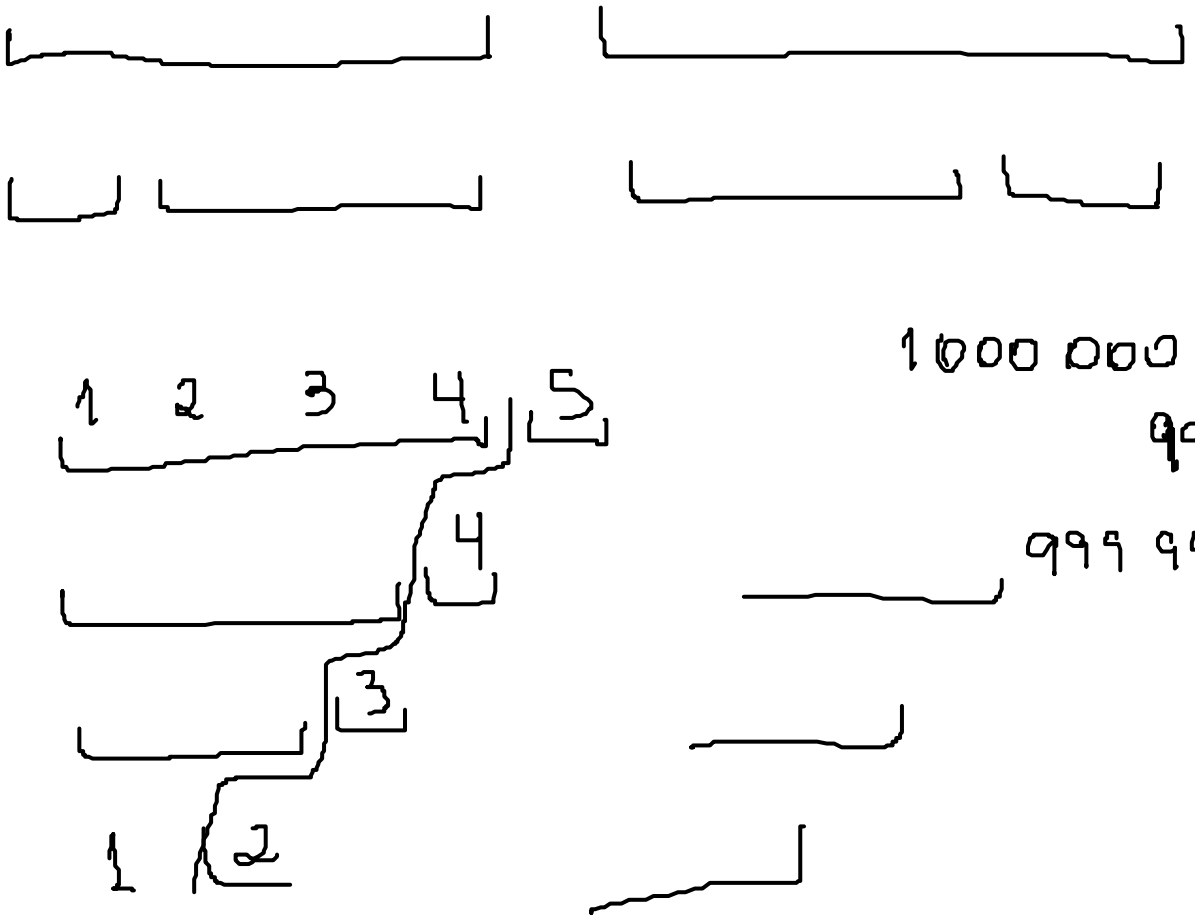
i = 1 : mostra (v, 0) } v[0]

i = 0 : mostra (v, -1) }

i = -1 :

```
void mostra_v_rec2 (int v[], int i){
    if (i >= 0){
        mostra_v_rec2 (v, i-1);
        printf ("%d", v[i]);
    }
}
mostra_v_rec2 (v, T-1);
```


|x1|
q
p
r
p
q-1
q+1
r
r
|x2|
p
q
r
p
q-1
q+1
r
p
r
p


1 2 3 4 5
4
3
1 2
1000000
999999
999998